HeteroReason: Heterogeneous FPGA-GPU Acceleration for Disaggregated Speculative Reasoning

Zehuan Zhang[1], Quan Deng[1,2], Zhibo Ren[1], Hao Mark Chen[1], Guoyu Li[1], Xuchun Hu[1], Jose G. F. Coutinho[1], Ce Guo[1], Wayne Luk[1], Zhiqiang Que[1,3], and Hongxiang Fan[1]
[1]Imperial College London [2]Tsinghua University [3]University of Bristol

Abstract

Large Reasoning Models (LRMs) have achieved state-of-the-art performance in reasoning tasks by utilizing Chain-of-Thought (CoT) reasoning. To achieve fast execution speed, speculative reasoning techniques have been introduced in prior work, which adopt a lightweight draft model for candidate token generation followed by process reward models (PRMs) for verification and a strong target model for refinements. This paper identifies that the existing speculative reasoning paradigm follows a strictly forward-only reasoning trajectory, which lacks robustness and can lead to severe error propagation if early reasoning steps are suboptimal. Furthermore, executing these disparate inference schemes, including sequential drafting and parallel verification on homogeneous GPU platforms, can lead to severe resource underutilization. To address this, we propose HeteroReason, an algorithm-hardware co-designed heterogeneous FPGA-GPU inference paradigm specifically tailored for LRM speculative reasoning. At the algorithmic level, we introduce a backtracking-enhanced workflow that enables the system to recover from low-quality states and explore alternative reasoning trajectories, significantly improving reasoning robustness. At the system level, the draft model is offloaded to the FPGA while deploying the PRM and target models on GPUs. A specialized workflow is optimized to achieve prefill-decode disaggregation, which exploits shadow synchronization to overlap GPU-side refinements with FPGA-side token updates to effectively hide synchronization latency. To mitigate inherent sequential constraints, we propose a step-ahead speculation and refinement scheduling scheme, transitioning the system from a sequential execution scheme to a parallelized pipeline. Experimental evaluations demonstrate that the backtracking mechanism achieves an average reasoning accuracy improvement of up to 4.2% across diverse reasoning benchmarks. Moreover, built on the heterogeneous platforms consisting of AMD U280 or V80 FPGAs, with NVIDIA GeForce RTX 3090 GPU platforms, HeteroReason achieves 1.01×–1.42× latency speedups and 1.25×–1.57× improvements in energy efficiency under different model configurations, compared to homogeneous GPU baselines.

1 Introduction

LLMs have emerged as a revolutionary AI paradigm, exhibiting advanced capabilities across a wide range of applications including natural language processing [8, 18, 19], computer vision [41, 46, 53] and healthcare [42, 63]. Recently, the emergence of reasoning-focused LLMs, Large Reasoning Models (LRMs) [60], such as OpenAI o1 [51]/o3 [50] and DeepSeek-R1 [13], has led to state-of-the-art performance in reasoning tasks [5, 10, 11, 17, 37]. Although LRMs share similar architecture backbones with conventional LLMs and rely on autoregressive token-by-token prediction, their inference process differs: traditional LLMs typically aim for direct response generation [19], while LRMs utilize Chain-of-Thought (CoT) [59] reasoning to decompose a complex task into step-by-step reasoning sequences, yielding the final answer. This paradigm [13, 51] has demonstrated remarkable advancements in reasoning-intensive tasks.

Despite these impressive performance gains, the inference speed of this model family faces an inherent challenge: autoregressive decoding generates reasoning sequences sequentially, which introduces computational dependency and leads to a linear increase in latency with sequence length. The accumulated length of CoTs in LRMs further exacerbates this issue, thus impeding their practical deployment for real-time applications, where delayed responses degrade user experience. To mitigate this issue, speculative reasoning techniques [38, 52] have emerged as promising solutions and have been successfully evaluated in reasoning domains. Currently, it is a growing and active line of research spanning both academic and industry labs [9, 38, 43, 52, 56], with substantial inference speedups underscoring its value in latency-sensitive settings. The core insight is that LRMs solve reasoning tasks by decomposing sequential steps, with each individual reasoning step hinging on semantic insights more than on exact tokens, enabling more tolerant approximations [52]. Therefore, the techniques leverage a smaller lightweight draft reasoning model to autoregressively generate candidate tokens for individual reasoning steps, with a process reward model (PRM) or a more capable but slower target model to verify the speculative steps, guaranteeing the correct trajectory for the whole reasoning process. In this way, the target model confirms or adjusts candidate tokens from the draft model with far lower invocation frequency, thereby significantly boosting inference speed without compromising accuracy [38].

In the standard speculative reasoning paradigm, the algorithm adheres to a strictly forward-only reasoning trajectory in the generation process, exhibiting weak robustness and inefficiency. For instance, if an early reasoning step yields low-quality results that are accepted to the reasoning chain, this may trigger error propagation and degrade overall accuracy since intermediate steps are interdependent. On the hardware aspect, draft models and target models exhibit disparate computational patterns [35]. Specifically, a large portion (up to 80% [52]) of the speculated steps are accepted. Therefore, in most cases, smaller draft models perform memory-intensive autoregressive decoding to generate logic sequences, whereas larger PRMs or target models conduct compute-intensive parallel verification. Furthermore, extensive CoTs produce a substantial amount of lightweight reasoning work, demanding frequent execution of small-scale operations for inference. However, GPUs struggle to

MICRO 2026, Athens, Greece
2026. ACM ISBN 978-X-XXXX-XXXX-X/XX/XX
https://doi.org/XXXXXXX.XXXXXXX

provide an energy-efficient solution for lightweight model inference, and frequent workload switching incurs significant kernel launch overhead. Consequently, there is a critical need to improve algorithmic robustness and develop tailored hardware platforms to unlock the acceleration potential of speculative inference for LRMs.

Although customized hardware accelerators have been investigated as shown in Table 1, *these existing accelerators are exclusively tailored for standard token-wise speculative decoding and strictly adhere to a rigid forward-only trajectory.* They are inherently unsuited for the step-wise speculative reasoning paradigm, which entails significantly more complex computation and dynamic scheduling such as coordinating three disparate models and enabling backtracking mechanism as detailed in Sec. 2. To achieve the goal of joint algorithm-hardware optimization for LRMs, three research challenges remain: **(1) Robustness Bottlenecks in the forward-only reasoning trajectory.** The standard speculative reasoning proceeds in a strictly forward-only manner, lacking a mechanism to recover from early-stage suboptimal generations, which severely limits reasoning robustness. **(2) Mismatched computational patterns across models and inference phases.** The draft model prioritizes sequential autoregressive decoding, whereas the PRM performs parallel verification, and the parameter-heavy target model is invoked less frequently for refinements. More importantly, speculative reasoning follows a step-wise paradigm that interleaves prefilling and decoding phases. Executing these disparate patterns on a homogeneous hardware platform inevitably leads to severe inefficiency and resource underutilization. **(3) Inherent sequential dependencies limit execution speed.** The iterative process of drafting, verification, and refinements is executed in a sequential manner. The launch of each model depends on the results from the preceding stage. This rigid dependency restricts concurrent execution across different stages, leading to pipeline stalls that hinder further acceleration.

Table 1: Comparison against Existing Work.

Framework	Target Workload	Model Inference Pattern	Backtracking -aware	Hardware
LP-Spec [24]	SpecDecoding (token-wise)	Augmented Target	✗	PIM-NPU
SpecPIM [35]		Draft-Target	✗	PIM-GPU
SADDLE [57]			✗	PIM-GPU
DFVG [47]			✗	FPGA-GPU
Ours	SpecReason (step-wise)	Draft-PRM-Target	✓	FPGA-GPU

To address the aforementioned challenges, we propose HeteroReason, an algorithm-hardware co-designed heterogeneous FPGA-GPU system specifically tailored to accelerate LRM speculative inference, as illustrated in Fig. 1. At the algorithmic level, we introduce **(1) a backtracking-enhanced algorithm** to endow the model with backtracking capabilities, enabling it to escape suboptimal local optima and switch to alternative reasoning trajectories. At the system level, we present **(2) an FPGA-GPU heterogeneous system with optimized workflows**. The system offloads the draft model to an FPGA and deploys the PRM and target model on GPUs, which accommodates hardware platform advantages. The execution workflow, featuring shadow synchronization for draft model status updates, is optimized to achieve prefill-decode disaggregation on the local FPGA accelerator. At the scheduling level, we introduce **(3) a step-ahead speculation and refinement scheme with system optimizations.** To mitigate sequential constraints, this scheme allows the draft and target model to speculatively generate tokens for subsequent reasoning steps while the PRM concurrently verifies the current step, which overlaps speculation, verification and refinements, thereby minimizing pipeline stalls and reducing overall latency. Moreover, asynchronous target prefetching and dynamical KV cache management strategies are introduced to optimize GPU and FPGA implementations, further contributing to system speedups. Our main contributions can be summarized as follows:

- A backtracking-enhanced algorithm that enables recovery from suboptimal early generations, effectively improving the robustness and overall accuracy. (Sec. 3)
- A heterogeneous FPGA-GPU system tailored to accelerate LRM speculative inference, with the specialized workflow achieving prefill-decode disaggregation on the local accelerator via a shadow synchronization mechanism. (Sec. 4)
- A step-ahead speculation and refinement strategy that transitions the sequential paradigm to a parallelized pipeline, integrated with system optimizations tailored for GPU and FPGA architectures, enhancing the execution speed and resource utilization. (Sec. 5)

2 Background and Motivation

We introduce the background, the paradigm of reward-guided speculative reasoning (RSD) and analyze its computational characteristics and patterns. Subsequently, we elucidate the motivation for developing a heterogeneous system, and identify the challenges encountered in the efficient execution of speculative reasoning.

2.1 Background

2.1.1 Large Reasoning Models (LRMs). LRMs simulate human-like cognition by prioritizing deliberate reasoning before response generation [28]. This capability is primarily driven by the scaling of inference-time compute, which enables long CoTs for complex reasoning. Specifically, LRMs break a complex problem down into intermediate steps, and then explore potential solutions and verify reasoning paths before reaching a final answer. This paradigm demonstrates superior performance in reasoning tasks such as advanced mathematics [61] and medical diagnostics [31]. The o1 [51] and o3 [50] model series leverage CoT reasoning, while DeepSeek's R1 [13] proves the potential of reinforcement learning in developing thinking capabilities of models. As a result, by employing explicit intermediate processing steps, structured logical reasoning capabilities are enhanced, thereby enabling effective solutions to complex problems. However, the reasoning chains introduce substantial computational complexity, demanding higher computational costs than traditional LLMs, resulting in increased latency.

2.1.2 Speculative Reasoning. The speculation concept derives from computer architecture [1]. Speculative decoding [2, 4, 7, 34, 36, 49] accelerates traditional LLM decoding by using a lightweight draft model to conduct faster but less accurate generation, with a strong target model verifying the candidates. Speculative reasoning generalizes speculative decoding by enabling speculation at the step level,

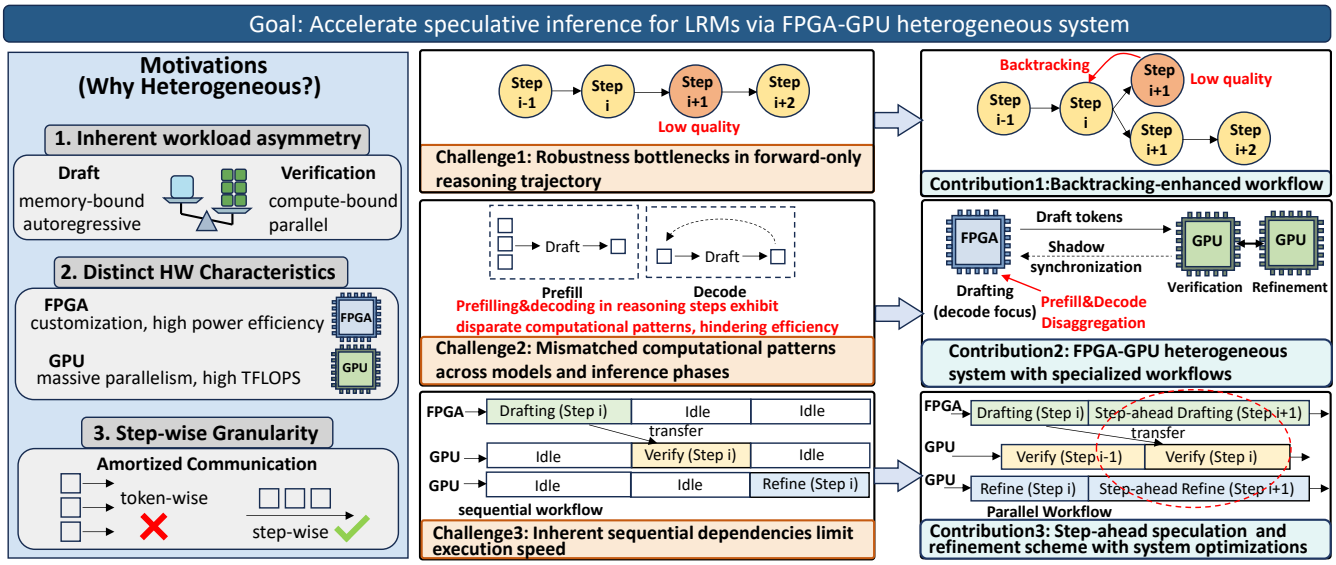

Figure 1: Overview of HeteroReason: Goals, Motivations, Challenges and Contributions.

as the reasoning process does not require the exact reproduction of identical tokens. Unlike speculative decoding, which enforces token-level equivalence, speculative reasoning emphasizes semantic alignment. This property allows reasoning to naturally tolerate approximation [52], offering substantial potential for latency reduction in LRMs. Several explorations have been carried out. RSD [38] utilizes a PRM to assess intermediate steps, and the resulting rewards dynamically determine the invocation of the target model, balancing computational cost and result quality. SpecReason [52] uses a smaller draft model to speculatively conduct intermediate reasoning steps and a larger base model to confirm or refine the speculated results, exploiting the semantic tolerance. Furthermore, the experiments demonstrate that speculative reasoning is complementary to speculative decoding, and these two optimizations can be combined to deliver further latency reductions. Despite these algorithmic advancements, specialized hardware support for speculative reasoning remains unexplored.

2.2 Algorithmic Workflow and Analysis

The step-wise speculative reasoning paradigm operates as an iterative process as illustrated in Figure 2. To formalize this, let x denote the initial prompt as inputs, which are processed by the draft model to generate a candidate reasoning step $\hat{y}_i$. This candidate is then verified by the PRM. Subsequently, the process branches based on the reward score. If the reward score satisfies the criterion, $\hat{y}_i$ is accepted as the valid step y_i. Conversely, if the score falls below the predefined threshold, the candidate is discarded, and the more capable target model is invoked to generate a refined step y_i. Afterwards, the context is updated as $z_i = [x, y_{1:i-1}]$ serving as inputs. The process repeats until completion.

The algorithmic flow involves three distinct models each exhibiting varying characteristics in terms of parameter scale, compute patterns, and invocation frequency. As outlined in Figure 2 (left), the draft model is significantly smaller (e.g., ~1B parameters) than

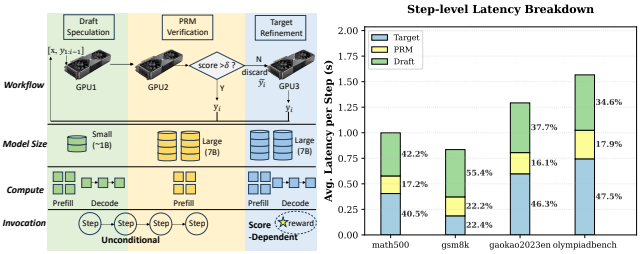

Figure 2: Overview of the speculative reasoning paradigm: the vanilla workflow (left) and the corresponding latency breakdown (right).

both the PRM and target models (e.g., 7B parameters). Regarding operational patterns, the draft model is invoked unconditionally at each reasoning step to perform speculation, necessitating both prefill and autoregressive decoding operations. Similarly, the PRM is invoked unconditionally for step verification, but it requires only prefill operations to assess the speculated sequence. Furthermore, the target model is responsible for refinements, and the invocation is score-dependent. It is invoked only when the reward score falls below a predefined threshold. When triggered, the target model executes both prefill and decoding operations to generate a higher-quality step.

To quantify the latency breakdown, we conducted profiling experiments across four reasoning datasets to analyze the average step-level latency as shown in Figure 2 (right). The results show that draft and target are the main contributors to total latency, accounting for 34.6%–55.4% and 22.4%–47.5%, respectively, while PRM contributes 16.1%–22.2%. Despite its smaller parameter size, the draft model still incurs substantial latency due to frequent unconditional invocations that require both prefill and autoregressive

decoding. The target model also contributes significantly to the overall latency because of its larger model size.

2.3 Motivation of Heterogeneous System

Existing LLMs are predominantly deployed on high-performance GPUs, attaining remarkable performance. Step-wise speculative reasoning introduces a distinct computational paradigm that challenges the efficiency of homogeneous GPU clusters. Our proposed FPGA-GPU heterogeneous system is motivated by the following critical observations:

2.3.1 Inherent Workload Asymmetry. Speculative inference exhibits a fundamental decoupling between token generation and verification. The *lightweight* draft model typically undergoes an autoregressive decoding process to generate candidate tokens, which is inherently sequential and *memory-intensive*. The *parameter-heavy* PRM performs parallel verification of the entire candidate sequence, which is a *compute-intensive* process. If the verification scores fall below a threshold, the system resorts to the target model for refinements, which is *parameter-heavy* but *invoked score-dependently*. As a result, homogeneous GPU platforms struggle to efficiently accommodate the diverse computational characteristics and inherent asymmetry.

2.3.2 Distinct Hardware Characteristics. GPUs excel at parallel processing and high-throughput, arithmetic-intensive workloads, yet they come with considerable power consumption and suffer from resource underutilization and compromised efficiency when handling memory-bound tasks. By contrast, while FPGAs possess lower peak computational capacity than GPUs, they are characterized by superior energy efficiency and architectural reconfigurability, which enables flexible design customization tailored to specific computational demands. The small size of the draft model makes it feasible to fit on an FPGA. Moreover, the autoregressive decoding is fundamentally bottlenecked by memory bandwidth rather than by compute capacity. By harnessing on-chip resources to develop a dedicated accelerator, the FPGA can minimize off-chip memory access overhead and achieve rapid decoding. Consequently, the synergistic integration allows for the exploitation of the full potential of heterogeneous hardware platforms, thereby optimizing hardware utilization and enhancing the overall system performance.

2.3.3 Amortized Communication. A critical bottleneck in distributed inference is the inter-device communication overhead. Completing a comprehensive reasoning task typically necessitates the generation of thousands of tokens. If communication is performed at a token-level granularity, the frequent, small-scale communication incurs prohibitive latency, severely limiting overall system throughput. However, this bottleneck is significantly mitigated in the speculative reasoning paradigm, which performs communication at a step-wise granularity. In typical configurations, each reasoning step generates a sequence of fewer than 200 tokens [3, 38, 52], enabling the effective amortization of communication overheads. For instance, in PCIe-based transfers, the communication latency of a single-step token sequence remains within the microsecond scale. Therefore, by shifting from fine-grained, token-level transfers to coarse-grained, step-level transfers, the inter-device communication costs are rendered marginal relative to the computation costs, thus facilitating efficient heterogeneous execution.

2.4 Challenge Analysis

Achieving high-performance, efficient speculative inference still faces critical challenges in both algorithmic design and hardware support. Existing workflows lack robust error-recovery mechanisms, and no dedicated hardware systems have been developed for efficient implementations.

2.4.1 Robustness Bottlenecks in Forward-Only Reasoning Trajectory. The vanilla workflow adheres to a strictly forward-only generation paradigm. This "speculation-verification-conditional refinements" process introduces a critical robustness bottleneck in complex reasoning. If an early reasoning step receives a low reward score but the system lacks viable alternatives, the workflow is forced to continue with suboptimal context. Consider a scenario depicted in Figure 3: The draft model generates a speculated step with a low reward score. Although the target model is invoked for refinement, it may still produce a refinement step with a low reward score due to the inherent complexity of the prompt or cumulative context noise. In such cases, the system must commit this refined yet low-quality step to the reasoning chain. Consequently, the suboptimal output becomes a permanent fixture of the context, potentially triggering error propagation across all downstream steps.

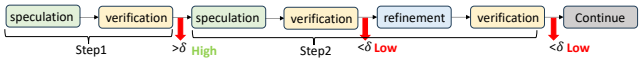

Figure 3: Robustness Bottlenecks in Forward-Only Reasoning Trajectory.

Since reasoning steps are interdependent, any committed suboptimal steps potentially propagate errors through the entire reasoning chain. This vulnerability is exacerbated by the inherent fidelity gap of the draft model. Due to their smaller parameter scale and the application of approximate optimization techniques (e.g., quantization), the draft model is not able to consistently guarantee high-quality outputs across diverse task difficulties. However, the vanilla workflow lacks a mechanism to change previous potentially low-quality steps.

2.4.2 Intra-Device Prefill-Decode Interleaving of the Draft Model. The PRM and target model are characterized by large parameter sizes and high computational intensity. The target model executes both prefill and decoding phases when triggered, while its invocation is conditional. Consequently, both models are deployed on the GPU to leverage superior parallel processing capabilities. For the lightweight draft model, a dedicated FPGA-based accelerator is developed for efficient speculation. However, the current speculative reasoning paradigm necessitates a local prefill operation for the draft model at each reasoning step to continue the decoding process. This interleaving of prefill-decoding operations poses significant hardware design challenges due to their fundamentally distinct computational characteristics.

2.4.3 Inherent Sequential Dependencies Limiting Pipeline Parallelism and Resource Utilization. The standard speculative reasoning paradigm follows a sequential dependency: the draft model on the FPGA generates candidates, followed by the PRM and target model on the GPU for verification and refinements. This step-wise execution leads to inter-device idle time, resulting in severe resource underutilization across the heterogeneous platform. Specifically, the GPU remains idle while awaiting speculative tokens from the FPGA, and similarly, the FPGA stalls while awaiting feedback from the GPU. Consequently, the total latency of each iteration equals the sum of the execution time on separate devices plus the communication overhead. To mitigate the resource underutilization and boost the speed, it is imperative to design a parallel pipeline scheduling scheme that overlaps the drafting process on the FPGA with the verification and refinements on the GPU, thereby alleviating bubbles and minimizing end-to-end system latency.

3 Backtracking-Enhanced Reasoning

To resolve the robustness bottleneck, we propose a backtracking mechanism to decouple the reasoning process from suboptimal local optima. The key insight is that if both the draft and target models fail to yield high-quality results at the current step, the bottleneck likely resides in the preceding context. By maintaining a structured history of accepted states, the system can selectively backtrack to earlier "anchor points" to explore alternative reasoning trajectories. The backtracking works as follows as shown in Figure 4.

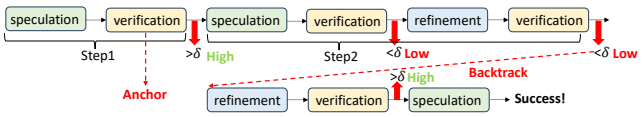

Figure 4: Backtracking Mechanism

The backtracking mechanism is activated only when both the draft and target model produce a low-reward step. This ensures that backtracking is reserved for scenarios where the current context is identified as fundamentally problematic, thereby preventing unnecessary computational overhead. Upon triggering, the system identifies valid anchors by traversing the reasoning history backward. A step is considered a valid anchor if it was previously accepted based on the draft model's speculation without target intervention. This criterion ensures that we backtrack to positions where the PRM deemed the draft output inherently satisfactory. For a failure at step t, the algorithm searches for such ancestors within a lookback window of K steps (e.g., $t-2, t-3$). Selecting an anchor at $t-n$ allows the system to regenerate the sequence from $t-n+1$, effectively purging the corrupted context and providing a fresh logical foundation for downstream steps. By restricting anchors to non-intervened steps, the workflow prioritizes the exploration of genuinely novel reasoning paths originating from draft-accepted states with high reward scores, rather than repeatedly refining the same failed trajectory. To bound the inference latency and computational costs, the total number of backtracking attempts per problem is capped at 5, providing a principled trade-off between exhaustive search space exploration and inference efficiency.

4 FPGA-GPU Heterogeneous System

4.1 System Overview

The design overview of HeteroReason is illustrated in Fig. 5, where an FPGA-based accelerator and GPUs are interconnected via the PCIe bus. In this heterogeneous setup, the draft model is deployed on the FPGA, which is specifically optimized to minimize per-token latency during the decoding phase. The PRM and the target model are deployed on separate GPUs for parallel verification and refinements of the speculative sequences. These platforms exchange initial states, speculative sequences, and feedback through the PCIe.

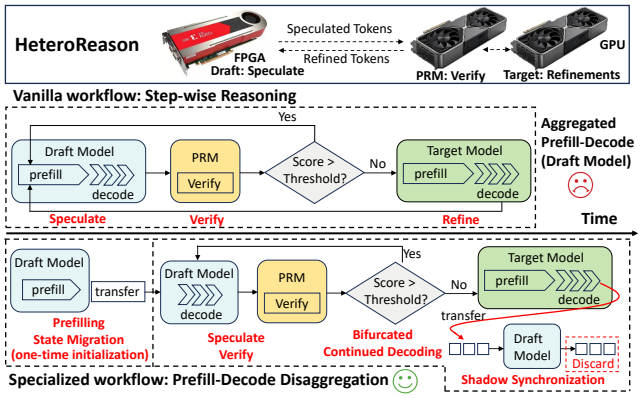

Figure 5: System Overview of HeteroReason.

This system achieves a synergistic workload-hardware match. The drafting and verification workloads are disaggregated and offloaded to distinct platforms suited for the arithmetic density. This disaggregation paradigm effectively resolves the resource underutilization caused by mismatched compute characteristics, typically encountered in homogeneous setups. Furthermore, by exploiting a customized FPGA design for the decoding phase of the draft model, the system harnesses the advantages of heterogeneous acceleration, achieving superior energy efficiency compared to GPU-only solutions.

4.2 Prefill-Decode Disaggregation Paradigm

To overcome the challenges stated in Section 2.4.2 and achieve disaggregation of computational patterns, a specialized workflow is optimized for the heterogeneous speculative inference system. As illustrated in Figure 5, the workflow consists of three distinct phases: (1) Prefilling and State Migration, (2) Speculating and Verification, and (3) Bifurcated Continued Decoding.

Phase 1: Prefilling and State Migration. The workflow begins with the prefilling stage on the GPU. Upon processing the initial prompt, the GPU generates the first predicted token along with its corresponding KV cache. Subsequently, the system performs state migration. The initial KV-cache for the draft model is transferred from the host memory to the FPGA's HBM via the PCIe bus. This phase initializes the speculative environment, enabling the FPGA to begin the iterative decoding process using its customized dataflow pipeline for low-latency drafting. Note that this initialization occurs only once per reasoning task, the one-time transfer latency is rendered negligible relative to the overall generation time.

Phase 2: Speculating and Verification. The FPGA executes the drafting process until the current step is completed. The resultant sequence of speculative tokens is submitted to the PRM on the GPU to conduct a single-step parallel verification.

Phase 3: Bifurcated Continued Decoding. Depending on the PRM reward scores, the system dynamically branches into two execution paths to enable continued decoding. In the case where the draft tokens satisfy the acceptance criteria, the FPGA's KV-cache already preserves the hidden states of both the context z_i and the newly accepted tokens $\hat{y}_i$. Therefore, the draft model immediately resumes the decoding process for the subsequent step y_{i+1}. This seamless continued decoding bypasses redundant prefill operations for the validated prefix.

If the draft tokens are rejected, the system enters the refinement path. The erroneous tokens are discarded, and the target model on the GPU is invoked for refinements. In this scenario, the draft model is precluded from continuing because the autoregressive decoding is strictly dependent on the integrity of the previous KV cache, but the current KV cache contains invalid states corresponding to the rejected tokens. The target model performs a prefill operation using the validated context z_i to generate refined tokens. In the vanilla workflow, the FPGA-based draft model performs a local prefill to update its KV cache for the subsequent computation.

To mitigate this bottleneck, we implement shadow synchronization, which overlaps the GPU's refinement phase with the FPGA's state updates. Upon receiving a rejection signal, the FPGA immediately triggers a *pointer rollback* in its KV-cache management unit, reverting the state to the last validated token. As the target model autoregressively produces refined tokens on the GPU, these tokens are streamed back to the FPGA via PCIe. Simultaneously, the draft model performs a forward pass to ingest the refined tokens and update its KV cache, discarding the output logits. Since the inference latency of the draft model is significantly lower than that of the target model, the FPGA's state update can be fully overlapped with the refinement process on the GPU, effectively hiding the synchronization latency. This process acts as a *shadow* to the refinement phase on the GPU, hence the name *shadow synchronization*. This ensures that the FPGA's drafting environment is fully synchronized and ready for the next drafting cycle by the time the target model completes its reasoning step. Note that if the backtracking mechanism is triggered, the reasoning process reverts to the preceding step, where the target model is invoked to provide a high-fidelity alternative. In such scenarios, shadow synchronization applies as well. Specifically, the reverted sequence position and its corresponding KV cache pointers are transferred to the FPGA, enabling updates of internal states based on the feedback from the target model, ensuring that subsequent speculation aligns with the refined context. While this backtracking mechanism enhances algorithmic robustness, it inherently introduces computational redundancy. Consequently, the frequency of backtracking attempts is strictly constrained, as introduced in Section 3, to balance reasoning quality with system throughput.

Overall, the specialized workflow adheres to the principle of prefill-decode disaggregation, which dedicates the FPGA-based accelerator to decoding in the drafting phase while the GPU handles parallel prefilling and verification, thus circumventing the architectural mismatch. This disaggregated approach allows for independent scaling of drafting, verification and refinements.

5 Step-Ahead Speculation and Refinement

To address the sequential dependency challenges identified in Section 2.4.3, we propose a step-ahead speculation and refinement strategy with system optimizations. Our core insight is that the idle states of both the draft and target models, typically caused by awaiting verification results, can be leveraged to preemptively advance the subsequent reasoning step. By initiating the next iteration concurrently with verification and inter-device communication, the sequential execution is transformed into a parallel pipeline, maximizing hardware utilization. As illustrated in Figure 6, after the speculated tokens for the current step are submitted to the PRM for verification, the FPGA-based draft model and the GPU-based target model continue decoding for the subsequent step without stalling.

5.1 Asynchronous Target Prefetching&Caching

To boost the overall performance, an asynchronous prefetching and caching strategy is proposed for the target model on the GPU. Specifically, the target model initiates generation for the next potential reasoning step immediately upon completion of the current step, rather than waiting for the verification outcome. By triggering the target model earlier in the pipeline, a portion of its inference latency is hidden. Since the target model is typically slower than the draft model, it may prefetch only a partial generation. When prefetch execution is interrupted because a speculative step is accepted, the generated tokens from the partial target run (without KV-cache) are cached. These cached tokens facilitate efficient refinements and support the backtracking mechanism. When refinement or backtracking operations are triggered, the system reuses the partial outputs whenever the current context shares common prefixes with the cached sequences. Therefore, this caching mechanism is crucial for making refinements efficient, with which refinement operations leverage prior computations rather than incur full target model inference cost, effectively minimizing the computational overhead of error recovery.

5.2 Scheduling

With these system optimizations, the proposed scheduling scheme yields speedups across three scenarios.

Accept Scenario: If the PRM validates the current step with a high reward score, the step-ahead speculation proceeds until the next speculative step is completed. Simultaneously, the step-ahead refinement continues on the GPU. In this case, the latency of each reasoning step is determined by the maximum of the drafting time and the verification time plus communication overhead.

Reject-Refinements Scenario: Conversely, if the PRM assigns a low score to the current step, the step-ahead speculative tokens are immediately discarded. The step-ahead refinement continues to produce high-quality replacement tokens, while the draft model on the FPGA performs shadow synchronization to update its KV-cache using the refined tokens from the target model. Given that the draft model's inference latency is significantly lower than that of the target model, this synchronization overhead is fully overlapped,

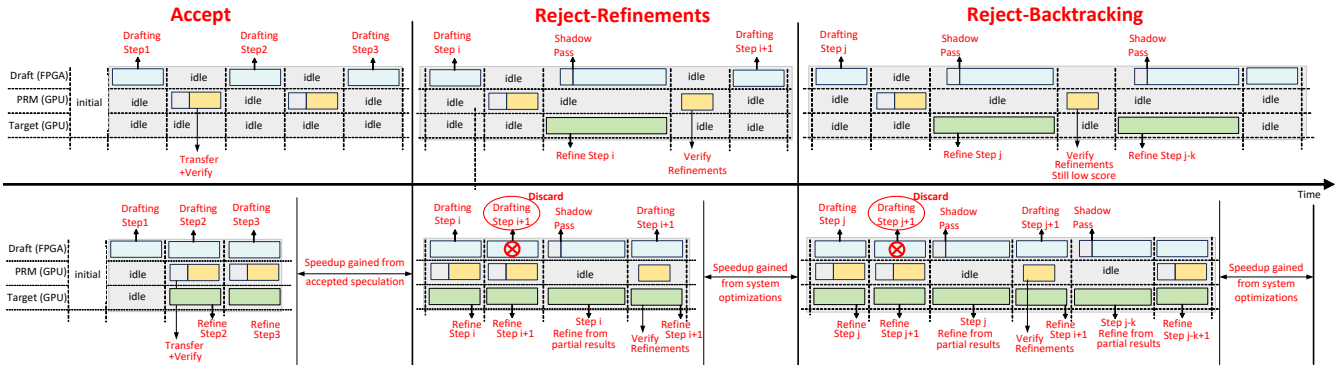

Figure 6: Step-Ahead Speculation and Refinement Scheduling Scheme.

ensuring the FPGA is prepared for the next iteration without additional stalls. In this scenario, the reuse of cached partial results contributes to the acceleration of the refinement process.

Reject-Backtracking Scenario: If the refined step still fails to satisfy the verification criteria, the backtracking mechanism is activated, reverting the process to a preceding step. The target model then updates the historical states, while the FPGA synchronizes its internal KV-cache accordingly. Similar to the refinement scenario, the speedup here is also driven by the reuse of cached partial results from the target model's preemptive execution.

Overall, by replacing idle hardware cycles with step-ahead speculation and refinement, this scheduling scheme effectively mitigates hardware bubbles and improves the utilization, leading to latency reductions across all possible scenarios.

5.3 Design of FPGA-based Accelerator

In the proposed heterogeneous system, the lightweight draft model is offloaded to the FPGA side to autoregressively generate speculative tokens. As the draft model is typically around the 1B parameter scale, rendering both the model weights and the dynamic KV cache can be fully accommodated within the FPGA's off-chip memory. To achieve low per-token latency, the accelerator employs a fully-streaming architecture with the hybrid mapping strategies proposed in FlexLLM [65], completing the decoding process without buffering intermediate activations in off-chip memory.

The proposed scheduling scheme requires dynamical adaptation according to reward scores from PRM. To facilitate this responsiveness without incurring significant latency penalties, we propose a lightweight Key-Value Cache Management Unit (KVMU). As depicted in Figure 7, the KVMU employs a triplet of pointer registers to autonomously track the KV cache state: the Validated Pointer (Ptr_{valid}) represents the last verified context boundary, the Current Pointer (Ptr_{curr}) tracks the current verification step, and the Step-Ahead Pointer (Ptr_{ahead}) marks the step-ahead prediction frontier.

To ensure seamless KV-cache management, the KVMU controller operates in four distinct modes, closely coupled with the scheduling phases: **Initialization Mode**: This mode is activated during the initial system setup or upon triggering the backtracking mechanism. In this state, the three pointers (Ptr_{valid}, Ptr_{curr}, and Ptr_{ahead}) are reset or updated to specific addresses corresponding to the initial prompt or the reverted reasoning position. This re-synchronization

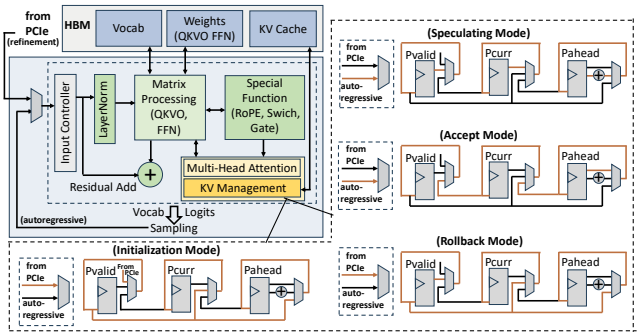

Figure 7: Overview of the FPGA-based decoding accelerator and its four operational modes: (a) Initialization, (b) Speculation, (c) Accept, and (d) Rollback. The internal MUX and KVMU pointer logic adapt to different modes, ensuring synchronization via single-cycle state transitions.

ensures that the internal states are correctly restored before resuming speculative forward progress. **Speculation Mode**: During normal autoregressive speculating, the verified boundaries (Ptr_{valid} and Ptr_{curr}) remain stationary. Only Ptr_{ahead} is advanced by the local incrementer as the draft model generates speculative tokens, pushing the frontier of the speculatively cached KV entries. **Accept Mode**: When the FPGA receives an Accept signal, signifying a successful verification of the last speculative step, the KVMU updates pointers. Ptr_{valid} is updated with the value of Ptr_{curr}, and Ptr_{curr} is concurrently updated to the Ptr_{ahead}. The system then seamlessly resumes speculative forward progress. **Rollback Mode**: Upon receiving a Reject signal, signifying a misprediction, the system instantly executes a pointer rollback to discard the invalid speculatively generated history. Ptr_{valid} is directly reloaded into both Ptr_{curr} and Ptr_{ahead} registers.

In addition, as shown in Figure 7, the input source selection is also dynamically managed to align with the operational modes of the KVMU controller. During **Speculation** and **Accept** modes, the accelerator maintains the autoregressive pipeline by sourcing tokens from the local sampling unit. This internal feedback loop ensures that the FPGA can continuously generate speculative tokens without the latency overhead of host-device communication. In

contrast, **Initialization** and **Rollback** modes necessitate synchronization with the host. In these states, the input MUX is redirected to the PCIe interface, fetching high-fidelity tokens generated by the target model. This allows the accelerator to rapidly transition from a mispredicted speculative path to a refinement path. Thus, the subsequent drafting phase is initialized with the correct context, ensuring that the speculation process remains anchored to the high-quality reasoning chain.

This proposed KVMU design enables efficient, lightweight KV cache management that directly empowers the heterogeneous scheduling. Furthermore, the KVMU is designed as a modular, plug-and-play component, facilitating its seamless integration into existing FPGA-based LLM acceleration architectures.

6 Experiments

6.1 Experimental Setup

Implementation and Hardware. HeteroReason is evaluated on FPGA and GPU platforms. We consider two AMD FPGA boards, U280 and V80, and adopt NVIDIA RTX 3090 GPU platforms. Table 2 presents the specifications of the hardware platforms employed in our experiments. The FPGA designs are built on TAPA [20] and adopt W8A8 quantization. We perform synthesis and P&R using Vitis HLS 2022.2. The power consumption is estimated based on implementation reports.

Table 2: Hardware Specifications

Metric	RTX 3090	U280	V80
Compute units	328 Tensor Cores	9,024 DSPs	10,848 DSPs
Frequency	1695 MHz	200 (300*) MHz	300* MHz
Technology	8 nm	16 nm	7 nm
Peak compute	35.6 FP32 TFLOPS	8 FP32 TFLOPS	58 FP32 TFLOPS
Memory	24 GB GDDR6X	8 GB HBM2	32 GB HBM2e
Bandwidth	936 GB/s	460 GB/s	820 GB/s

*Due to the license issue of TAPA [20] and RapidStream [21, 22], the FPGA performance is projected to 300 MHz.

Models. To thoroughly evaluate the performance of our proposed system in handling diverse reasoning tasks, we employ the Qwen2.5-Math-Instruct model family [62]. The selection of the PRMs and target models follows the configurations in [38]. Two configurations are evaluated: (1) 0.5B draft model - 7B PRM - 7B target model (2) 0.5B draft model - 7B PRM - 1.5B target model.

In the baseline configurations, the draft model, the PRM and target model are distributed across separate GPUs. The popular framework vLLM 0.9.2 is used as the inference engine. In our design, we replace the GPU-based draft model with our FPGA implementation and incorporate the proposed optimizations. Communication between the hardware platforms is handled via the PCIe bus. In alignment with interactive edge computing scenarios, all experiments are conducted using a batch size of 1 to evaluate real-time inference performance.

Hyperparameter Settings. The settings of hyperparameters follow the methodology in RSD [38], we set both the PRM acceptance threshold and the backtracking trigger threshold to 0.7. To bound the computational overhead during error recovery, we cap the total backtracking attempts per problem at 5.

Datasets. We conduct evaluations on four representative datasets of varying difficulty to benchmark mathematical and logical reasoning capabilities:

- **Math500** [25]: A subset of the MATH dataset consisting of 500 challenging competition-level problems.
- **GSM8K** [10]: A collection of grade school math word problems requiring multi-step linguistic and numerical reasoning.
- **Gaokao2023EN** [39]: English-translated problems from the 2023 Chinese College Entrance Examination, reflecting high-school level proficiency.
- **OlympiadBench** [23]: A rigorous benchmark featuring International Olympiad-level problems to test extreme reasoning limits.

Metrics. To comprehensively evaluate the proposed framework, we employ four key metrics to assess both algorithmic and hardware performance. (1) **Accuracy (%):** Measured as the percentage of correctly answered problems across the datasets. We track this metric across different configurations to quantify the reasoning improvements brought by the backtracking mechanism, and to verify that system-level optimizations (e.g., prefetching & caching) do not compromise algorithmic fidelity. (2) **Latency (s/problem):** We measure the average end-to-end time taken to complete the entire reasoning process for a single problem. This metric directly reflects the user-perceived responsiveness, which is critical for real-time interactive applications. (3) **Goodput (Tokens/s):** We calculate goodput as the total number of accepted generated tokens divided by the end-to-end execution latency, which demonstrates the pipeline efficiency under the step-ahead speculation and refinement scheme. (4) **Energy Efficiency (J/Token):** The energy efficiency is defined as the total system energy consumption integrating the operating power of both the FPGA and GPU over the execution time divided by the total generated tokens. This metric assesses the average energy cost per token and demonstrates the power savings of the FPGA-GPU heterogeneous system against homogeneous GPU baselines.

FPGA On-Board Measurement. We evaluate the on-board performance on the AMD U280 FPGA running at 200 MHz. These results are used to cross-validate with our simulator and performance model. Achieving P&R at 300 MHz requires RapidStream [21, 22], which cannot be accessed due to licensing constraints. Following [65], the measured performance is scaled by projecting the clock frequency at 300MHz, and the performance on AMD V80 FPGA is estimated.

Simulator. To evaluate the end-to-end performance of the proposed heterogeneous architecture, we developed a specialized trace-driven simulator. The hardware execution latencies are derived from rigorous profiling: the FPGA accelerator's performance model is cross-validated with the on-board performance, while the GPU inference latencies are measured from empirical implementations. The inter-device communication overhead is modeled analytically by dividing the transferred data volume by the effective PCIe bandwidth. To ensure a fair and highly accurate evaluation, the simulator strictly replays execution traces 1:1 from the 3-GPU baseline. Specifically, we log the exact reasoning trajectory for each problem, including the total number of reasoning steps, the number of generated tokens per step, and the PRM's precise accept/reject decisions,

and deterministically map them to our simulated environment. Furthermore, the simulator fully integrates our proposed step-ahead speculation and refinement scheduling strategy, accurately capturing the parallel pipeline execution strategy to calculate the final system latency and goodput.

6.2 Algorithmic Experimental Results

Table 3 presents the accuracy evaluation across various reasoning benchmarks under two distinct model configurations, executed on a 3-GPU baseline using BF16. The integration of the backtracking mechanism (BRSD) consistently yields notable accuracy improvements of +2.0% and +5.0% over the forward-only RSD baseline across all tested datasets. Furthermore, the complete system (BRSD + optimizations), which incorporates full system-level optimizations, maintains these accuracy gains of average improvements of +1.7% and +4.2%, respectively. This demonstrates that our system optimizations effectively accelerate inference with minimal accuracy degradation (< 0.8% vs. backtracking alone), rendering the approach bringing accuracy gains from backtracking to remain highly practical.

Crucially, the experimental results reveal that the accuracy gain derived from backtracking is significantly more pronounced when employing a less capable target model. As shown in Table 3, Config 2 with a lightweight 1.5B target model achieves a +4.2% average improvement under the fully optimized system, whereas Config 1 with a powerful 7B target model yields a +1.7% gain. This disparity indicates when the target model has limited capacity, its refinement generations are inherently more susceptible to logical deviations and are more likely to fail in fully correcting the draft model's errors. Under the vanilla forward-only workflow, the system is forced to commit to these suboptimal refinements, which permanently corrupt the subsequent reasoning chains. By enabling the system to escape from these suboptimal states and backtrack to alternative promising paths, this mechanism effectively compensates for the target model's capacity limitations.

Table 3: Accuracy results (%) across model configurations. Δ shows change vs. RSD baseline. Bold indicates best per column.

Method	Gaokao	GSM8K	MATH500	Olympiad	Avg	Δ Avg
Config1: 0.5B draft - 7B PRM - 7B Target						
RSD (baseline)	56.9	87.3	64.8	26.5	58.8	–
BRSD	**58.4**	**89.5**	67.0	**28.6**	**60.8**	**+2.0**
BRSD + optimizations	57.9	89.3	**67.2**	27.9	60.5	1.7
Config2: 0.5B draft - 7B PRM - 1.5B Target						
RSD (baseline)	42.9	71.3	50.2	17.0	45.3	–
BRSD	**46.0**	78.0	**56.8**	**20.7**	**50.3**	**+5.0**
BRSD + optimizations	45.2	**78.6**	54.8	19.7	49.5	+4.2

6.3 Hardware Experimental Results

6.3.1 End-to-end Results. Fig. 8 presents a comprehensive comparison of latency, goodput, and energy efficiency between our HeteroReason system and two homogeneous-GPU baselines across four diverse reasoning datasets. The weak baseline only adds backtracking to the vanilla workflow, while the strong baseline further incorporates the step-ahead speculation and refinement scheduling strategy. Overall, our heterogeneous FPGA-GPU systems, equipped with either a U280 or V80 FPGA, both outperform the weak baseline, delivering lower end-to-end latency, higher throughput, and substantial gains in energy efficiency. In particular, the system equipped with the V80 FPGA further narrows the performance gap with the strong GPU baseline.

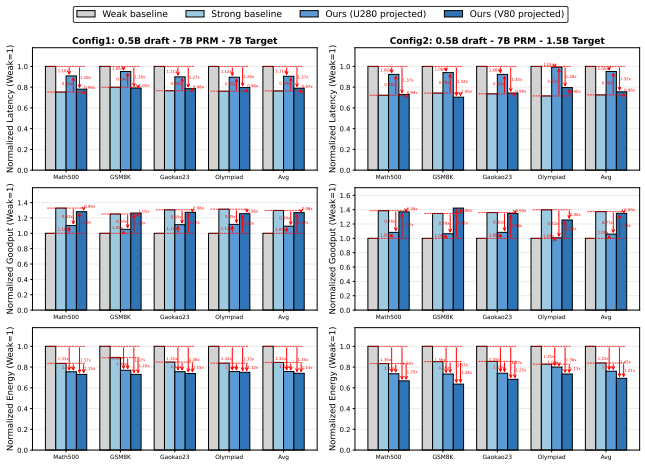

Figure 8: Comparisons of Latency, Goodput and Energy Efficiency against Weak and Stronger Baselines.

Latency Reduction. As illustrated in the upper panels of Figure 8, our heterogeneous implementations reduce average latency by about 1.01×–1.12× (U280), and 1.26×–1.42× (V80) over the weak baselines, while the V80-equipped system achieves end-to-end latency comparable to the strong baseline. This performance advantage is consistent across all benchmarks. The speedup is primarily attributed to the prefill-decode disaggregation paradigm and the step-ahead speculation and refinement scheduling scheme, which enable a parallel pipeline associated with a decoding-only process for the draft model and reuse of generated tokens for the target model. In contrast, in the GPU baselines, draft models conduct both prefilling and decoding phases for each reasoning step, with the workflow bounded by strict sequential dependencies, thereby limiting execution speed.

Goodput Enhancement. The middle panels of Figure 8 exhibit the system goodput. Our heterogeneous implementations improve the goodput by 1.01×–1.12× (U280) and 1.25×–1.42× (V80) over the weak baselines, while the V80-equipped system achieves goodput comparable to the strong baseline. By preemptively advancing the subsequent reasoning step, the system significantly mitigates inter-device idle states and improves hardware utilization.

Energy Efficiency Gains. The energy efficiency advantages are evaluated by comparing the consumed energy per token, as shown in the lower panels of Figure 8. Our systems achieve about 1.3× and 1.1× higher energy efficiency than the weak and strong baselines. These efficiency gains primarily stem from hardware-tailored optimizations on the FPGA, as its operating power consumption is significantly lower than that of the GPU.

Overall, the experimental results demonstrate that our FPGA-GPU heterogeneous architecture successfully bridges the gap between disparate computational patterns, providing a highly scalable and sustainable solution for speculative reasoning.

More experiments are conducted for further analysis. Unless otherwise specified, the following experiments are conducted based on the heterogeneous system equipped with the U280.

6.3.2 Comparisons with RSD. To evaluate the system-level efficiency and performance overhead of our proposed backtracking-enhanced algorithm, we conduct a comparative analysis against the vanilla RSD framework. We deploy our backtracking algorithm on both a standard homogeneous 3-GPU baseline and our proposed HeteroReason system. The normalized speedups relative to the RSD baseline are illustrated in Figure 9.

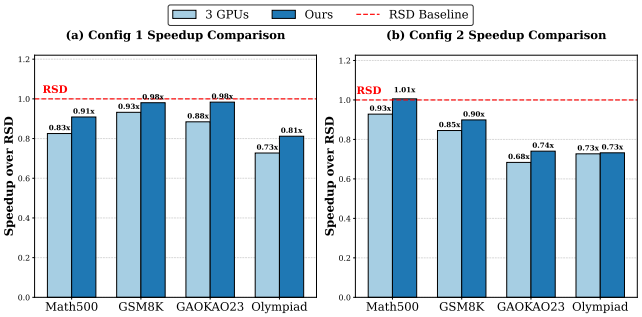

Figure 9: Latency Comparisons against the RSD Framework.

When executed on a homogeneous GPU platform, the introduction of the backtracking mechanism inherently degrades execution speed. As shown in Figure 9, the 3-GPU setup consistently falls below the RSD baseline, achieving 0.73× to 0.93×, and 0.68× to 0.93× the speed of RSD in Config 1 and Config 2, respectively. This performance drop highlights a critical limitation: while backtracking successfully recovers reasoning accuracy, the extra overheads impose latency penalties on standard GPU implementations.

Our proposed HeteroReason mitigates these hardware bottlenecks. Across all benchmarks and configurations, the heterogeneous system implements backtracking-enhanced approaches and accelerates the overall inference, reducing the speed degradation.

These results demonstrate the advantages of our algorithm-hardware co-design approach. By pairing the robust backtracking algorithm with a customized heterogeneous architecture, our system successfully delivers both higher accuracy and enhanced robustness while accelerating the execution speed, ultimately providing a faster and more reliable speculative reasoning system.

6.4 Ablation Study

To further investigate the performance contributions of our proposed optimization techniques, we conduct an ablation study as shown in Figure 10. The baseline denotes homogeneous GPU execution with backtracking enabled. (T2) adds the step-ahead speculation and refinement scheduling scheme, and (T1) further integrates the FPGA-based prefill-decode disaggregation techniques. We also conduct experiments to analyze the impact of backtracking.

Step-Ahead Speculation and Refinement (T2). The step-ahead speculation and refinement (T2) optimization implements

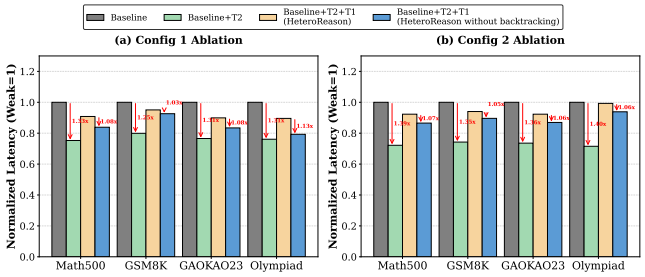

Figure 10: Ablation study with the step-ahead speculation and refinement scheme (T2), prefill-decode disaggregation paradigm (T1), and backtracking.

pipeline parallelism to overlap the speculating and refinement process with the verification process. The incorporation of T2 scheme delivers consistent latency reductions, achieving 1.25×–1.33× speedup in Config 1 and 1.35×–1.40× speedup in Config 2 compared with the baselines. These results explicitly demonstrate that transitioning from a sequential paradigm to a parallel pipeline with prefetching and caching optimizations effectively mitigates inter-device idle states and boosts the overall system speed. Note that this technique applies to diverse platforms, exhibiting high generality.

FPGA-based Accelerator with Prefill-Decode Disaggregation (T1). Building upon the T2 optimizations, the integration of T1 slightly degrades the speedup. But the heterogeneous system still shows higher energy efficiency as demonstrated in previous experiments, which is primarily attributed to lower power of FPGA.

Backtracking. Implementations on HeteroReason without backtracking reduce latency by 1.03×–1.13× in Config 1 and 1.05×–1.07× in Config 2. This confirms the speed-accuracy trade-off that backtracking improves accuracy but introduces additional latency overhead. HeteroReason with backtracking enabled already outperforms GPU baselines as demonstrated in Section 6.3, and shifts this trade-off frontier.

6.5 Communication Analysis

Since the heterogeneous system involves communication between the FPGA and GPU, we conduct experiments to evaluate the communication costs and sensitivity. Experimental results demonstrate that the communication costs are negligible. Across both configurations and all the datasets, the maximum communication latency accounts for only 0.0123% of the end-to-end latency. These results indicate that transferring a batch of tokens in each step instead of the KV cache effectively minimizes inter-device communication overhead.

Furthermore, we quantify the speculation stall ratio, which captures cases where the FPGA completes step-ahead speculation and becomes idle while waiting for verification results from the GPU. In the heterogeneous system, the speculation stall portion across all the steps ranges from 4.95% to 26.57%, with an average of 17.50%. This indicates a trade-off. Faster speculation can improve overall system performance, but also increases the probability that the draft model finishes before PRM verification completes, thereby increasing the relative stall portion even if the end-to-end latency decreases.

6.6 Energy Overheads of Wasted Tokens

The step-ahead speculation and refinement scheme launches the next speculative execution before the current verification result is received. This aggressive scheme may introduce wasted computation when the PRM rejects the current speculative step. The prefetched tokens are then discarded because they were generated from an invalid reasoning state. To quantify this cost, we measure the additional energy consumed by these discarded tokens on the homogeneous 3-GPU platforms under Config 1 to assess the energy overheads of step-ahead scheduling.

Table 4: Speedup and wasted-token energy overhead.

Dataset	MATH500	GSM8K	Gaokao2023En	OlympiadBench
Speedup	1.33×	1.25×	1.31×	1.31×
Overhead	0.719%	0.480%	0.631%	0.696%

Table 4 shows the wasted-token overhead is consistently small across all evaluated datasets. The step-ahead prefetching provides a 1.25×–1.33× latency speedup with only 0.480%–0.719% additional energy consumption. HeteroReason can incur lower energy overhead due to the lower power of FPGA. These results indicate that the energy cost of discarded computation is negligible compared with the latency improvement enabled by step-ahead execution.

The low overheads are expected for two reasons. First, wasted computation is incurred only on rejected verification paths, and the rejection probability in our execution traces is typically below 30%. Therefore, most prefetched tokens are eventually reused by the valid reasoning trajectory rather than discarded. Second, prefetching can be interrupted. Once the reject signal is received from the PRM, the system stops the invalid step-ahead execution, which further prevents unnecessary generation after the reasoning state has already been invalidated. As a result, this scheduling scheme converts idle hardware cycles into substantial latency reduction while adding only marginal energy overhead.

6.7 Discussion on Edge GPU/FPGA/ASIC for Speculation

The size of the draft model is smaller than the PRM and the target model, making it feasible to map onto resource-constrained accelerators. Edge GPUs, FPGAs, and ASICs are all potential platforms for executing the draft model and enabling step-ahead speculation and refinement. HeteroReason deploys the draft model on the FPGA.

Speculation on Edge GPU. We evaluate the W8A8 quantized draft model on NVIDIA Jetson Orin Nano. The software stack uses Jetson-optimized PyTorch, source-built vLLM, and the FlashInfer attention backend. Compared with FPGAs, edge GPUs contain multiple parallel processing cores, but the decoding speed is about 17 tokens/s across different prefill and decode lengths, since the decoding process is dominated by memory access rather than arithmetic throughput. As a result, its execution efficiency is limited by the mismatch between GPU-style SIMT parallelism and sequential token generation. The decoding speed is even lower than the 7B target model running on the RTX 3090 GPU. Therefore, replacing the FPGA draft engine with the Jetson edge GPU would degrade the performance, which is estimated to slow down the overall system by approximately 5×–6× compared with the FPGA-based HeteroReason. Consequently, moving the draft model to an edge GPU can reduce power consumption, but it does not fundamentally address the compute underutilization and memory-bound behavior of speculative drafting. By utilizing FPGA-based customized fully streaming design with dynamical KV cache management, lower latency can be achieved for these memory-intensive workloads.

Speculation on ASIC. Although ASICs allow for specialized hardware optimization, they suffer from prohibitive development costs and lengthy tape-out cycles. In contrast, the FPGA platform offers hardware specialization while preserving reconfigurability. Adapting to new techniques typically requires generating and deploying a new bitstream rather than fabricating a new chip.

Therefore, we use the FPGA prototype to explore an optimized algorithm-hardware co-design, providing practical design insights for future ASIC implementations.

7 Related Work

Algorithmic Development of Speculative Reasoning. Speculative decoding accelerates inference by drafting and verifying tokens strictly based on probability distributions. Recent studies on speculative reasoning have shifted towards semantic-level speculation [16, 55, 64]. SCoT [56] explores CoT-level drafting and selection and relies on task-specific LoRA alignment for the draft and selector models, unlike training-free HeteroReason. To mitigate rejections caused by expression divergence, where semantics are identical but tokens differ, [14, 58] probes the internal hidden states or utilizes the reflective capacity to concentrate on semantic verification, enhancing the acceptance rate of speculated outputs and accelerating the execution process. To further reduce the computational overhead introduced by the target model, a growing body of work [6, 15, 29, 32, 33, 40, 43, 48] focuses on dynamic routing and lightweight verification. Recently, the speculative reasoning paradigm has demonstrated efficacy in multi-modal tasks [26, 44] and agentic workflows [30, 66]. DREAM-R further improves multimodal speculative reasoning through RL-based refined drafting and specialized verification. Its verifier-side advances are complementary to HeteroReason's verifier-agnostic system mechanisms [27].

Customized Accelerators for Speculative Decoding. Studies on hardware accelerators for speculative decoding have been conducted. LP-Spec [24] accelerates mobile inference using GEMM-enhanced LPDDR5-PIM combined with hardware-aware token pruning. SpecPIM [35] utilizes architecture-dataflow co-exploration to address model heterogeneity in PIM-enabled speculative inference. SADDLE [57] enhances speculative decoding throughput on PIM-GPU heterogeneous systems by introducing an adaptive draft sequence length mechanism and an arithmetic intensity-aware dynamic operator scheduler. DFVG [47] proposes a heterogeneous FPGA-GPU architecture for speculative decoding, offloading the latency-sensitive draft generation to an FPGA while leveraging a GPU for high-throughput target verification.

8 Limitation

In HeteroReason, we propose an algorithm–hardware co-designed heterogeneous FPGA–GPU inference paradigm for speculative reasoning. In the current implementation, some experimental results are obtained through a combination of simulation and on-board evaluation. Future work will focus on developing a fully integrated, end-to-end demonstration system. However, several current limitations may be addressed through further system-level optimization and engineering effort. From a technical perspective, the proposed methods should also be evaluated across a broader range of model architectures, including linear-attention models [12, 54] and diffusion language models [45], to demonstrate their generalizability.

9 Conclusion

This work presents HeteroReason, a heterogeneous FPGA-GPU system specifically designed to accelerate speculative inference for LRMs, offloading memory-bound, sequential drafting to FPGA while reserving GPUs for compute-intensive verification and refinements. The proposed architecture introduces three key innovations: a backtracking-enhanced algorithm that successfully recovers from suboptimal reasoning states to improve overall LRM accuracy, a specialized workflow with shadow synchronization to achieve prefill-decode disaggregation, and a step-ahead speculation and refinement scheduling scheme that transforms rigid sequential dependencies into a parallel pipeline. Experimental results across diverse reasoning benchmarks demonstrate that HeteroReason delivers 1.01×–1.42× reduction in latency and 1.25×–1.57× gain in energy efficiency compared to homogeneous GPU baselines while improving reasoning accuracy. In conclusion, this algorithm-hardware co-design provides a highly scalable and robust solution for efficient LRM deployment.

Acknowledgment

The support of the UK EPSRC (Grants EP/V028251/1, EP/S030069/1, and EP/X036006/1), UKRI (Grant 256), KIAT and AMD is gratefully acknowledged.

A Artifact Appendix

A.1 Abstract

The artifact contains several components. The algorithmic component reproduces the Table 3 accuracy results for vanilla RSD, backtracking-enhanced RSD (BRSD), and BRSD with optimizations. The hardware component provides a trace-driven simulator and plotting workflow for reproducing the Figure 8 latency, goodput, and energy results from packaged CSV/JSONL inputs without rerunning model inference.

A.2 Artifact Check-list

- **Algorithm:** speculative reasoning with RSD, BRSD, and BRSD with optimized execution.
- **Program:** Python scripts, shell runners, step-profile CSVs, JSONL token logs, reference result summaries, and plotting utilities.
- **Models:** Qwen2.5 0.5B draft, Qwen2.5-Math-PRM-7B, Qwen2.5 7B target, and Qwen2.5 1.5B target. Model checkpoints are not redistributed.
- **Datasets:** Math500, GSM8K, Gaokao2023EN, and OlympiadBench.
- **Compute platform:** Algorithmic runs use 3× NVIDIA RTX 3090 GPUs in the reference setup. The packaged hardware simulator and Figure 8 plotting workflow run from CSV/JSONL traces and do not require GPUs.
- **Expected time:** The algorithmic smoke test takes a few minutes after model initialization; the Config1/Math500 key result takes about 1–2 hours; the full algorithmic workflow can take about 1–2 days depending on GPU availability. The packaged hardware plotting workflow takes seconds, and regenerating simulator CSVs from the packaged traces takes minutes.

A.3 Description

The artifact codebase: https://zenodo.org/records/22015981 or https://github.com/zehuanzhang/HeteroReason. The artifact is organized into several directories. `AE_alg` contains the algorithmic reproduction workflow under `Algorithm`. `AE_hardware` contains the trace-driven latency/energy simulator, the packaged step-profile CSVs and logs, precomputed simulator outputs. `AE_hardware_V80` is used for V80 simulations.

The three Table 3 algorithmic methods are:

- `rsd`: the vanilla speculative reasoning baseline.
- `brsd`: RSD with backtracking.
- `brsd_optimized`: BRSD with the optimizations.

These correspond to the implementation names `beam1`, `bbeam1`, and `bbeam1_prefetch_cache`, respectively.

A.4 Installation

After downloading the artifact, install the Python dependencies and set local model paths for algorithmic reproduction:

```
pip install -r AE_alg/requirements.txt
export DRAFT_MODEL=/path/to/Qwen2.5-0.5B-Instruct
export PRM_MODEL=/path/to/Qwen2.5-Math-PRM-7B
export CONFIG1_TARGET_MODEL=/path/to/Qwen2.5-7B-Instruct
export CONFIG2_TARGET_MODEL=/path/to/Qwen2.5-1.5B-Instruct
export CUDA_VISIBLE_DEVICES=0,1,2
```

The hardware simulator and Figure 8 plotting workflow use packaged CSV/JSONL inputs and do not require model checkpoints.

A.5 Algorithmic Experiment Workflow

The smoke test validates the environment by running the three Table 3 methods on eight fixed Config1/Math500 examples:

```
cd AE_alg/Algorithm
bash run_smoke.sh
```

The key-result workflow focuses on Config1/Math500. The default key-result command runs `brsd_optimized`; reviewers may also run the three methods separately:

```
cd AE_alg/Algorithm
bash run_key_results.sh

KEY_METHODS="rsd" bash run_key_results.sh
KEY_METHODS="brsd" bash run_key_results.sh
KEY_METHODS="brsd_optimized" bash run_key_results.sh
```

The full algorithmic workflow reproduces Table 3 across both configurations, all four datasets, and all three methods:

```
cd AE_alg/Algorithm
```

```
bash run_full_results.sh
```

A.6 Hardware Simulator and Figure Workflow

The hardware component can reproduce Figure 8 without rerunning model inference:

```
cd AE_camera
python figure8/plot_figure8.py --normalized
```

A.7 Evaluation and Expected Results

The algorithmic smoke test reports per-method accuracy and latency on the fixed eight-example subset. The key-result workflow reproduces the Config1/Math500 Table 3 entries, with `brsd_optimized` as the recommended artifact result. The full workflow reproduces all Table 3 algorithmic accuracy values across Config1 and Config2.

The hardware workflow regenerates Figure 8 and the corresponding CSV/JSON metric summaries. The expected metrics include average latency, goodput, and average energy per problem.

A.8 Notes

Algorithmic latency is platform-dependent and can vary with GPU types. The hardware simulator and plotting workflow are deterministic for the supplied CSV/JSONL inputs. Model checkpoints are referenced by local paths or Hugging Face identifiers and are not redistributed with the artifact.

References

[1] F. Warren Burton. 1985. Speculative Computation, Parallelism, and Functional Programming. *IEEE Trans. Comput.* 34, 12 (1985), 1190–1193.

[2] Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. 2024. Medusa: Simple LLM Inference Acceleration Framework with Multiple Decoding Heads. *arXiv preprint arXiv:2401.10774* (2024).

[3] Hao Mark Chen, Guanxi Lu, Yasuyuki Okoshi, Zhiwen Mo, Masato Motomura, and Hongxiang Fan. 2025. Rethinking Optimal Verification Granularity for Compute-Efficient Test-Time Scaling. *arXiv preprint arXiv:2505.11730* (2025).

[4] Hao Mark Chen, Wayne Luk, Ka Fai Cedric Yiu, Rui Li, Konstantin Mishchenko, Stylianos I Venieris, and Hongxiang Fan. 2024. Hardware-Aware Parallel Prompt Decoding for Memory-Efficient Acceleration of LLM Inference. *arXiv preprint arXiv:2405.18628* (2024).

[5] Hao Mark Chen, Zhiwen Mo, Guanxi Lu, Shuang Liang, Lingxiao Ma, Wayne Luk, and Hongxiang Fan. 2026. Fasttts: Accelerating test-time scaling for edge LLM reasoning. In *Proceedings of the 31st ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*. 732–748.

[6] Zigeng Chen, Xinyin Ma, Gongfan Fang, Ruonan Yu, and Xinchao Wang. 2025. VeriThinker: Learning to Verify Makes Reasoning Model Efficient. *arXiv preprint arXiv:2505.17941* (2025).

[7] Zhuoming Chen, Avner May, Ruslan Svirschevski, Yu-Hsun Huang, Max Ryabinin, Zhihao Jia, and Beidi Chen. 2024. Sequoia: Scalable and Robust Speculative Decoding. *Advances in Neural Information Processing Systems* 37 (2024), 129531–129563.

[8] Aakanksha Chowdhery et al. 2023. PaLM: Scaling Language Modeling with Pathways. *Journal of Machine Learning Research* 24, 240 (2023), 1–113.

[9] Yuanlin Chu, Bo Wang, Xiang Liu, Hong Chen, Aiwei Liu, and Xuming Hu. 2025. SSR: Speculative Parallel Scaling Reasoning in Test-Time. *arXiv preprint arXiv:2505.15340* (2025).

[10] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training Verifiers to Solve Math Word Problems. *arXiv preprint arXiv:2110.14168* (2021).

[11] Codeforces. 2025. Codeforces: Competitive Programming Platform. https://codeforces.com. Accessed: 2025-03-18.

[12] Tri Dao and Albert Gu. 2024. Transformers are SSMs: Generalized Models and Efficient Algorithms Through Structured State Space Duality. *arXiv preprint arXiv:2405.21060* (2024).

[13] Daya Guo and others. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *Nature* (2025).

[14] Ximing Dong, Shaowei Wang, Dayi Lin, Boyuan Chen, and Ahmed E Hassan. 2026. Beyond Tokens: Semantic-Aware Speculative Decoding for Efficient Inference by Probing Internal States. *arXiv preprint arXiv:2602.03708* (2026).

[15] Tianyu Fu, Yi Ge, Yichen You, Enshu Liu, Zhihang Yuan, Guohao Dai, Shengen Yan, Huazhong Yang, and Yu Wang. 2025. R2R: Efficiently Navigating Divergent Reasoning Paths with Small-Large Model Token Routing. *arXiv preprint arXiv:2505.21600* (2025).

[16] Yichao Fu, Rui Ge, Zelei Shao, Zhijie Deng, and Hao Zhang. 2025. Scaling Speculative Decoding with Lookahead Reasoning. *arXiv preprint arXiv:2506.19830* (2025).

[17] Wenxi Gao, Guanxi Lu, Didi Zhu, Hao Mark Chen, Quan Deng, Zhican Wang, Jiankang Deng, and Hongxiang Fan. 2026. Model Guides You How to Draw: Adaptive Visual Gating for Unified Multimodal Reasoning. *arXiv preprint arXiv:2607.10004* (2026).

[18] Gemma Team. 2024. Gemma: Open Models Based on Gemini Research and Technology. *arXiv preprint arXiv:2403.08295* (2024).

[19] Aaron Grattafiori et al. 2024. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783* (2024).

[20] Licheng Guo, Yuze Chi, Jason Lau, Linghao Song, Xingyu Tian, Moazin Khatti, Weikang Qiao, Jie Wang, Ecenur Ustun, Zhenman Fang, Zhiru Zhang, and Jason Cong. 2023. TAPA: A Scalable Task-Parallel Dataflow Programming Framework for Modern FPGAs with Co-Optimization of HLS and Physical Design. *ACM Transactions on Reconfigurable Technology and Systems* 16, 4 (2023), 1–31.

[21] Licheng Guo, Pongstorn Maidee, Yun Zhou, Chris Lavin, Eddie Hung, Wuxi Li, Jason Lau, Weikang Qiao, Yuze Chi, Linghao Song, Yuanlong Xiao, Alireza Kaviani, Zhiru Zhang, and Jason Cong. 2023. RapidStream 2.0: Automated Parallel Implementation of Latency-Insensitive FPGA Designs Through Partial Reconfiguration. *ACM Transactions on Reconfigurable Technology and Systems* 16, 4 (2023), 1–30.

[22] Licheng Guo, Pongstorn Maidee, Yun Zhou, Chris Lavin, Jie Wang, Yuze Chi, Weikang Qiao, Alireza Kaviani, Zhiru Zhang, and Jason Cong. 2022. RapidStream: Parallel Physical Implementation of FPGA HLS Designs. In *Proceedings of the 2022 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. 1–12.

[23] Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. 2024. OlympiadBench: A Challenging Benchmark for Promoting AGI with Olympiad-Level Bilingual Multimodal Scientific Problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 3828–3850.

[24] Siyuan He, Zhantong Zhu, Yandong He, and Tianyu Jia. 2025. LP-Spec: Leveraging LPDDR PIM for Efficient LLM Mobile Speculative Inference with Architecture-Dataflow Co-Optimization. *arXiv preprint arXiv:2508.07227* (2025).

[25] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring Mathematical Problem Solving with the MATH Dataset. *arXiv preprint arXiv:2103.03874* (2021).

[26] Pengfei Hu, Meng Cao, Yingyao Wang, Yi Wang, Jiahua Dong, Jun Song, Yu Cheng, Bo Zheng, and Xiaodan Liang. 2025. Thinking with Drafts: Speculative Temporal Reasoning for Efficient Long Video Understanding. *arXiv preprint arXiv:2512.00805* (2025).

[27] Yunhai Hu, Zining Liu, Xiangyang Yin, Tianhua Xia, Bo Bao, Eric Sather, Vithursan Thangarasa, and Sai Qian Zhang. 2026. DREAM-R: Multimodal Speculative Reasoning with RL-Based Refined Drafting, Precise Verification, and Fully Parallel Execution. *arXiv preprint arXiv:2605.28678* (2026).

[28] Zhengyu Hu, Jianxun Lian, Zheyuan Xiao, Seraphina Zhang, Tianfu Wang, Nicholas Jing Yuan, Xing Xie, and Hui Xiong. 2025. Unveiling the Learning Mind of Language Models: A Cognitive Framework and Empirical Study. *arXiv preprint arXiv:2506.13464* (2025).

[29] Chengsong Huang, Tong Zheng, Langlin Huang, Jinyuan Li, Haolin Liu, and Jiaxin Huang. 2026. RelayLLM: Efficient Reasoning via Collaborative Decoding. *arXiv preprint arXiv:2601.05167* (2026).

[30] Haoyu Huang, Jinfa Huang, Zhongwei Wan, Xiawu Zheng, Rongrong Ji, and Jiebo Luo. 2026. SpecEyes: Accelerating Agentic Multimodal LLMs via Speculative Perception and Planning. *arXiv preprint arXiv:2603.23483* (2026).

[31] Zhongzhen Huang, Gui Geng, Shengyi Hua, Zhen Huang, Haoyang Zou, Shaoting Zhang, Pengfei Liu, and Xiaofan Zhang. 2025. O1 Replication Journey–Part 3: Inference-Time Scaling for Medical Reasoning. *arXiv preprint arXiv:2501.06458* (2025).

[32] Vansh Kapoor, Aman Gupta, Hao Chen, Anurag Beniwal, Jing Huang, and Aviral Kumar. 2026. TRIM: Hybrid Inference via Targeted Stepwise Routing in Multi-Step Reasoning Tasks. *arXiv preprint arXiv:2601.10245* (2026).

[33] Sangmook Lee, Dohyung Kim, Hyukhun Koh, Nakyeong Yang, and Kyomin Jung. 2026. Confidence-Guided Stepwise Model Routing for Cost-Efficient Reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 40. 31483–31491.

[34] Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast Inference from Transformers via Speculative Decoding. In *International Conference on Machine Learning*. PMLR, 19274–19286.

[35] Cong Li, Zhe Zhou, Size Zheng, Jiaxi Zhang, Yun Liang, and Guangyu Sun. 2024. SpecPIM: Accelerating Speculative Inference on PIM-Enabled System via Architecture-Dataflow Co-Exploration. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*. 950–965.

[36] Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2025. EAGLE-3: Scaling Up Inference Acceleration of Large Language Models via Training-Time Test. *arXiv preprint arXiv:2503.01840* (2025).

[37] Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, Junhao Zheng, Pei-Jie Wang, Xiuyi Chen, Yingying Zhang, Fei Yin, Jiahua Dong, Zhijiang Guo, Le Song, and Cheng-Lin Liu. 2025. From System 1 to System 2: A Survey of Reasoning Large Language Models. *arXiv preprint arXiv:2502.17419* (2025).

[38] Baohao Liao, Yuhui Xu, Hanze Dong, Junnan Li, Christof Monz, Silvio Savarese, Doyen Sahoo, and Caiming Xiong. 2025. Reward-Guided Speculative Decoding for Efficient LLM Reasoning. *arXiv preprint arXiv:2501.19324* (2025).

[39] Minpeng Liao, Wei Luo, Chengxi Li, Jing Wu, and Kai Fan. 2024. MARIO: MAth Reasoning with Code Interpreter Output–A Reproducible Pipeline. *arXiv preprint arXiv:2401.08190* (2024).

[40] Weizhe Lin, Xing Li, Zhiyuan Yang, Xiaojin Fu, Hui-Ling Zhen, Yaoyuan Wang, Xianzhi Yu, Wulong Liu, Xiaosong Li, and Mingxuan Yuan. 2025. TrimR: Verifier-Based Training-Free Thinking Compression for Efficient Test-Time Scaling. *arXiv preprint arXiv:2505.17155* (2025).

[41] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2024. Improved Baselines with Visual Instruction Tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 26296–26306.

[42] Lei Liu, Xiaoyan Yang, Junchi Lei, Yue Shen, Jian Wang, Peng Wei, Zhixuan Chu, Zhan Qin, and Kui Ren. 2024. A Survey on Medical Large Language Models: Technology, Application, Trustworthiness, and Future Directions. *arXiv preprint arXiv:2406.03712* (2024).

[43] Siran Liu and Cyril Y He. 2026. ConfSpec: Efficient Step-Level Speculative Reasoning via Confidence-Gated Verification. *arXiv preprint arXiv:2602.18447* (2026).

[44] Yuhan Liu, Lianhui Qin, and Shengjie Wang. 2025. Small Drafts, Big Verdict: Information-Intensive Visual Reasoning via Speculation. *arXiv preprint arXiv:2510.20812* (2025).

[45] Guanxi Lu, Hao Chen, Yuto Karashima, Zhican Wang, Daichi Fujiki, and Hongxiang Fan. 2026. Adablock-dllm: Semantic-aware diffusion llm inference via adaptive block size. In *International Conference on Learning Representations*, Vol. 2026. 140018–140033.

[46] Haoyu Lu, Wen Liu, Bo Zhang, Bingxuan Wang, Kai Dong, Bo Liu, Jingxiang Sun, Tongzheng Ren, Zhuoshu Li, Hao Yang, Yaofeng Sun, Chengqi Deng, Hanwei Xu, Zhenda Xie, and Chong Ruan. 2024. DeepSeek-VL: Towards Real-World Vision-Language Understanding. *arXiv preprint arXiv:2403.05525* (2024).

[47] Shaoqiang Lu, Yangbo Wei, Junhong Qian, Dongge Qin, Shiji Gao, Yizhi Ding, Qifan Wang, Chen Wu, Xiao Shi, and Lei He. 2026. DFVG: A Heterogeneous Architecture for Speculative Decoding with Draft-on-FPGA and Verify-on-GPU. In *Proceedings of the 31st ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*. 602–617.

[48] Monishwaran Maheswaran, Rishabh Tiwari, Yuezhou Hu, Kerem Dilmen, Coleman Hooper, Haocheng Xi, Nicholas Lee, Mehrdad Farajtabar, Michael W. Mahoney, Kurt Keutzer, and Amir Gholami. 2025. Arbitrage: Efficient Reasoning via Advantage-Aware Speculation. *arXiv preprint arXiv:2512.05033* (2025).

[49] Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, Chunan Shi, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. 2024. SpecInfer: Accelerating Large Language Model Serving with Tree-Based Speculative Inference and Verification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*. 932–949.

[50] OpenAI. 2025. OpenAI o3-mini System Card.

[51] OpenAI Team. 2024. OpenAI o1 System Card. *arXiv preprint arXiv:2412.16720* (2024).

[52] Rui Pan, Yinwei Dai, Zhihao Zhang, Gabriele Oliaro, Zhihao Jia, and Ravi Netravali. 2025. SpecReason: Fast and Accurate Inference-Time Compute via Speculative Reasoning. *arXiv preprint arXiv:2504.07891* (2025).

[53] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. 2024. SAM 2: Segment Anything in Images and Videos. *arXiv preprint arXiv:2408.00714* (2024).

[54] Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li. 2021. Efficient Attention: Attention with Linear Complexities. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 3531–3539.

[55] Junhan Shi, Yijia Zhu, Zhenning Shi, Dan Zhao, Qing Li, and Yong Jiang. 2025. SpecCoT: Accelerating Chain-of-Thought Reasoning through Speculative Exploration. *Findings of the Association for Computational Linguistics: EMNLP 2025* (2025), 24405–24415.

[56] Jikai Wang, Juntao Li, Jianye Hou, Bowen Yan, Lijun Wu, and Min Zhang. 2025. Efficient Reasoning for LLMs Through Speculative Chain-of-Thought. *arXiv preprint arXiv:2504.19095* (2025).

[57] Runze Wang, Qinggang Wang, Haifeng Liu, Long Zheng, Xiaofei Liao, Hai Jin, and Jingling Xue. 2026. Adaptive Draft Sequence Length: Enhancing Speculative Decoding Throughput on PIM-Enabled Systems. In *2026 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 1–15.

[58] Yixuan Wang, Yijun Liu, Shiyu Ji, Yuzhuang Xu, Yang Xu, Qingfu Zhu, and Wanxiang Che. 2025. Think Before You Accept: Semantic Reflective Verification for Faster Speculative Decoding. *arXiv preprint arXiv:2505.18629* (2025).

[59] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *Advances in Neural Information Processing Systems* 35 (2022), 24824–24837.

[60] Fengli Xu, Qianyue Hao, Zefang Zong, Jingwei Wang, Yunke Zhang, Jingyi Wang, Xiaochong Lan, Jiahui Gong, Tianjian Ouyang, Fanjin Meng, Chenyang Shao, Yuwei Yan, Qinglong Yang, Yiwen Song, Sijian Ren, Xinyuan Hu, Yu Li, Jie Feng, Chen Gao, and Yong Li. 2025. Towards Large Reasoning Models: A Survey of Reinforced Reasoning with Large Language Models. *arXiv preprint arXiv:2501.09686* (2025).

[61] Haotian Xu, Xing Wu, Weinong Wang, Zhongzhi Li, Da Zheng, Boyuan Chen, Yi Hu, Shijia Kang, Jiaming Ji, Yingying Zhang, Zhijiang Guo, Yaodong Yang, Muhan Zhang, and Debing Zhang. 2025. RedStar: Does Scaling Long-CoT Data Unlock Better Slow-Reasoning Systems? *arXiv preprint arXiv:2501.11284* (2025).

[62] An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. 2024. Qwen2.5-Math Technical Report: Toward Mathematical Expert Model via Self-Improvement. *arXiv preprint arXiv:2409.12122* (2024).

[63] Lin Yang, Shawn Xu, Andrew Sellergren, Timo Kohlberger, Yuchen Zhou, Ira Ktena, Atilla P. Kiraly, Faruk Ahmed, Farhad Hormozdiari, Tiam Jaroensri, Eric Wang, Ellery Wulczyn, Fayaz Jamil, Theo Guidroz, Chuck Lau, Siyuan Qiao, Yun Liu, Akshay Goel, Kendall Park, Arnav Agharwal, Nick George, Yang Wang, Ryutaro Tanno, David G. T. Barrett, Wei-Hung Weng, S. Sara Mahdavi, Khaled Saab, Tao Tu, Sreenivasa Raju Kalidindi, Mozziyar Etemadi, Jorge Cuadros, Gregory Sorensen, Yossi Matias, Katherine Chou, Greg Corrado, Joelle K. Barral, Shravya Shetty, David J. Fleet, S. M. Ali Eslami, Daniel Tse, Shruthi Prabhakara, Cory Y. McLean, Dave Steiner, Rory Pilgrim, Christopher Kelly, Shekoofeh Azizi, and Daniel Golden. 2024. Advancing Multimodal Medical Capabilities of Gemini. *arXiv preprint arXiv:2405.03162* (2024).

[64] Wang Yang, Xiang Yue, Vipin Chaudhary, and Xiaotian Han. 2025. Speculative Thinking: Enhancing Small-Model Reasoning with Large Model Guidance at Inference Time. *arXiv preprint arXiv:2504.12329* (2025).

[65] Jiahao Zhang, Zifan He, Nicholas Fraser, Michaela Blott, Yizhou Sun, and Jason Cong. 2026. FlexLLM: Composable HLS Library for Flexible Hybrid LLM Accelerator Design. *arXiv preprint arXiv:2601.15710* (2026).

[66] Shuzhang Zhong, Baotong Lu, Qi Chen, Chuanjie Liu, Fan Yang, and Meng Li. 2026. DualSpec: Accelerating Deep Research Agents via Dual-Process Action Speculation. *arXiv preprint arXiv:2603.07416* (2026).