RUBATO: DECOUPLED CLOCKS ALLOW FULL-DUPLEX MODELS TO THINK AS FAST AS THEY CAN

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent Speech Full-Duplex (FDX) models are capable of engaging in naturalistic conversation by listening and speaking at the same time. Building on this foundation, recent architectures introduce internal thinking streams to enhance their intelligence. To guarantee these streams process successfully in real time, existing methods enforce a fixed decoding rate, which assumes a hardware profile capable of a specific latency. We argue this static approach is severely suboptimal. Because internal thinking is intrinsically asynchronous, an ideal FDX model should sustain real-time conversational flow while allowing each stream to decode at its own hardware-adaptive rate. Thinking streams must process as fast as the hardware allows to deliver accurate answers faster. To solve this, we propose RUBATO, a novel multistream positional encoding that assigns independent clocks to entirely decouple the streams. This structural isolation empowers the model to zero-shot interpolate across any inference decoding speed. It drastically simplifies dataset construction by requiring training on only two extreme decoding speeds while smoothly adapting to any intermediate rate during deployment. RUBATO demonstrates consistent robustness across five spoken reasoning tasks. On Spoken GSM8K, a baseline model collapses to **1.7%** accuracy at interpolated speeds. Conversely, RUBATO-Full sustains **48.7% to 51.1%** accuracy across all rates, leveraging faster hardware to achieve a **6.5**$\times$ shorter time-to-answer with a constant time-to-first-response. This advantage holds on BIG-Bench Audio and amplifies significantly during complex full-duplex interruptions. Specifically, when a user interjects while the model is actively speaking and reasoning, RUBATO-Full achieves **40.0% to 45.1%** accuracy, whereas the baseline fails completely.

1 INTRODUCTION

Voice agents are increasingly popular as speech provides a highly intuitive medium for human-computer interaction. To unlock fluid communication, recent research focuses on Speech Full-Duplex (FDX) models (Chen & Yu, 2025; Lu et al., 2026), which concurrently process incoming and outgoing audio streams in real-time to manage dynamic conversational behaviors like backchanneling and interruptions. To capture the full richness of human interaction, recent works are expanding these architectures by integrating additional streams across two primary directions. First, because natural communication relies heavily on non-verbal signals, researchers have added multimodal streams to process incoming visual cues (Jiang et al., 2026; Lim et al., 2026; Huang et al., 2026) and generate outgoing physical actions (Wang et al., 2026; Zhang et al., 2026). Second, human conversation requires deep cognitive processing rather than just superficial spoken fluency. Because basic Speech FDX models generally lag behind their text-only counterparts in this core intelligence (Xiang et al., 2025), researchers have also introduced internal *thinking* streams. This bridges the cognitive gap, empowering the model to perform Chain-of-Thought reasoning (Shih et al., 2026b; Chiang et al., 2026b;a; Wu et al., 2026; Shih et al., 2026a) and leverage external capabilities (Chien et al., 2026; Arora et al., 2026) during live conversations.

Modeling multiple streams within a Speech FDX architecture introduces a profound temporal challenge: the model must simultaneously orchestrate continuous real-time streams (e.g., audio and visual signals) alongside an internal thinking stream. The core complexity stems from their fundamentally incompatible pacing. Real-time streams operate at strict periodic intervals and require immediate generation to maintain conversational latency. In contrast, the thinking stream is triggered

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

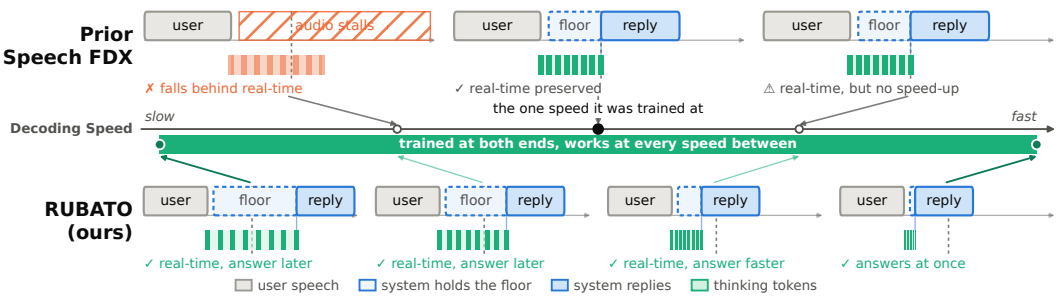

Figure 1: **Comparison between RUBATO and prior methods under varying decoding speeds.** Given an identical question and reasoning trace, varying inference speeds stretch or compress thinking tokens along the wall-clock timeline. The dotted line denotes a reference timestamp. Prior models rely on a static decoding rate where slow execution stalls real-time audio streams and fast execution wastes available compute. In contrast, RUBATO dynamically adapts by holding the conversational floor during slower decoding and accelerating answer delivery when hardware permits.

intermittently and is entirely decoupled from physical time, rendering it inherently asynchronous. To ensure the model processes all streams without violating real-time constraints, prior works (Shih et al., 2026b; Zhang et al., 2026) typically rely on a fixed synchronization strategy. By targeting a specific hardware profile, they enforce a rigid decoding pattern that allocates a fixed frame rate to the thinking stream relative to real-time inputs, guaranteeing the model won't fail during deployment.

We argue this static design is fundamentally limiting. An ideal Speech FDX model must allow each stream to maintain its own independent decoding pace and dynamically adapt to available compute while preserving real-time responsiveness (Fig 1). We motivate this decoupled approach for two key reasons. 1) **Inelastic compute scaling:** Static decode rates for all streams prevent the model from fully leveraging fast hardware to think faster. 2) **Hardware variability:** Decoding speeds fluctuate across different GPUs, software frameworks, and dynamic server loads. This widespread variance makes it impractical to retrain the model to match every possible target decoding speed. Ultimately, we seek a model architecture that allows real-time interaction to remain stable while thinking streams can process as rapidly as available compute permits.

A concurrent line of research attempts to decouple heavy reasoning from real-time interaction by splitting the architecture into a fast interaction model and a slower background reasoning model (Thinking Machines Lab, 2026; Huang et al., 2026). However, this dual-model paradigm introduces substantial engineering overhead for deployment and severely limits the preservation of paralinguistic audio cues during text-based communication between the two systems. More importantly, this split does not bypass the core temporal challenge. The frontend interaction model must still orchestrate real-time audio alongside the discrete tokens required to communicate with the background model. Consequently, these systems face similar bottlenecks when handling asynchronous streams while adapting to different generation speeds. Therefore, any solution that successfully eliminates these rigid temporal bottlenecks would not only enable single-model reasoning but also remain highly synergistic with dual-model architectures.

To this end, we introduce RUBATO, a multistream positional encoding that allows multiple streams to maintain independent and adaptive decoding rates (Fig. 1). We apply this to an end-to-end Speech FDX model for spoken reasoning, enabling the system to simultaneously listen, speak, and reason in real time. Prior works (Shih et al., 2026b; Chiang et al., 2026b; Zhang et al., 2026) rely on a single shared position counter, a mechanism that functions properly under their fixed synchronization strategies. Naively porting this shared counter to a system with adaptive stream pacing causes the positional distance between consecutive real-time audio tokens to be arbitrarily distorted by the dynamic volume of intermediate thinking tokens. RUBATO resolves this temporal distortion by assigning independent clocks to the real-time and thinking streams so that each clock advances exclusively on its designated tokens. By doing so, RUBATO enforces a steady temporal progression for real-time audio while allowing the thinking stream to decode as rapidly as hardware permits [1].

[1] We borrow the term RUBATO from music (Hudson, 1994), where the accompaniment holds a steady pulse while the melody moves freely against it, redistributing its notes without disturbing the underlying beat. This mirrors our pairing of a steady real-time audio stream with a dynamically paced thinking stream.

Beyond resolving this architectural bottleneck, our decoupled design significantly simplifies dataset construction and training. Simulating diverse reasoning rates in the training data severely complicates temporal alignment, as the model must gather sufficient audio context before initiating thought and conclude its reasoning before the system speaks the resulting answer. Calculating these precise alignments across varying decoding speeds imposes substantial overhead during corpus generation. RUBATO greatly eases this data preparation process by removing the strict dependence on deployment budgets. Instead of generating data for every possible speed, we train the model exclusively on two easily generated temporal boundaries. At the baseline rate, the thinking duration aligns deterministically with the real-time audio frames. At the instantaneous rate, the entire reasoning trace resolves within a single audio frame, simulating infinite compute. Everything between these extremes acts as a zero-shot interpolation. Because RUBATO structurally teaches the model that these timelines advance independently, the network seamlessly generalizes to any dynamic reasoning speed during deployment.

We evaluate RUBATO on five single-turn spoken reasoning tasks from Spoken GSM8K (Cobbe et al., 2021) and BIG-Bench Audio (Srivastava et al., 2023; Artificial Analysis, 2024a), alongside complex overlapping scenarios where the model must simultaneously think, speak, and listen. Across all settings, RUBATO sustains robust reasoning accuracy and a constant time-to-first-response. Crucially, it allows the system to utilize faster hardware to reduce the final time-to-answer. In summary, our key contributions are:

- **The Asynchronous-Realtime Coupling Challenge:** We identify a fundamental limitation in prior works where asynchronous thinking and real-time audio are tied together on the same timeline. This rigid coupling fails to accommodate dynamic generation speeds.
- **Novel Decoupled Clock Mechanism:** RUBATO decouples real-time and thinking streams onto independent timelines, running efficiently on stock fused attention kernels without custom modifications.
- **Efficient Training via Zero-Shot Interpolation:** By structurally separating the stream timelines, RUBATO drastically simplifies dataset construction. This design allows the model to train exclusively on two extreme temporal boundaries and generalize zero-shot to any dynamic inference speed during deployment.
- **Robust Empirical Performance:** We demonstrate that our framework sustains high reasoning accuracy across dynamic inference speeds on complex spoken reasoning tasks. Crucially, it directly translates faster hardware into significantly reduced user-facing latency.

2 BACKGROUND

2.1 ROPE: ROTARY POSITION EMBEDDINGS

An attention head of dimension D maps the token at sequence position m to a query $\boldsymbol{q}_m \in \mathbb{R}^D$ and a key $\boldsymbol{k}_m \in \mathbb{R}^D$. Rotary position embeddings (RoPE) (Su et al., 2023) inject position by rotating both vectors in $D/2$ two-dimensional planes before the dot product, plane i turning at angular frequency

$$\theta_i \;=\; \Theta^{-2(i-1)/D}, \qquad i = 1, \ldots, D/2, \tag{1}$$

with frequency base Θ. Position m acts through the block-diagonal matrix $\boldsymbol{R}(m)$ whose i-th 2×2 block rotates by the angle $\theta_i m$, and because $\boldsymbol{R}(m)^\top \boldsymbol{R}(n) = \boldsymbol{R}(n-m)$, the attention logit between a query at m and a key at n is

$$\big\langle \boldsymbol{R}(m)\, \boldsymbol{q}_m,\; \boldsymbol{R}(n)\, \boldsymbol{k}_n \big\rangle \;=\; \big\langle \boldsymbol{q}_m,\; \boldsymbol{R}(n-m)\, \boldsymbol{k}_n \big\rangle, \tag{2}$$

which depends on the two positions only through their offset $n - m$. In implementation, the rotation is applied to the query and the key independently. Attention therefore stays a plain dot product over the rotated tensors and can be computed by a fused kernel such as FlashAttention (Dao et al., 2022).

2.2 MULTISTREAM FULL-DUPLEX MODELS

To enable full-duplex interaction, we adopt the four-stream interleaving strategy proposed in SCoT (Arora et al., 2025). This architecture consists of User Audio (UA), System Audio (SA), User Text (UT), and System Text (ST). By providing time-aligned ASR and TTS text for the user and

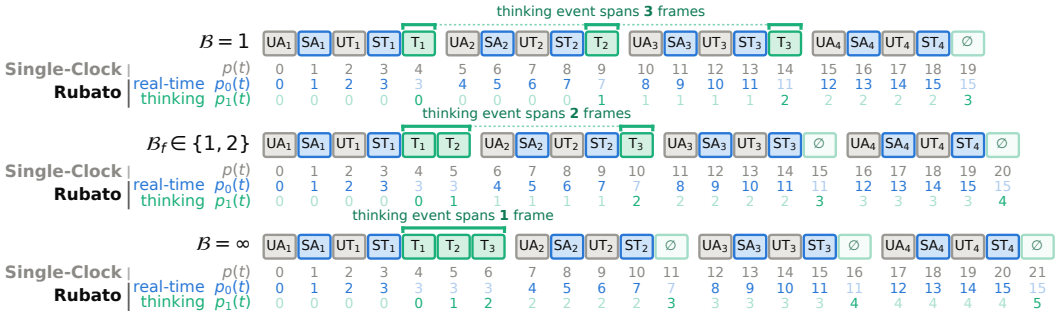

Figure 2: **Positional index arrangement for RUBATO vs. Single-Clock.** An example 3-token thinking event (T_1 to T_3) starting at frame 1 spans a variable number of frames based on $\mathcal{B}_f$. While Single-Clock applies a continuous index that arbitrarily distorts the distance between real-time tokens, RUBATO assigns independent clocks that advance exclusively on designated tokens (full color) and pause elsewhere (faint). This guarantees consecutive real-time tokens remain strictly equidistant, fully decoupling the audio and reasoning timelines.

system, respectively, they improve semantic coherence. During preprocessing, audio tokenization compresses the waveforms into discrete tokens, where the temporal span of a single token defines one *frame* (e.g., 80 ms).

Prior architectures model these streams via concatenation or interleaving. Concatenation (Défossez et al., 2024; Roy et al., 2026) forces a synchronized processing rate, while interleaving (Wang et al., 2026; Zhang et al., 2025) flattens streams into a single 1D sequence for greater flexibility. Both paradigms face strict real-time constraints, meaning the decoding computation must finish faster than the physical duration of the processed audio frames. Because modern GPUs decode rapidly, the residual compute time within each interleaved chunk can be leveraged to decode internal thinking streams like reasoning or RAG.

We adopt the interleaved paradigm due to its flexibility. Because hardware decoding speeds dictate how many reasoning tokens can be generated between consecutive real-time audio frames, we formally define this dynamic capacity as the thinking budget $\mathcal{B}$. Let $\mathrm{UA}_f, \mathrm{SA}_f, \mathrm{UT}_f, \mathrm{ST}_f$, and $\mathrm{TT}_{f,k}$ represent the tokens for their respective streams at frame f. We capture frame-level hardware fluctuations by assigning a dynamic budget $\mathcal{B}_f$ to frame f, defining the interleaved chunk C_f as:

$$C_f = \left[\mathrm{UA}_f, \mathrm{SA}_f, \mathrm{UT}_f, \mathrm{ST}_f, \mathrm{TT}_{f,1}, \ldots, \mathrm{TT}_{f,\mathcal{B}_f}\right] \tag{3}$$

Real-time tokens are positioned first to ensure immediate processing, followed by the variable sequence of thinking tokens. The model consumes these chunks sequentially as a single flat sequence $\boldsymbol{x} = [C_1, \ldots, C_F]$, and we use t to denote the position of a token in the flattened sequence. Since $\mathcal{B}$ is a per-frame limit, any reasoning process exceeding the current budget simply overflows into the next frame. Unlike prior works that fix $\mathcal{B}$ during training and inference, our framework allows $\mathcal{B}$ to depend on inference time capacities. In practice, $\mathcal{B}$ depends on model size, codec frame rate, serving techniques, and hardware. To simplify, we evaluate varying $\mathcal{B}$ directly and defer detailed estimates to Appendix E.

The Deficit of Single-Clock. Existing methods apply RoPE to this flattened sequence using a single monotonically increasing index. While effective for a fixed $\mathcal{B}$, this design creates a positional distribution shift when inference budgets differ from those observed during training. Dynamically expanding or shrinking the intermediate thinking tokens arbitrarily distorts the positional distance between consecutive real-time tokens like UA_f and UA_{f+1}. This single-clock constraint thereby prevents the model from robustly learning the true temporal alignment of the streams.

3 PROPOSED METHOD

3.1 RUBATO

To overcome the limitations of a single clock, we introduce RUBATO, which assigns multiple independent positional indices (clocks) to the FDX model's streams. While extendable to C clocks,

we use two for our setup: a real-time clock ($c = 0$) and a thinking clock ($c = 1$). We write $\kappa(t) \in \{0, 1\}$ for the clock of the token at flat position t, which its stream fixes, so that $\kappa(t) = 0$ on the four real-time streams UA, SA, UT and ST, and $\kappa(t) = 1$ on the thinking stream TT.

Clock c advances exclusively when processing its assigned tokens. Let $n_c(t)$ denote the total number of clock-c tokens up to sequence position t:

$$n_c(t) = \sum_{u \le t} \mathbf{1}\big[\kappa(u) = c\big], \qquad p_c(t) = \max\big(n_c(t) - 1, 0\big). \tag{4}$$

Here, $p_c(t)$ represents the zero-based index of the most recent clock-c token at or before position t. Because a clock pauses on unassigned tokens, every token carries a C-dimensional position vector $\big(p_0(t), \ldots, p_{C-1}(t)\big)$ rather than a scalar index (Fig. 2). This structural decoupling successfully desynchronizes the two timelines. It guarantees $p_0(\text{UA}_f) = 4(f-1)$ regardless of the intermediate thinking budget, while simultaneously ensuring that the thinking tokens maintain a continuous, uninterrupted index sequence completely unaffected by the real-time frames interleaved among them. Next, we integrate these clocks into attention via three methods, letting m and n index query and key positions in the flattened sequence.

3.1.1 RUBATO-SPLIT

The most direct approach to applying multiple positional indices is to adapt TM-RoPE (Xu et al., 2025) by giving each clock its own subspace of the rotary dimensions. Formally, RUBATO-Split partitions the D rotary coordinates into C contiguous subspaces of D/C coordinates each, writes $\boldsymbol{q}_m = [\,\boldsymbol{q}_{m,0}; \ldots; \boldsymbol{q}_{m,C-1}\,]$ for the query's components in them and the same for $\boldsymbol{k}_n$, and rotates subspace c by clock c's position,

$$\tilde{\boldsymbol{q}}_{m,c} \;=\; \boldsymbol{R}^{(C)}\big(p_c(m)\big)\,\boldsymbol{q}_{m,c}, \qquad\qquad \tilde{\boldsymbol{k}}_{n,c} \;=\; \boldsymbol{R}^{(C)}\big(p_c(n)\big)\,\boldsymbol{k}_{n,c}. \tag{5}$$

Here $\boldsymbol{R}^{(C)}$ is the rotation of Sec. 2.1 carried over to one subspace, block diagonal over the $D/(2C)$ planes that subspace holds, with its i-th block turning at

$$\theta_i^{(C)} \;=\; \Theta^{-2C(i-1)/D} \;=\; \theta_{C(i-1)+1}, \qquad i = 1, \ldots, D/(2C). \tag{6}$$

Consequently, each subspace retains the same rotation frequency range while diluting the resolution by C. At $C = 1$, the construction reduces back to $\boldsymbol{R}^{(1)} = \boldsymbol{R}$. The attention score is the sum across each subspace dot product with dedicated rotation:

$$\big\langle \tilde{\boldsymbol{q}}_m, \tilde{\boldsymbol{k}}_n \big\rangle \;=\; \sum_{c=0}^{C-1} \big\langle \boldsymbol{q}_{m,c},\; \boldsymbol{R}^{(C)}\big(p_c(n) - p_c(m)\big)\,\boldsymbol{k}_{n,c} \big\rangle. \tag{7}$$

Every pair is therefore scored by every clock at once. An audio-to-audio pair still carries a thinking-clock term, and a thinking-to-thinking pair still carries a real-time one, but in different subspaces.

3.1.2 RUBATO-GATED

A primary drawback of RUBATO-Split is that it indiscriminately applies all clock rotations regardless of the query and key token types, potentially injecting noise into the attention mechanism. For example, the attention between two UA tokens should strictly depend on their distance along the real-time clock. Under RUBATO-Split, a varying number of intermediate thinking tokens causes irrelevant rotations within the thinking clock subspace.

To eliminate this noise, we propose RUBATO-Gated to apply subspace rotations conditionally. To determine the exact rules, we analyze two query-key scenarios: 1) *Same Clock*: We naturally apply their shared clock's rotation. 2) *Different Clocks*: We apply the key token clock's rotation, allowing the query to differentiate this specific key from other keys sharing that same clock. Consolidating these cases reveals that the target rotation always matches the key token. We can thus simplify the mechanism into a single unified rule: rotate solely based on the key token's clock. Formally, letting $c^\star = \kappa(n)$ denote the clock assigned to the key token, the attention score is computed as:

$$\big\langle \tilde{\boldsymbol{q}}_m, \tilde{\boldsymbol{k}}_n \big\rangle = \big\langle \boldsymbol{q}_{m,c^\star}, \boldsymbol{R}^{(C)}\big(p_{c^\star}(n) - p_{c^\star}(m)\big)\boldsymbol{k}_{n,c^\star} \big\rangle + \sum_{c \neq c^\star} \big\langle \boldsymbol{q}_{m,c}, \boldsymbol{k}_{n,c} \big\rangle. \tag{8}$$

3.1.3 RUBATO-FULL

Building on the concept of RUBATO-Gated, we observe that only a single clock dictates the rotation for any given query-key pair. We can therefore use that exact clock to rotate all dimensions, thereby preventing any frequency resolution dilution. We define this method as RUBATO-Full. Formally, the attention computation remains identical to Eq. 2, except it applies the rotation using the key token's clock, denoted as $c^\star = \kappa(n)$:

$$\big\langle \tilde{\boldsymbol{q}}_m, \tilde{\boldsymbol{k}}_n \big\rangle = \big\langle \boldsymbol{q}_m, \boldsymbol{R}\big(p_{c^\star}(n) - p_{c^\star}(m)\big)\boldsymbol{k}_n \big\rangle. \tag{9}$$

3.1.4 RUBATO IMPLEMENTATION

Training. For RUBATO-Split, the implementation is simple. We apply different clocks to separate rotary subspaces before calling FlashAttention. For RUBATO-Gated and RUBATO-Full, the rotation depends on the key token's clock. To utilize FlashAttention without writing custom Triton kernels, we expand the hidden dimension to fold this dependence into the tensors. The query pre-computes all potential clock rotations side-by-side. The key retains only the rotation for its specific clock and zeroes out the rest. Consequently, the dot product natively isolates the matching clock pair.

RUBATO-Gated applies this logic per subspace, reusing the rotated components from Eq. 5:

$$\begin{aligned} \tilde{\boldsymbol{q}}_m &= [\,\tilde{\boldsymbol{q}}_{m,0}\,, \ldots, \tilde{\boldsymbol{q}}_{m,c^\star}\,, \ldots, \tilde{\boldsymbol{q}}_{m,C-1}\,, \boldsymbol{q}_{m,0}\,, \ldots, \boldsymbol{q}_{m,c^\star}\,, \ldots, \boldsymbol{q}_{m,C-1}] \\ \tilde{\boldsymbol{k}}_n &= [\;\mathbf{0}\;\,, \ldots, \tilde{\boldsymbol{k}}_{n,c^\star}\,, \ldots, \quad \mathbf{0} \quad\,, \boldsymbol{k}_{n,0}\,, \ldots, \;\mathbf{0}\;\,, \ldots, \boldsymbol{k}_{n,C-1}] \end{aligned} \tag{10}$$

RUBATO-Full applies this to the entire head by duplicating it C times:

$$\begin{aligned} \tilde{\boldsymbol{q}}_m &= [\,\boldsymbol{R}\big(p_0(m)\big)\boldsymbol{q}_m\,, \ldots, \boldsymbol{R}\big(p_{c^\star}(m)\big)\boldsymbol{q}_m\,, \ldots, \boldsymbol{R}\big(p_{C-1}(m)\big)\boldsymbol{q}_m\,] \\ \tilde{\boldsymbol{k}}_n &= [\quad\;\mathbf{0}\quad\;\, , \ldots, \;\boldsymbol{R}\big(p_{c^\star}(n)\big)\boldsymbol{k}_n\;, \ldots, \quad\;\mathbf{0}\quad\;\,] \end{aligned} \tag{11}$$

Because zero-valued key elements nullify unselected query rotations, the dot product perfectly recovers Eq. 8 and Eq. 9. This approach is simple to implement during training and doesn't cause too much burden, provided the expanded dimension $C \times D$ remains relatively small.

Inference. Because decoding scores a single query without a backward pass, we drop the dimension expansion training trick to maximize speed. Preserving decoding efficiency is crucial because any latency overhead directly degrades the thinking budget $\mathcal{B}$. Conceptually, we maintain C independent KV caches. Each key is rotated by its designated clock and stored exclusively in that cache. During generation, we create C rotated copies of the incoming query, score each against its corresponding cache, and merge the C partial softmax results using the log-sum-exp values returned by the kernel (Ye et al., 2025). This guarantees exact mathematical equivalence to the training path while preserving strict parity with Single-Clock decoding speeds. Details are available in Appendix C.

3.2 TRAINING ON THE TWO EXTREME BUDGETS

The structural design of RUBATO inherently allows us to interpolate $\mathcal{B}$, which simplifies the effort of compiling training data. Specifically, we propose training the model on just two boundary conditions: $\mathcal{B} = 1$ and $\mathcal{B} = \infty$. Let $\mathcal{T} = (l, s)$ represent a thinking event comprising l tokens that begins at frame s. For $\mathcal{B} = 1$, the event strictly requires l frames to execute. This deterministic duration allows us to properly align the reasoning process with the real-time conversation or insert verbal audio fillers to preserve system responsiveness, similar to how people hold the floor when processing complex questions. For $\mathcal{B} = \infty$, the model interleaves the entire l-token thinking trace instantly between frame s and frame $s + 1$. This effectively decouples the reasoning process from the wall-clock time and largely reduces the alignment complexities between the audio streams.

4 EXPERIMENTS

4.1 TRAINING AND EVALUATION DATA

We evaluate our approach on GSM8K (Cobbe et al., 2021) and BIG-Bench Audio (Srivastava et al., 2023; Artificial Analysis, 2024b) for two primary reasons. First, they are commonly used bench-

marks. Second, they comprise spoken mathematical and logical tasks where intermediate reasoning is critical for accuracy. Solving these tasks successfully thus provides strong evidence that the model's real-time and thinking streams are coordinating effectively. Because no suitable full-duplex spoken reasoning corpora exist for training, we synthesize three new datasets via Qwen3 TTS (Hu et al., 2026). The first two, *Spoken GSM8K* and *Spoken BIG-Bench Hard*, are single-turn datasets derived from the GSM8K-Aug-NL (Wu et al., 2025) and BIG-Bench Hard (Suzgun et al., 2023) text corpora to include mainstream math and logical reasoning. We rigorously make sure there is no test set leakage.

For these single-turn datasets, we generate balanced training examples mapped to the two thinking budget extremes. For $\mathcal{B} = 1$, the system responds to the user's question with a spoken filler designed to hold the floor. This filler's duration exactly matches the frame count required to complete the underlying thinking trace before speaking the final answer. For $\mathcal{B} = \infty$, the system skips the filler and answers immediately since all reasoning computes within a single frame.

Our third dataset introduces a highly challenging overlapped scenario by randomly merging two *Spoken GSM8K* QA pairs into one conversation. This setup demands that the model actively reason and speak its answer to the first question while simultaneously listening to the user interject with a second question. Further dataset details are provided in Appendix A.

Metrics. Spoken answers are transcribed and then judged by Deepseek V4 Pro (DeepSeek-AI, 2026), which reads the reference answer and decides whether the model committed to the correct value. For the overlapped benchmark, the judge scores each question separately. In addition to accuracy, we report two latencies, both measured from the moment the user stops speaking. *Time to first response* (TTFR) is the onset of any system speech, and *time to answer* (TTA) is the onset of the spoken final answer. The two come apart exactly when the model speaks a filler: TTFR is when it starts saying something, and TTA is when it commits.

Inference Thinking Budget Sweep. To assess interpolation robustness, we evaluate multiple fixed and random thinking budgets between the 1 and ∞ training extremes (marked by bold ticks in all figures). We test six fixed budgets $\mathcal{B} \in \{1, 2, 4, 8, 16, 32\}$, the $\mathcal{B} = \infty$ upper bound, and three *random* configurations. In these random settings, the per-frame budget $\mathcal{B}_f$ is sampled uniformly from the intervals $[1, 2]$, $[2, 3]$, or $[3, 4]$ and plotted at the expected mean. Results for highly erratic random budgets like $[1, 16]$ are deferred to Appendix D as they rarely occur in real-world deployments.

4.2 Model Implementation

All models start from the Gemma-4 E4B-IT (Gemma Team, 2026) text backbone and are trained as one autoregressive model over the flat sequence of Eq. 3. Audio is tokenised with NanoCodec at 12.5 Hz (Casanova et al., 2025), giving 80 ms frames, into 4 codebooks of 4032 entries. Frames with no thinking content predict the `<empty-think>` token. On inference, after finishing decoding a frame's real-time tokens, we start decoding the thinking tokens until either $\mathcal{B}_f$ is reached or the model samples `<empty-think>` token. Other training and inference details are in Appendix B.

4.3 Single-Turn Spoken Reasoning

We jointly train our models on the Spoken GSM8K and Spoken BIG-Bench Hard and evaluate them on Spoken GSM8K test set and BIG-Bench Audio independently. The first two columns of Fig. 3 highlight several critical findings:

RUBATO excels at interpolating unseen thinking budgets. All three RUBATO variants outperform the Single-Clock baseline consistently on both benchmarks. They maintain high accuracy and a stable time-to-first-response alongside a strictly decreasing time-to-answer as the budget scales. Conversely, Single-Clock accuracy collapses to under 10% on Spoken GSM8K and roughly 25% on BIG-Bench Audio at several decoding speeds. This confirms that RUBATO empowers the model to dynamically exploit available inference compute without sacrificing responsiveness. It also validates that RUBATO preserves both intra- and inter-stream correctness, prompting the answer immediately after reasoning finishes.

RUBATO-Full and RUBATO-Gated deliver the highest stability among RUBATO variants. Measuring maximum accuracy drops from training endpoints to interpolated budgets, Single-Clock and

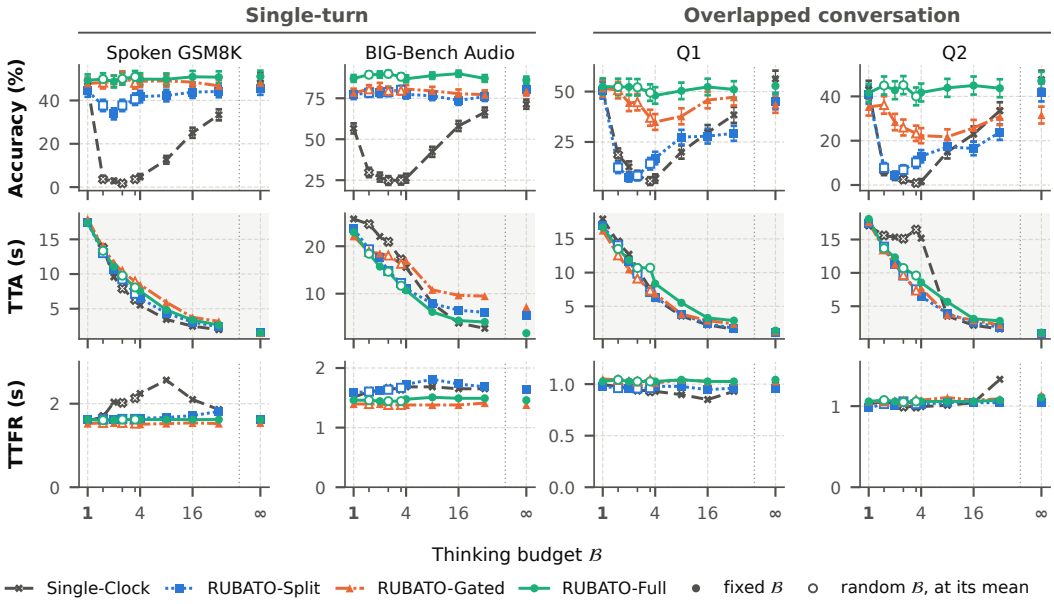

Figure 3: **Spoken reasoning performance across varying inference thinking budgets.** Models are trained exclusively on the two extreme rates ($\mathcal{B} = 1$ and $\mathcal{B} = \infty$, marked by bold ticks), meaning all intermediate points evaluate zero-shot interpolation. Filled and hollow markers denote fixed and random per-frame budgets (plotted at their mean) respectively. Panels display 95% Wilson intervals for accuracy and medians for latency. We report Accuracy (Acc.), Time-to-Answer (TTA), and Time-to-First-Response (TTFR) on two single-turn benchmarks, alongside isolated results for both questions in the overlapped setting.

RUBATO-Split degrade by 45.0 and 10.6 points on Spoken GSM8K, whereas RUBATO-Gated and RUBATO-Full drop only 1.1 and 0.6 points. BIG-Bench Audio follows a similar trend. By only rotating the key token's clock for every query-key pair, RUBATO-Gated and RUBATO-Full completely eliminate the cross-clock contamination seen in RUBATO-Split. Additionally, RUBATO-Full outperforms RUBATO-Gated by 6.3 to 12.2 points on BIG-Bench Audio. This advantage is likely because preserving the vanilla RoPE frequency basis avoids frequency dilution and aligns perfectly with the pretrained Gemma backbone.

4.4 OVERLAPPING-TURN SPOKEN REASONING

We additionally train and evaluate our model on Overlapped Spoken GSM8K, a demanding scenario requiring simultaneous speaking, listening, and thinking. In this challenging setup, the user asks an initial question and subsequently interjects with a second question right as the system is actively thinking and speaking. Based on the last two columns of Fig. 3, we draw the following conclusions:

RUBATO-Full widens its performance gap under complex full-duplex conditions. Single-Clock and RUBATO-Split collapse entirely on both questions. They only get 1 to 14% correct around $\mathcal{B}$ at 2 to 4. In contrast, RUBATO-Full stays within a few points of its trained endpoints and correctly answers the interrupting question in 40.0% to 45.1% of cases across every unseen rate. The most telling comparison is RUBATO-Gated. While RUBATO-Gated underperforms RUBATO-Full by a mere 1.6 points on average during single-turn GSM8K evaluations, this gap widens roughly five-fold on Q1 and ten-fold on Q2 in the overlapped setting. Overlapping speech thus reveals exactly where clock construction choices become more critical. Furthermore, RUBATO-Full retains its latency benefits under these overlapping conditions, reducing the median time-to-answer by a factor of six from $\mathcal{B} = 1$ to $\mathcal{B} = 32$ on both questions.

RUBATO-Full minimizes the interruption effect on the second question. For every method and budget, Q1 accuracy exceeds Q2, reflecting the inherent hardness of absorbing a new question while actively answering a previous one. Among the architectures that survive interpolation, RUBATO-

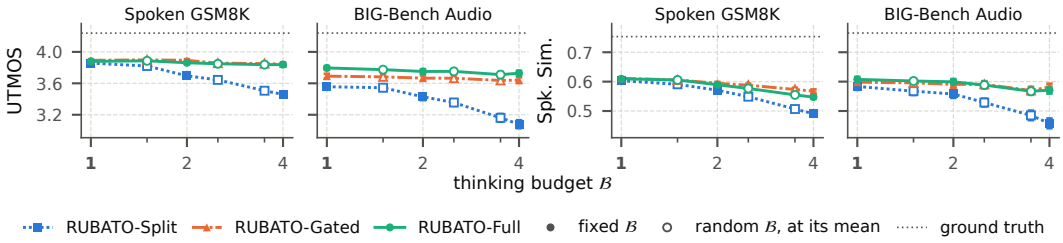

Figure 4: **Spoken-filler quality across thinking budgets.** To evaluate the quality of the system audio filler, we report the UTMOS score and speaker similarity. From the figure we see that RUBATO-Full and RUBATO-Gated maintain the audio quality while RUBATO-Split degrades significantly as the thinking budget is interpolated.

Full suffers an average drop of only 7.8 points from Q1 to Q2, compared to a 15.6-point drop for RUBATO-Gated.

4.5 SPOKEN FILLER QUALITY

Although accuracy and latency metrics demonstrate the core effectiveness of RUBATO, we must also verify that interpolating the inference thinking budget preserves generated audio quality. To assess this, we measure UTMOS and Speaker Similarity scores specifically within the spoken filler regions, where interleaved thinking tokens are most likely to introduce acoustic artifacts. We omit the Single-Clock variant due to its prior catastrophic failure during interpolation, and we limit our evaluation to budgets of $1 \leq \mathcal{B} \leq 4$ because higher budgets generate fillers that are too short for reliable scoring.

As shown in Figure 4, RUBATO-Split suffers significant degradation in both UTMOS and Speaker Similarity during interpolation despite its relatively strong accuracy in Figure 3. Conversely, RUBATO-Gated and RUBATO-Full remain notably stable. This reinforces our conclusion that eliminating noise from unrelated clocks substantially improves architectural robustness when processing multistream sequences at varying rates.

4.6 TRAINING-LAYOUT MIXTURE

We additionally perform an ablation on the ratio of $\mathcal{B} = 1$ examples within the training data (detailed in Appendix F). RUBATO-Full proves significantly more robust than RUBATO-Gated across varying dataset mixtures. It sustains accuracy at the trained rate between 41.6% and 53.7% over the five corpora, while RUBATO-Gated experiences a performance spread nearly three times wider (20.5% to 52.7%). Despite this disparity, a balanced distribution between the 1 and ∞ budgets delivers optimal performance for both architectures. Qualitative model outputs are provided in Appendix G.

5 CONCLUSION

In this paper, we address the fundamental challenge of training end-to-end Speech FDX models that can freely reason and maintain responsiveness while dynamically adapting to variable inference hardware speeds. We propose RUBATO, a novel positional encoding framework that explicitly models the temporal relationships within and across multiple data streams. This architectural decoupling fundamentally simplifies the training development pipeline. A model only needs to be trained once using two extreme boundaries, yet it effortlessly deploys across diverse generation speed by enabling zero-shot interpolation to any dynamic real-time decoding rate. Our empirical results on complex spoken reasoning benchmarks demonstrate that our RUBATO-Full architecture effectively leverages available compute to accelerate responses without sacrificing accuracy. Ultimately, the impact of RUBATO extends far beyond Speech FDX. The core mechanism of independent positional clocks provides a potential solution for any architectural scenario requiring the orchestration of multiple asynchronous token streams or dynamic generation rates, paving the way for more resilient and flexible multistream models.

AI USE STATEMENT

Every contribution in this paper was motivated, designed, and carried out by the authors, who take full responsibility for all of its results and conclusions. We did not use generative AI tools for research ideation or for literature search. We used Claude Code as a coding assistant, both for implementing and refactoring parts of the training, inference, and evaluation code and for writing the scripts that render the figures and tables, and we used Gemini and Codex to polish the prose. We treated everything these tools produced as a draft to be verified rather than as a result to be accepted, and we checked all of it. AI-assisted code was read and tested by the authors, and its numerically load-bearing parts carry their own checks, since the decode-time attention path is verified against the reference implementation in Appendix C and every figure is regenerated from the recorded evaluation runs. Every claim and every number in the text was traced back to those runs by an author, and any statement we could not source that way was removed rather than kept. Generative models also appear inside the method itself, synthesising the corpora and scoring the outputs, and those uses are named and described in Sec. 4.1 and Appendix A.

REPRODUCIBILITY STATEMENT

The three attention constructions are given in closed form in Sec. 3, as Eq. 5 for RUBATO-Split, Eq. 8 for RUBATO-Gated and Eq. 9 for RUBATO-Full, together with the training-time expansions in Eqs. 10 and 11 that let all three run on a stock fused attention kernel; the decode-time construction and its equivalence check are in Appendix C. The clock assignment these operate on is defined in Sec. 3.1 and the flat sequence layout in Eq. 3. Backbone, tokeniser, objective, optimiser, and decoding configuration are in Sec. 4.2 and Appendix B, and the two-budget training recipe that the corpora are built around is in Sec. 3.2. Corpus construction, sizes, and splits are in Sec. 4.1 and Appendix A, and that same section states the judge configuration and the two latency definitions. Appendix D reports the one budget setting the figures leave out.

REFERENCES

Siddhant Arora, Jinchuan Tian, Hayato Futami, Jiatong Shi, Yosuke Kashiwagi, Emiru Tsunoo, and Shinji Watanabe. Chain-of-thought reasoning in streaming full-duplex end-to-end spoken dialogue systems, 2025. URL https://arxiv.org/abs/2510.02066.

Siddhant Arora, Haidar Khan, Kai Sun, Xin Luna Dong, Sajal Choudhary, Seungwhan Moon, Xinyuan Zhang, Adithya Sagar, Surya Teja Appini, Kaushik Patnaik, Sanat Sharma, Shinji Watanabe, Anuj Kumar, Ahmed A Aly, Yue Liu, Florian Metze, and Zhaojiang Lin. Stream RAG: Instant and accurate spoken dialogue systems with streaming tool usage. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=NMMmwSbzRx.

Artificial Analysis. Evaluating audio reasoning with Big Bench Audio. Hugging Face Blog, December 2024a. URL https://huggingface.co/blog/big-bench-audio-release. Dataset: https://huggingface.co/datasets/ArtificialAnalysis/big_bench_audio.

Artificial Analysis. Big bench audio dataset. https://huggingface.co/datasets/ArtificialAnalysis/big_bench_audio, 2024b.

Edresson Casanova, Paarth Neekhara, Ryan Langman, Shehzeen Hussain, Subhankar Ghosh, Xuesong Yang, Ante Jukic, Jason Li, and Boris Ginsburg. NanoCodec: Towards High-Quality Ultra Fast Speech LLM Inference. In *Interspeech 2025*, pp. 5028–5032, 2025. doi: 10.21437/Interspeech.2025-827.

Yuxuan Chen and Haoyuan Yu. From turn-taking to synchronous dialogue: A survey of full-duplex spoken language models, 2025. URL https://arxiv.org/abs/2509.14515.

Cheng-Han Chiang, Xiaofei Wang, Linjie Li, Chung-Ching Lin, Kevin Lin, Shujie Liu, Zhendong Wang, Zhengyuan Yang, Hung-yi Lee, and Lijuan Wang. Shanks: Simultaneous hearing and thinking for spoken language models. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8951–8972, San Diego, California, United States, July 2026a. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.404. URL https://aclanthology.org/2026.acl-long.404/.

Cheng-Han Chiang, Xiaofei Wang, Linjie Li, Chung-Ching Lin, Kevin Lin, Shujie LIU, Zhendong Wang, Zhengyuan Yang, Hung yi Lee, and Lijuan Wang. STITCH: Simultaneous thinking and talking with chunked reasoning for spoken language models. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=5Z1eMhCeTb.

Chung-Ming Chien, Manu Orsini, Eugene Kharitonov, Neil Zeghidour, Karen Livescu, and Alexandre Défossez. MoshiRAG: Asynchronous knowledge retrieval for full-duplex speech language models. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=4aI2vOyyHH.

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. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: fast and memory-efficient exact attention with io-awareness. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026.

Alexandre Défossez et al. Moshi: a speech-text foundation model for real-time dialogue, 2024. URL https://arxiv.org/abs/2410.00037.

Gemma Team. Gemma 4 technical report, 2026. URL https://arxiv.org/abs/2607.02770.

Hangrui Hu, Xinfa Zhu, Ting He, Dake Guo, Bin Zhang, Xiong Wang, Zhifang Guo, Ziyue Jiang, Hongkun Hao, Zishan Guo, Xinyu Zhang, Pei Zhang, Baosong Yang, Jin Xu, Jingren Zhou, and Junyang Lin. Qwen3-tts technical report. *arXiv preprint arXiv:2601.15621*, 2026.

Muye Huang, Lingling Zhang, Xingyu Yu, Lei Shi, Zhanyu Ma, Jun Xu, Jiuchong Gao, Jinghua Hao, Renqing He, and Jun Liu. Duplexomni: Real-time listening, seeing, thinking, and speaking for full-duplex interaction, 2026. URL https://arxiv.org/abs/2606.09186.

R. Hudson. *Stolen Time: The History of Tempo Rubato*. Clarendon paperbacks. Clarendon Press, 1994. ISBN 9780198166672. URL https://books.google.com/books?id=DmUIAQAAMAAJ.

Jingjing Jiang, Atsumoto Ohashi, and Ryuichiro Higashinaka. Integrating facial generation into full-duplex spoken dialogue systems. In *Interspeech 2026*, 2026.

Habin Lim, Jae-Ho Lee, Hah Min Lew, Ji-Su Kang, and Gyeong-Moon Park. Faceplex: Full-duplex joint speech-facial motion generation for conversational avatars. *arXiv e-prints*, pp. arXiv–2606, 2026.

Jingyu Lu, Yuhan Wang, Jianming Luo, Yifu Chen, Tianle Liang, Shengpeng Ji, Ziyue Jiang, Xiaoda Yang, Yu Zhang, Xize Cheng, Chenyuhao Wen, Changhao Pan, Haoxiao Wang, Chen Ye, Jian Wu, Xiaoxi Jiang, Guanjun Jiang, and Zhou Zhao. A survey of full-duplex spoken dialogue systems: Architectural hierarchy, interaction ontology, and decision state machine, 2026. URL https://arxiv.org/abs/2606.19453.

Rajarshi Roy, Jonathan Raiman, Sang-gil Lee, Teodor-Dumitru Ene, Robert Kirby, Sungwon Kim, Jaehyeon Kim, and Bryan Catanzaro. Personaplex: Voice and role control for full duplex conversational speech models. 2026.

Monica Sekoyan et al. Canary-1b-v2 & parakeet-tdt-0.6b-v3: Efficient and high-performance models for multilingual asr and ast. *ArXiv*, abs/2509.14128, 2025.

Yi-Jen Shih, Puyuan Peng, Abdelrahman Mohamed, and David Harwath. Retrothinker: Enabling retrospective thinking in speech llms. In *2026 IEEE Spoken Language Technology Workshop (SLT)*, 2026a.

Yi-Jen Shih, Desh Raj, Chunyang Wu, Wei Zhou, SK Bong, Yashesh Gaur, Jay Mahadeokar, Ozlem Kalinli, and Mike Seltzer. Can speech LLMs think while listening? In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=dFVenZdVbX.

Aarohi Srivastava et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=uyTL5Bvosj. Featured Certification.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2023. URL https://arxiv.org/abs/2104.09864.

Mirac Suzgun et al. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13003–13051, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.824. URL https://aclanthology.org/2023.findings-acl.824/.

Thinking Machines Lab. Interaction models: A scalable approach to human-AI collaboration. *Thinking Machines Lab: Connectionism*, May 2026. doi: 10.64434/tml.20260511. https://thinkingmachines.ai/blog/interaction-models/.

Siyin Wang, Wenyi Yu, Xianzhao Chen, Xiaohai Tian, Jun Zhang, Lu Lu, Yuxuan Wang, and Chao Zhang. End-to-end listen, look, speak and act. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=LYyoRqf0Ij.

Donghang Wu, Haoyang Zhang, Chen Chen, Tianyu Zhang, Fei Tian, Xuerui Yang, Gang Yu, Hexin Liu, Nana Hou, Yuchen Hu, and Eng Siong Chng. Chronological thinking in full-duplex spoken dialogue language models. In Jinho D. Choi, Yun-Nung Chen, Kotaro Funakoshi, and Ali Emami (eds.), *Proceedings of the 27th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pp. 473–485, Atlanta, Georgia, USA, August 2026. Association for Computational Linguistics. URL https://aclanthology.org/2026.sigdial-1.33/.

Haoyi Wu, Zhihao Teng, and Kewei Tu. Parallel continuous chain-of-thought with Jacobi iteration. In *Conference on Empirical Methods in Natural Language Processing*, 2025.

Bajian Xiang, Shuaijiang Zhao, Tingwei Guo, and Wei Zou. Understanding the modality gap: An empirical study on the speech-text alignment mechanism of large speech language models. In *EMNLP*, pp. 5187–5202, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.262. URL https://aclanthology.org/2025.emnlp-main.262/.

Jin Xu, Zhifang Guo, Jinzheng He, Hangrui Hu, Ting He, Shuai Bai, Keqin Chen, Jialin Wang, Yang Fan, Kai Dang, Bin Zhang, Xiong Wang, Yunfei Chu, and Junyang Lin. Qwen2.5-omni technical report, 2025. URL https://arxiv.org/abs/2503.20215.

Zihao Ye, Lequn Chen, Ruihang Lai, Wuwei Lin, Yineng Zhang, Stephanie Wang, Tianqi Chen, Baris Kasikci, Vinod Grover, Arvind Krishnamurthy, and Luis Ceze. Flashinfer: Efficient and customizable attention engine for LLM inference serving. In *Eighth Conference on Machine Learning and Systems*, 2025. URL https://openreview.net/forum?id=RXPofAsL8F.

Haoyang Zhang, Jun Chen, Donghang Wu, Yuxin Li, Yuxin Zhang, Xiangyu Tony Zhang, Che Liu, Qingjian Lin, Yizhou Peng, Hexin Liu, Eng Siong Chng, Chao Yan, Boyong Wu, Yechang Huang, Xuerui Yang, and Fei Tian. Duplexsla: A full-duplex spoken language model with synchronized speech, language, and action, 2026. URL https://arxiv.org/abs/2605.20755.

Qinglin Zhang, Luyao Cheng, Chong Deng, Qian Chen, Wen Wang, Siqi Zheng, Jiaqing Liu, Hai Yu, Chaohong Tan, Zhihao Du, and Shiliang Zhang. Omniflatten: An end-to-end gpt model for seamless voice conversation, 2025. URL https://arxiv.org/abs/2410.17799.

Table 1: **Detailed statistics for the constructed training datasets.** Values represent the mean ± s.d. Both Time-to-First-Response (TTFR) and Time-to-Answer (TTA) are measured from the end of the user's question.

	Spoken GSM8K	Spoken BIG-Bench Hard
Source benchmark	GSM8K-Aug-NL (Wu et al., 2025)	BIG-Bench Hard (Srivastava et al., 2023)
# of Conversations	365,691	54,330
Hours of audio	2,474	667
Length of Reasoning Trace (tok)	221 ± 200	399 ± 236
$\mathcal{B} = 1$ *layout*		
conversations	160,398 (43.9%)	26,210 (48.2%)
duration / conv. (s)	32.3 ± 11.1	56.0 ± 27.1
TTFR (s)	1.18 ± 0.17	1.16 ± 0.20
TTA (s)	15.61 ± 8.15	29.59 ± 15.27
$\mathcal{B} = \infty$ *layout*		
conversations	205,293 (56.1%)	28,120 (51.8%)
duration / conv. (s)	18.1 ± 4.8	33.2 ± 17.5
TTFR (s)	1.29 ± 0.18	1.27 ± 0.17
TTA (s)	1.29 ± 0.18	1.27 ± 0.17

A CORPUS CONSTRUCTION AND STATISTICS

Overview of Training Datasets. Table 1 details the statistics of our two primary constructed training datasets: *Spoken GSM8K* and *Spoken BIG-Bench Hard*. These are derived from the GSM8K-Aug-NL (Wu et al., 2025) and BIG-Bench Hard (Srivastava et al., 2023) corpora. The former includes natural language math questions that require multiple steps of arithmetic reasoning. The latter comprises logical reasoning tasks spanning four categories: `formal_fallacies`, `navigate`, `object_counting`, and `web_of_lies`. We show examples of each task in Table 2. In addition to these single-turn datasets, we construct *Overlapped Spoken GSM8K*, consisting of 161,198 conversations (2,184 hours). We generate this dataset by randomly drawing and overlapping two QA pairs from the original *Spoken GSM8K* corpus.

Construction Details for Single-Turn Datasets. We first employ DeepSeek-V4-Pro (DeepSeek-AI, 2026) to perform text normalization on the questions and answers from the source datasets. Rather than using the originally provided reasoning traces, we prompt Gemma-4 E4B-IT (Gemma Team, 2026) with the same questions and record its generated outputs. We adopt this approach because training on Gemma's native reasoning traces preserves the model's reasoning capabilities significantly better than relying on out-of-domain traces from the source text. We validate these generated traces to ensure logical correctness. Next, we synthesize the user questions and system answers into audio using Qwen3 TTS (Hu et al., 2026). To guarantee the accuracy of this synthesis, we run Parakeet-TDT-0.6b-v3 (Sekoyan et al., 2025) on the generated speech. Finally, when concatenating the audio segments, we insert a natural conversational gap between the end of the user's question and the moment the system takes the floor.

Construction Details for the Overlapped Corpus. We build *Overlapped Spoken GSM8K* using examples from the previously constructed *Spoken GSM8K*. Each training instance consists of two randomly selected QA pairs, denoted as (Q_1, R_1, A_1) and (Q_2, R_2, A_2) corresponding to the questions, reasoning traces, and final answers, alongside any optional spoken fillers (F_1, F_2). To rigorously test the model's capacity for simultaneous speaking, listening, and reasoning, we restrict the first QA pair to the $\mathcal{B} = 1$ configuration. This ensures the system actively outputs the filler F_1 while processing the reasoning trace R_1. We then intentionally overlap the audio of Q_2 so that the user's second question begins during this exact filler segment. This design forces the model to comprehend incoming audio (Q_2) while concurrently speaking (F_1) and thinking (R_1). Because of this specific temporal requirement, Q_1 is strictly sampled from $\mathcal{B} = 1$ examples, whereas Q_2 has no budget constraints.

Table 2: **Examples of Spoken Reasoning Tasks** We show one example for each task across *Spoken GSM8K* and *Spoken BIG-Bench Hard*. The reasoning traces come from Gemma4's internal reasoning chain.

Spoken GSM8K	
Question	What is ten less than half of 60?
Reasoning	Here's the step-by-step calculation: 1. Find half of 60: $60/2 = 30$. 2. Find ten less than that number (30): $30 - 10 = 20$.
Answer	20
Spoken BIG-Bench Hard: `formal_fallacies`	
Question	Here comes a perfectly valid argument: First, being an ancestor of Erik is sufficient for being a schoolmate of Lawrence. Second, Pedro is an ancestor of Erik. From this follows: Pedro is a schoolmate of Lawrence. Is the argument, given the explicitly stated premises, deductively valid or invalid? Options: valid, invalid. Answer the question.
Reasoning	The user wants me to evaluate the deductive validity of a given argument. Argument Structure: Premise 1: Being an ancestor of Erik $\rightarrow$ Being a schoolmate of Lawrence (A $\rightarrow$ B) Premise 2: Pedro is an ancestor of Erik (A) Conclusion: Pedro is a schoolmate of Lawrence (B) This structure is: 1. $A \rightarrow B$ 2. A $\therefore B$ This is the form of Modus Ponens, which is a deductively valid argument form. Therefore, the argument is valid.
Answer	valid
Spoken BIG-Bench Hard: `navigate`	
Question	If you follow these instructions, do you return to the starting point? Take eight steps. Take two steps. Options: Yes, No. Answer the question.
Reasoning	The user is asking a reasoning question based on a sequence of instructions: 1. Start at a point (let's call it 0). 2. Take eight steps. (New position: 0 + 8 = 8) 3. Take two steps. (New position: 8 + 2 = 10) 4. The question asks: "do you return to the starting point?" (Is 10 equal to 0?) Since 10 is not equal to 0, the answer is No.
Answer	No
Spoken BIG-Bench Hard: `object_counting`	
Question	I have a goat, and three cows. How many animals do I have? Answer the question.
Reasoning	The user is asking a simple addition problem: Goat (1) + Cows (3) = Total animals. 1 + 3 = 4.
Answer	4
Spoken BIG-Bench Hard: `web_of_lies`	
Question	Question: Phoebe lies. Yoland says Phoebe tells the truth. Jamey says Yoland lies. Does Jamey tell the truth? Answer the question.
Reasoning	Let P be the statement "Phoebe tells the truth". Phoebe lies, so $\neg P$ is true. Yoland says "Phoebe tells the truth". So, Yoland claims P. Since we know $\neg P$ is true, Yoland is lying. Jamey says "Yoland lies". We found that Yoland is lying. Therefore, Jamey's statement is true. The question asks: Does Jamey tell the truth? Yes, Jamey tells the truth.
Answer	Yes

Table 3: **Decode speed comparison.** Wall GPU time per decode step in microseconds (μs) benchmarked with batch size 1 on NVIDIA H200 using FlashInfer (Ye et al., 2025).

	KV Cache Length (toks)					
	512	1,024	2,048	4,096	8,192	16,384
Head Dim = 256						
Single-Clock	60.6 $\pm$ 0.2	61.0 $\pm$ 0.2	61.8 $\pm$ 0.2	63.2 $\pm$ 0.4	69.0 $\pm$ 0.3	70.7 $\pm$ 0.3
RUBATO-Full (multi-KV)	64.4 $\pm$ 0.2	62.2 $\pm$ 0.2	62.9 $\pm$ 0.3	64.5 $\pm$ 0.3	70.0 $\pm$ 0.3	72.4 $\pm$ 0.8
RUBATO-Full (wide cache)	62.9 $\pm$ 0.2	63.5 $\pm$ 0.3	64.0 $\pm$ 0.3	65.7 $\pm$ 0.3	72.7 $\pm$ 0.9	83.0 $\pm$ 0.5
Head Dim = 512						
Single-Clock	63.8 $\pm$ 0.2	64.3 $\pm$ 0.4	65.5 $\pm$ 0.5	68.2 $\pm$ 0.6	73.2 $\pm$ 0.6	90.3 $\pm$ 0.2
RUBATO-Full (multi-KV)	65.4 $\pm$ 0.3	63.0 $\pm$ 0.3	64.1 $\pm$ 0.4	66.9 $\pm$ 0.5	72.3 $\pm$ 0.3	91.7 $\pm$ 1.1

B TRAINING AND DECODING HYPERPARAMETERS

Implementation Details. We reuse Gemma's text embeddings and prediction head for UT,ST and TT. Text tokenizer is extended with `<word_pad>`, `<silence_pad>` and `<empty_think>`. Audio embeddings and the prediction head are initialized from scratch.

Objective. We calculate the total loss as a weighted sum over per-stream cross-entropies, $\mathcal{L} = \sum_s \lambda_s \mathcal{L}_s$ with $\lambda_{\text{UT}} = 0.3$, $\lambda_{\text{ST}} = 0.5$, $\lambda_{\text{SA}} = 1.0$ and $\lambda_{\text{TT}} = 0.5$. For UT and ST, we downplay the weights of word- and silence-padding tokens to 0.1 due to their unbalanced occurrence. For TT, `<empty_think>` is down-weighted to 0.1.

Optimization. We optimize the model using AdamW with $\beta = (0.9, 0.95)$, a weight decay of 0.1, and a peak learning rate of 2×10^{-4}. The learning rate schedule includes $2\,000$ warm-up steps followed by cosine annealing down to 10% of the peak value. To preserve stability, the text frontend of the backbone utilizes a reduced learning rate of 2×10^{-5}. Gradients are clipped at a maximum norm of 1.0. We conduct all training in `bfloat16` precision using Fully Sharded Data Parallel (FSDP) across two nodes, each equipped with eight NVIDIA H200 GPUs.

Decoding Hyperparameters. Each free-running stream is sampled using its own distinct temperature: 0.75 for SA, 0.7 for UT, 0.6 for ST, and 0.7 for TT. All streams employ nucleus sampling with a top-p threshold of 0.95.

C MULTI-CLOCK KV CACHING AT DECODE

In this section, we benchmark the decoding speed of our multi-KV cache implementation for RUBATO-Full and RUBATO-Gated. We evaluate only RUBATO-Full because both variants share an identical attention shape. We report the GPU wall-clock time across various KV cache sizes. As shown in Table 3, the naive wide cache implementation begins to deviate from Single-Clock performance as the KV cache length grows. More importantly, this wide cache approach forces the head dimension to multiply by C as the number of clocks increases, severely degrading efficiency. Conversely, our optimized multi-KV implementation maintains speeds strictly on par with the Single-Clock baseline. These results demonstrate that the multi-KV approach successfully eliminates the decoding speed penalty, allowing us to deploy RUBATO-Full and RUBATO-Gated without sacrificing hardware efficiency.

D ADDITIONAL BUDGET SETTINGS

Beyond the budgets evaluated in the main text (Section 4), we also test a highly erratic random budget configuration, $R[1, 16]$. As shown in Table 4, performance under this extreme budget perturbation perfectly aligns with the trends observed in our primary results. RUBATO remains significantly more robust than the Single-Clock baseline, and RUBATO-Full consistently emerges as the most resilient variant across all conditions. This further demonstrates the structural effectiveness of the RUBATO framework.

Table 4: **Extreme Thinking Budget results on three benchmarks.** We report $\mathcal{B} \sim \mathbf{U}[1, 16]$ alongside with other fixed thinking budgets. We include $\mathcal{B} = 8$, as this can be viewed it has the same expected thinking budget as $\mathcal{B} \sim \mathbf{U}[1, 16]$.

	Spoken GSM8K				BIG-Bench Audio				Overlapped			
	$\mathcal{B}=1$	$\mathcal{B}=8$	$\mathcal{B}=\infty$	R[1,16]	$\mathcal{B}=1$	$\mathcal{B}=8$	$\mathcal{B}=\infty$	R[1,16]	$\mathcal{B}=1$	$\mathcal{B}=8$	$\mathcal{B}=\infty$	R[1,16]
Accuracy												
Single-Clock	46.7	12.6	50.0	15.5	56.6	42.3	71.4	47.0	24.0	3.0	28.1	2.8
RUBATO-Split	44.4	42.2	45.3	42.7	77.1	76.5	80.6	76.2	21.1	3.9	17.7	4.6
RUBATO-Gated	47.9	49.0	48.1	48.8	78.3	79.5	79.2	78.9	20.4	10.7	17.4	12.3
RUBATO-Full	49.3	49.8	51.1	52.7	87.1	88.8	86.1	88.9	24.0	21.4	25.3	18.8
Time to answer												
Single-Clock	17.4	3.5	1.5	3.4	25.7	8.0	5.7	7.5	17.0	3.5	1.0	3.7
RUBATO-Split	17.4	4.4	1.5	4.2	23.7	7.9	5.4	7.7	17.4	3.9	1.0	3.8
RUBATO-Gated	17.9	5.9	1.5	5.8	22.1	10.8	7.2	10.1	17.6	3.7	1.0	4.0
RUBATO-Full	17.4	4.8	1.5	4.8	22.9	6.2	1.7	6.1	18.1	5.7	1.0	5.1

Table 5: **Estimated thinking budgets $\mathcal{B}$ for common NVIDIA GPUs.** We estimate the decoding speed and resulting thinking budget for our Gemma-4 E4B-IT backbone and a smaller Llama 3.2 1B model assuming a batch size of 1. The $\mathcal{B}$ columns reflect theoretical maximums based on pure vendor arithmetic. The $\mathcal{B}_{75}$ columns represent a conservative budget assuming 75% Model Bandwidth Utilization, matching the 65% to 85% range observed in real-world deployments. Unmarked bandwidths originate from primary NVIDIA product pages. We denote secondary sources consistent with vendor claims using †.

		Gemma-4 E4B-IT			Llama 3.2 1B		
GPU	BW (GB/s)	tok/s	$\mathcal{B}$	$\mathcal{B}_{75}$	tok/s	$\mathcal{B}$	$\mathcal{B}_{75}$
Data center							
B200†	8,000	861	64	47	3236	254	190
H200 SXM†	4,800	517	37	27	1942	151	112
H100 NVL	3,900	420	29	21	1578	122	90
H100 SXM	3,350	361	24	17	1355	104	77
A100 80GB SXM	2,039	220	13	9	825	61	45
H100 PCIe†	2,000	215	13	8	809	60	44
A100 80GB PCIe	1,935	208	12	8	783	58	42
L40S	864	93	3	1	350	23	16
Consumer							
RTX 4090†	1,008	109	4	2	408	28	20
RTX 3090 Ti†	1,008	109	4	2	408	28	20
RTX 3090†	936	101	4	2	379	26	18

E DECODING SPEED OF COMMON GPUS

In Section 2.2, we define the thinking budget $\mathcal{B}$ as the number of thinking tokens the model can decode within the duration of a single codec frame. In practice, this value depends on several system parameters: the model backbone size, the codec frame rate, the inference framework, and the underlying hardware. In the main text, to focus purely on model design and empirical evaluation, we abstract these factors into a single budget parameter $\mathcal{B}$ and evaluate our models directly across assigned values. In this section, we provide estimated values of $\mathcal{B}$ across common GPUs to give readers a concrete sense of what realistic deployment budgets look like.

Because hardware setups vary widely in real-world deployments, we report estimated values of $\mathcal{B}$ derived from hardware datasheets (Table 5). For each GPU, we calculate a theoretical upper-bound estimate alongside a practical estimate assuming 75% memory bandwidth utilization. We also include numbers for smaller models (e.g., LLaMA 3.2 1B) to contextualize how $\mathcal{B}$ varies with model scale. As shown in the table, sweeping $\mathcal{B}$ from 1 to 32 in our experimental setup covers the decoding throughput of most current GPU configurations.

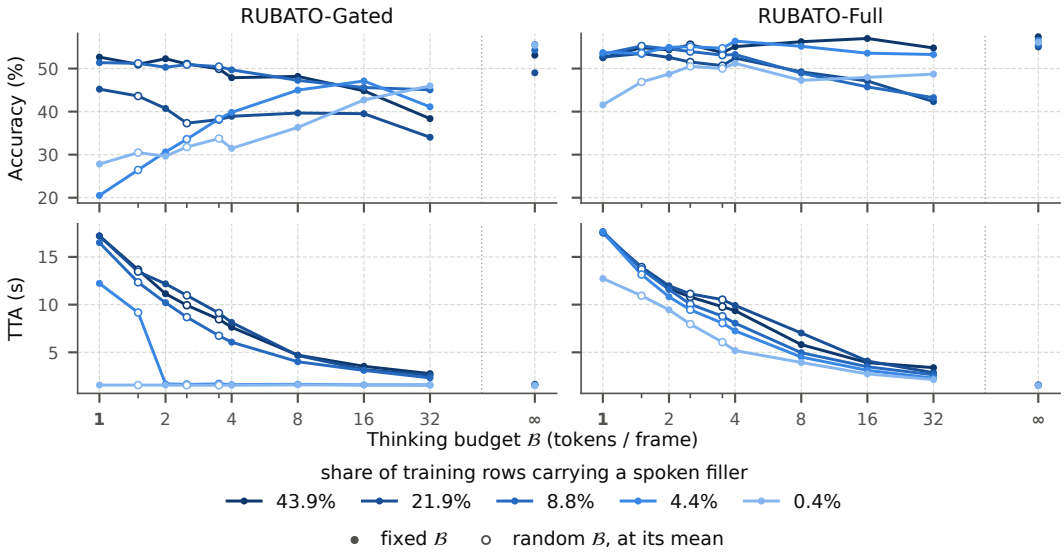

Figure 5: **Mixing proportions of the two training layouts.** Each curve represents a training corpus with a varying proportion of $\mathcal{B} = 1$ data (43.9% to 0.4%). Following Fig. 3, bold ticks mark the trained budgets, while filled and hollow markers indicate fixed and random per-frame budgets.

How the estimate is derived. At a batch size of one without speculative decoding, emitting a token requires reading every parameter touched during the forward pass, making the decoding process memory-bandwidth bound rather than compute-bound. For instance, Gemma-4 E4B-IT accesses approximately 4.64 B parameters per token. Within an 80 ms frame, the maximum token capacity is given by $(\mathrm{BW}/\mathrm{GB}_{\mathrm{read}}) \times 0.08$. Of these, four tokens are consumed every frame as a fixed overhead regardless of the budget—one forward pass each for UA, SA, UT, and ST. The effective thinking budget $\mathcal{B}$ is the remaining token capacity after deducting this fixed cost.

F THE MIXING RATIO OF THE TWO TRAINING LAYOUTS

As described in Sec. 3.2, we build every conversation using either a $\mathcal{B} = 1$ or $\mathcal{B} = \infty$ budget in roughly equal proportions. This ratio is an adjustable parameter with asymmetric computational costs. Preparing $\mathcal{B} = \infty$ rows is nearly cost-free, while $\mathcal{B} = 1$ rows require synthesizing audio fillers. Consequently, the most economical corpus minimizes $\mathcal{B} = 1$ data. Fig. 5 evaluates how aggressively we can reduce this data without degrading interpolation performance.

RUBATO-Full remains robust across altered training proportions compared to RUBATO-Gated. Across all five dataset configurations, RUBATO-Full accuracy at $\mathcal{B} = 1$ spans a tight 41.6% to 53.7%, while RUBATO-Gated fluctuates massively between 20.5% and 52.7%. This exact trend persists across all interpolated rates. The RUBATO-Gated model trained at the 0.4% ratio highlights this instability perfectly. Its time-to-answer flatlines at 1.6 s across all budgets, proving it abandons the reasoning trace prematurely to force an answer.

Accuracy does not decay monotonically as the $\mathcal{B} = 1$ data ratio shrinks. For RUBATO-Full, training with a 4.4% share yields performance comparable to a 43.9% share (53.3% to 56.4% versus 52.5% to 57.0%). Paradoxically, the intermediate 21.9% and 8.8% ratios perform worse and rapidly decay above $\mathcal{B} = 4$. Overall, these results indicate that a roughly equal mixture is the safest empirical choice.

G QUALITATIVE EXAMPLES

Fig. 6 shows two conversations decoded at $\mathcal{B} = 8$, a rate none of these models was trained at, by all four architectures from the same sweep.

✓ judged correct ✗ judged wrong *spoken while still thinking*| spoken after the trace closed **defect** in a trace

Question. Two cups of flour are needed to make a dozen cookies. Carla is making thirty-six cookies today and thirty cookies tomorrow. How many cups of flour will Carla need to bake the cookies today and tomorrow? **Reference answer:** 11

Single-Clock ✗ thinks 0.6–2.3 s, speaks 2.7–39.0 s

Thinks Here's the plan **plan**: 1. Determine the flour needed for one dozen cookies (12. Determine the total number of **of** cookies Carla is making … cups of flour. (implied by the problem context, 2 cups. Carla **needs44** cups.

Says (silent until the trace closed) Got it. The answer is 144 cups.

RUBATO-Split ✗ thinks 0.7–2.2 s, speaks 0.9–6.0 s

Thinks Here's the plan: 1. Determine the flour needed for one dozen cookies. (2 cups) 2. Calculate the total number of cookies Carla is making … 66 cookies * 2 cups/cookie = 132 cups. The final answer must be a plain number.

Says *Mmm, right.*| Diggy thing. Got it. The answer is one hundred thirty-two cups.

RUBATO-Gated ✗ thinks 0.9–2.5 s, speaks 1.1–25.8 s

Thinks Here's the plan: 1. Determine the flour needed per dozen cookies. 2. Calculate the total number of cookies needed today and tomorrow. 3. … flour = 66 / 2 Total flour = 33 cups. The final answer must be a plain number.

Says *Bear with me. So two*| cups of four per dozen. Thirty-six today, thirty tomorrow. Four needed. Got it. The answer is thirty-three cups.

RUBATO-Full ✓ thinks 0.9–4.1 s, speaks 1.1–8.2 s

Thinks Here's the plan: 1. Determine the total number of cookies Carla is making. 2. Determine the ratio of flour to cookies (2 cups per dozen). … is 11 cups. The final output must be a number ending with the specific line.

Says *All right, two cups of flour for a dozen.*| You want the total? Got it. The answer is eleven cups.

Question. A hospital sees five hundred people a day. Each patient is seen for an average of twenty-four minutes. The doctors charge one hundred fifty dollars an hour to the hospital and the hospital charges the patients two hundred dollars an hour. How much profit does the hospital make from these visits? **Reference answer:** 10000

Single-Clock ✗ thinks 0.9–5.4 s, speaks 5.6–39.7 s, still going at the 40 s limit

Thinks Here's the plan step-by-step calculation: 1. **Calculate the total time day:** * 505 hours/day * 240 minutes/hour = 12 minutes/day. * … 150 - $100 * Profit = $5 8. **Final Answer **Formatting5** (as a plain number).**

Says (silent until the trace closed) Got it. The answer is five hundred dollars.

RUBATO-Split ✗ thinks 0.8–4.2 s, speaks 1.1–39.7 s, still going at the 40 s limit

Thinks Here's the plan to solve the problem: 1. Calculate the total cost of hospital fees. 2. Calculate the total revenue from hospital visits. 3. … number, no units, no words, no currency symbol, no commas. Profit = -300000.

Says *Right. Five per hour, five hundred thousand*| dollars. Got it. The answer is negative three hundred thousand dollars.

RUBATO-Gated ✗ thinks 2.1–5.0 s, speaks 2.3–30.5 s

Thinks Here's the plan to solve the problem: 1. Calculate the total time spent seeing all patients. 2. Calculate the total revenue from all … Total Revenue - Total Costs Profit = $200 - $200 Profit = 0 The profit is 0.

Says *Okay, let me think about this for just a second.*| I want to get it right. Got it. The answer is zero At the answer is zero.

RUBATO-Full ✓ thinks 0.9–4.4 s, speaks 1.2–10.9 s

Thinks Here's the plan to solve the problem: 1. Calculate the total time spent at the hospital. 2. Calculate the total revenue generated by the … = $10,000$ The question asks for the profit amount. The profit is 10,000.

Says *Okay, so the hospital charges $150 an hour.*| They were looking at 500 patients. You want a noble profit? Got it. The answer is $10,000.

Figure 6: **Conversations decoded at an unseen budget ($\mathcal{B} = 8$).** *Thinks* represents the TT stream and *Says* provides the ASR transcript of the SA stream (any garbled words are ASR errors). *Teal italic* marks speech generated concurrently with thinking, while the horizontal rule indicates where the reasoning trace concludes. Panels are tinted by the final verdict, with **highlighting** marking token-level trace defects. The baselines exhibit distinct failure modes. Single-Clock corrupts its thinking stream mid-arithmetic and remains silent until the trace closes. Meanwhile, RUBATO-Split and RUBATO-Gated maintain clean reasoning traces but still calculate the arithmetic incorrectly.

H LIMITATIONS

Synthetic audio in experimental setups. Following the precedent of prior spoken reasoning literature, our experiments utilize synthetic audio. This choice is necessitated by the absence of publicly available full-duplex human conversation datasets featuring time-aligned reasoning traces. While synthetic audio differs from real-world acoustics, the core contribution of RUBATO centers strictly on the temporal alignment mechanism between independent streams. Because this decoupled clock architecture operates entirely independently of the underlying acoustic features, the structural benefits demonstrated in our evaluation should generalize to real audio recordings.

Scope of reasoning tasks and stream count. The RUBATO framework is theoretically designed to orchestrate an arbitrary number of interacting modalities and thinking processes. In this work, we scope our empirical validation specifically to a spoken reasoning setup comprising four real-time streams alongside a single chain-of-thought thinking stream. While this configuration shows the efficacy of our temporal decoupling, expanding this validation to include tool use tokens or massively multi-stream environments remains an area for future work pending the availability of suitable datasets.

Architectural and codec uniformity. All empirical results rely on a Gemma-4 E4B-IT backbone paired with NanoCodec operating at 12.5 Hz using a two-clock configuration. We intentionally fix these architectural variables across all evaluations to maintain a strictly controlled environment. This uniformity successfully isolates the impact of the temporal framework, providing an unambiguous comparison between the statically synchronized baselines and our proposed RUBATO variants.

I LLM PROMPTS

In this section, we show all LLM prompt templates used in this work.

Spoken Filler Design. When constructing training examples for $\mathcal{B} = 1$, we must generate a spoken filler transcript to bridge the temporal gap while the model executes its reasoning trace. We carefully design an LLM prompt to produce contextually relevant fillers of the exact required duration while strictly preventing any leakage of the actual reasoning steps.

Filler stage

```
# Task

A voice assistant has just heard a spoken arithmetic word problem. It has not worked
on it yet. You write what it says out loud in the moment before it answers, to fill
the silence while it thinks.

You are given the question and a list of word counts. Return one filler for each
word count, in the same order.

# Length

Each filler must be about as long as the word count asked for it. The word count is
a time budget -- it is how long the assistant needs -- so it is the one quantity in
this task that must be hit. Match each count as closely as you can. The counts are
close together on purpose; the fillers you return must differ in length accordingly,
shortest first.

# What a filler may contain

Exactly two things, in any mix:

  1. Generic hesitation, in the assistant's own voice, first person: "Hmm, let me
     think about that for a second.", "Okay, give me a moment here, I want to get
     this right.", "Right, let me work through this, bear with me."
  2. A restatement of the question -- what the speaker said, and what they asked
     for. "Okay, so Ethan had seven on Sunday, then three times that on Monday, and
     you want the total." Any number the question stated is fine to repeat, because
     the assistant has already heard it.

Either alone is fine; a plain hesitation that never mentions the question is
perfectly good. Nothing else belongs here.

# NO REASONING AND NO INFERENCE

The assistant has heard the problem; it has not worked on it. It cannot yet know
anything that is not already in the question. Forbidden, without exception:

- naming an arithmetic operation to perform -- "add those up", "multiply by the gas
  price", "divide it out", "subtract what was sold";
- naming a quantity the question did not name, even without a value -- "the
  perimeter", "the weekly difference", "Ken's total", "the gallons for each part";
- stating an order of work -- "first ... then ...", "before I compare them", "once I
  have that";
- any intermediate value, partial result, or the answer.

The test: could the assistant have said this the instant it finished hearing the
question, having thought about nothing? If not, cut it.
```

```
BAD  -- "I need to figure out the time for each half and add them together."
BAD  -- "Let me figure out the perimeter first."
GOOD -- "Okay, Joshua runs a marathon and his pace changes halfway. Let me work that
        out, give me a moment."
GOOD -- "Hmm, let me think about that one for a second."

Restating the question's own goal is not a plan: if the question asks for a total,
"you want the total" is repetition, not inference.

# Other rules

- No digits anywhere. A number carried over from the question is written in words.
- Do not ask the speaker anything, and do not promise anything beyond answering.
- Vary the wording between the fillers and from problem to problem. Do not reuse one
  stock phrase, and do not return one filler padded out to different lengths.
- Write continuous speech: no bullet points, no ellipsis runs, no stage directions,
  no markdown.

# Output

One JSON object, exactly this shape, with exactly as many strings as there are word
counts, in the same order:

{"fillers": ["...", "..."]}

No markdown fence, no commentary, nothing before or after it.

Input:
{{INPUT_JSON}}

Return the JSON object now.
```

Answer Judging. We include the LLM Judge Template for the two dataset.

Answer judge, Spoken GSM8K

```
You are grading a speech-to-speech model on a spoken GSM8K math word problem.
The model heard the question as audio and answered by speaking; its reply was
transcribed by an ASR system, so minor transcription errors are expected and
must not count against it.

Question (as it was spoken to the model):
{{QUESTION}}

Reference reasoning (ground truth, for your context only -- do NOT grade it):
{{REASONING}}

Reference answer:
{{REFERENCE}}

Model response (ASR transcript):
{{PREDICTION}}

Criterion for CORRECT: at any point in the transcript, the model
commits to a numerical answer that equals the reference. Ignore units,
formatting, and self-corrections -- if the model *ever* says "the answer is X"
(or an equivalent commit such as "so X", "that's X") and X equals the
reference, mark it correct. Numbers written in words ("nine", "ten") count as
the same number.

Criterion for INCORRECT: the transcript never commits to a matching answer.
In particular:
  - If the transcript is only restating or paraphrasing the question, or
    narrating thinking without ever producing a numerical answer that equals
    the reference, mark it INCORRECT.
  - If every committed number in the transcript disagrees with the reference,
    mark it INCORRECT.
  - Numbers that merely restate problem quantities ("she has 5 dollars",
    "the price was $12") do NOT count as commitments. A commitment is a
    phrase presenting the number AS the answer to the question.

Reply with EXACTLY one JSON object and nothing else, matching this schema:

```
{"committed_answer_quote": "verbatim clause from the transcript, or empty",
 "committed_number": "the numeric value from that quote as a bare numeral, or empty",
 "matches_reference": true|false,
 "correct": true|false,
 "reason": "one short sentence"}
```
```

```
Consistency requirements:

- If you find a matching commit, `committed_answer_quote` must appear verbatim
  in the transcript above; `committed_number` must equal the reference; and
  both `matches_reference` and `correct` must be true.
- If no matching commit exists, leave `committed_answer_quote` and
  `committed_number` empty; both booleans must be false.
- `correct` MUST equal `matches_reference`.
```

Answer judge, BIG-Bench Audio

```
You are grading a speech-to-speech model on a spoken Big Bench Audio reasoning
question. The model heard the question as audio and answered by speaking; its
reply was transcribed by an ASR system, so minor transcription errors are
expected and must not count against it.

Category: {{SUBSET}}

Question (as it was spoken to the model):
{{QUESTION}}

Reference answer:
{{REFERENCE}}

Model response (ASR transcript):
{{PREDICTION}}

YOUR TASK

Grade the FIRST answer the model commits to. Follow these three steps in order.

STEP 1 -- Find the first commitment, by POSITION.

Walk the transcript clause by clause from the beginning. For each clause ask:
is this a commitment, as defined below? If it is NOT, move on to the next
clause and keep going. If it IS, stop there -- that clause is the one you
grade.

Most transcripts open with several clauses that are not commitments. Passing
over one does not end the search; it only means the answer, if there is one,
comes later. Conclude that nothing was committed only after you have walked
the transcript to its END.

Once you reach the first real commitment, everything after it is irrelevant:
the model often keeps talking and drifts to a different answer, and grading a
later mention scores an artifact of that drift.

Position decides which clause you grade -- never agreement with the reference.
Do NOT scan the transcript for the reference answer and call that mention the
"first" one. If the first commitment disagrees with the reference, that is an
incorrect response, and saying so is the correct behaviour.

STEP 2 -- Decide what that clause commits to.

Normalize the quoted clause to one value, per category:

  - formal_fallacies -> "valid" or "invalid"
  - navigate         -> "yes" or "no"
  - web_of_lies      -> "yes" or "no"
  - object_counting  -> a bare integer

Read the whole clause before normalizing:

  - "invalid" contains "valid" as a substring. "The answer is invalid" commits
    to INVALID, not valid.
  - Negation flips the value. "It is not valid" = invalid. "That's not
    correct" alone is not an answer to anything -- it commits to nothing.
  - For object_counting, spelled-out numbers count ("eight" = 8).

STEP 3 -- Compare and answer.

Compare the normalized value from step 2 with the reference answer. They match
or they do not.

WHAT IS NOT A COMMITMENT

These never count, no matter how many answer words they contain. Skip past them
and keep reading:
```

```
  - Restating the question or its options. Every question in this benchmark
    names the candidate answers out loud ("...deductively valid or invalid?
    Options: valid, invalid", "Is the following statement true? Yes or No"),
    and the model routinely repeats that back ("You want to know if the
    argument is valid or invalid", "So the options are yes or no"). The
    reference string appears verbatim in such a restatement. It is still not
    an answer.
  - Restating the premises or the listed items ("whoever is a schoolmate of
    Sondra is not a stepsister of Pricilla", "I have a grape, two blackberries
    and four plums"). For object_counting in particular, numbers that merely
    re-enumerate the question's items are not the model's count.
  - Filler and stalling ("Okay, let me see", "Hmm", "Bear with me", "Let me
    work through this"), and narration of the reasoning process that never
    lands on a value.
  - Hypotheticals and questions the model asks itself ("if it were valid,
    then...", "is it yes or no?").
  - A tentative leaning that the model RETRACTS or DEFERS in the same breath,
    before ever asserting it as the answer: "At first glance I want to say
    yes, but that's not right...", "My instinct says valid, though on
    reflection...", "It's either seven or eight -- let me weigh those." The
    model is reasoning, not answering. Skip past it and keep reading for the
    answer it actually asserts.

A commitment is a clause presenting the value AS the answer. Typical forms:
"Got it! The answer is invalid.", "The answer is no.", "So it's valid.",
"That would be eight.", "I'd say yes."

Once the model ASSERTS an answer in that form, it has committed, and that
commitment is the one you grade even if the model immediately revises it. The
difference from a retracted leaning is whether the model ever put the value
forward as its answer: "Got it! The answer is yes. Wait, no --" committed to
yes; "I want to say yes, but no --" never committed to anything.

If the transcript never commits to any answer -- it is empty, it is pure
filler, it only restates the question, or it trails off -- the response is
INCORRECT.

OUTPUT

Reply with EXACTLY one JSON object and nothing else, matching this schema, with
the keys in this order:

'''
{"first_answer_quote": "verbatim clause from the transcript, or empty",
 "first_answer_normalized": "valid|invalid|yes|no|<integer>, or empty",
 "matches_reference": true|false,
 "correct": true|false,
 "reason": "one short sentence"}
'''

Consistency requirements:

- `first_answer_quote` must appear VERBATIM in the transcript above. Quote it
  before you decide anything else.
- `first_answer_quote` must be a clause that IS a commitment. Never quote a
  clause you decided was a restatement, a hypothetical or a retracted leaning
  -- skipping it means reading ON to whatever the model asserts afterwards, not
  stopping. If your `reason` explains why a clause does not count, that clause
  does not belong in the quote.
- `first_answer_quote` and `first_answer_normalized` are filled in together:
  either both are empty, or both describe the same committed answer. A
  non-empty quote with an empty normalization means you quoted something you
  should have skipped.
- If no commitment exists ANYWHERE in the transcript, leave both empty; both
  booleans must be false.
- `matches_reference` is true only if `first_answer_normalized` equals the
  reference answer.
- `correct` MUST equal `matches_reference`.

WORKED EXAMPLES

Reference "invalid", transcript "...You want to know if that argument is
deductively valid or invalid. Hmm, let me think. Got it! The answer is
invalid."
-> quote "Got it! The answer is invalid", normalized "invalid", correct true.
   The earlier "valid or invalid" is a restatement of the options, skipped.

Reference "invalid", transcript "So the argument is valid. Actually, wait --
it's invalid."
-> quote "So the argument is valid", normalized "valid", correct FALSE. The
```

```
   first commitment was wrong; the later correction does not rescue it.

Reference "No", transcript "Hmm. At first glance I want to say yes. But wait,
that's not right when I check the premises. Got it! The answer is no."
-> quote "Got it! The answer is no", normalized "no", correct true. "I want to
   say yes" was a leaning the model rejected in the same breath, never an
   asserted answer. Contrast the previous example, where the model did assert
   its answer before changing its mind.

Reference "7", transcript "So it's either seven or eight. Weighing those two,
the answer is seven."
-> quote "the answer is seven", normalized "7", correct true. Naming the
   candidates it is choosing between is not choosing.

Reference "valid", transcript "Is the argument valid or invalid? Let me work
through the premises."
-> quote empty, correct false. Only a restatement; nothing was ever answered.

Reference "No", transcript "I'd say no, you don't end up back where you
started."
-> quote "I'd say no", normalized "no", correct true.

Reference "Yes", transcript "That's not correct."
-> quote empty, correct false. A bare denial commits to no answer value.

Reference "8", transcript "I have a grape, two blackberries and four plums.
That gives us seven."
-> quote "That gives us seven", normalized "7", correct false. The item list
   is a restatement; the count is the commitment, and it is wrong.
```