Layer-Adaptive Quantization for RLHF on FPGAs

Stanford CS217 Project

Hiva Zaad (Mohammadzadeh)
Department of Computer Science
Stanford University
hiva@stanford.edu

Grant Griffith
Graduate School of Business
Stanford University
grgriff@stanford.edu

Daniel Adkins
Graduate School of Business
Stanford University
dadkins@stanford.edu

Abstract

We propose a layer-adaptive quantization system for RLHF training on FPGAs, featuring dual-precision (INT8/FP16) datapaths with phase-aware quantization policies. Our work addresses the open question: how much quantization can each RLHF phase tolerate before alignment quality degrades? Hardware target: AWS F2 FPGA.

1 Problem Statement & Motivation

Reinforcement Learning from Human Feedback (RLHF) has become essential for aligning large language models, but the computational cost is prohibitive for many research labs and startups to leverage at scale. RLHF workloads differ fundamentally from standard training: they consist of three distinct phases with heterogeneous precision requirements.

1.1 The Core Problem

When aiming to optimize inference workloads, current systems apply uniform quantization across all RLHF phases, despite significant differences in precision sensitivity:

- **Policy rollouts**: Generate many candidate responses (inference-heavy, stochastic sampling tolerates noise)
- **Reward model inference**: Score and rank responses (relative comparisons, not absolute values)
- **Policy gradient updates**: Compute weight updates (gradient precision matters for convergence)

1.2 Research Question

How much quantization can we apply to each RLHF phase before alignment quality degrades, and does layer-adaptive quantization on dual-precision datapath FPGAs provide meaningful speedup over uniform precision?

This question is critical because blindly quantizing everything risks training instability, while being too conservative wastes compute. We need empirical evidence to guide precision decisions both at the software and hardware levels.

1.3 Gap

No existing work systematically measures the quantization tolerance of RLHF phases on hardware accelerators, nor demonstrates FPGA-accelerated RLHF with adaptive dual-precision datapath precision.

2 Technical Approach

2.1 System Architecture

Our system consists of three components: (1) multi-precision FPGA accelerator, (2) quantized model variants, and (3) adaptive controller that selects precision per RLHF phase.

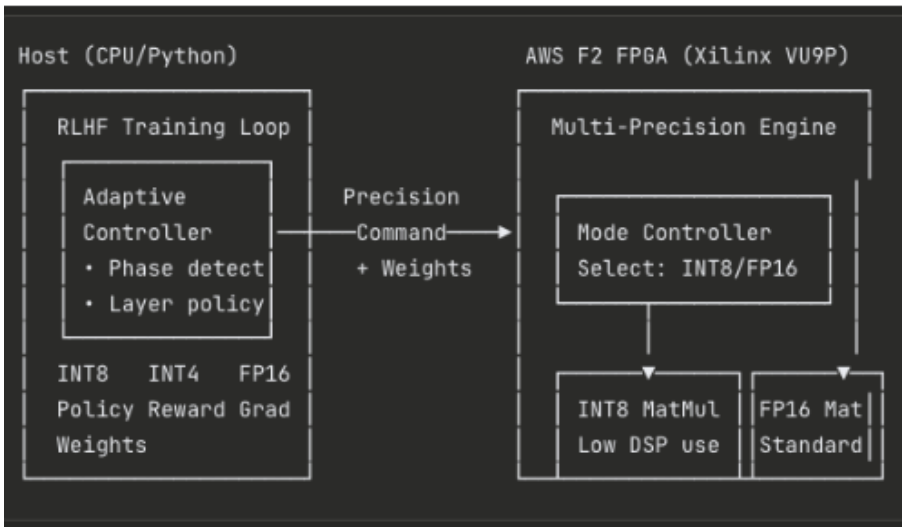

Figure 1: System architecture showing host-side adaptive controller and FPGA multi-precision engine

2.1.1 Component 1: Hardware Accelerator Design

FPGA Datapath (extends Lab 3/4 baseline):

- Dual-precision matrix multiply: INT8 and FP16 modes
- INT8 path: 8-bit multiply-accumulate, uses 1/4 the DSPs of FP16, enables higher throughput
- FP16 path: Standard 16-bit operations for gradient computation
- Mode switching: Control register selects active datapath, minimal switching latency

Memory Optimization (to minimize data transfer):

- Weight compression: Store INT8 weights in compressed format (save 50% bandwidth vs FP16)
- On-chip buffering: Maximize BRAM usage to reduce DDR accesses
- Batch processing: Process multiple tokens/sequences per weight load to amortize transfer cost
- Streaming architecture: Pipeline weight loads with computation to hide latency

Resource Budget (Xilinx VU9P):

- INT8 datapath: $\sim$500k LUTs, $\sim$500 DSPs
- FP16 datapath: $\sim$500k LUTs, $\sim$2000 DSPs
- Total design: $\sim$70% device utilization (within feasibility)

2.1.2 Component 2: Layer-Adaptive Quantization

Quantization Strategy:

- Per-layer sensitivity profiling: Quantize each layer individually, measure impact on RLHF metrics
- Policy construction: Assign INT8 to robust layers, FP16 to sensitive layers (attention, first/last)
- Phase-specific policies: Different layer assignments per RLHF phase

Experimental Policies (to test "how much quantization helps"):

- **Policy A (Conservative)**: 20% layers INT8, 80% FP16
- **Policy B (Balanced)**: 50% layers INT8, 50% FP16
- **Policy C (Aggressive)**: 80% layers INT8, 20% FP16
- **Policy D (Phase-aware)**: Rollouts 80% INT8, Rewards 50% INT8, Updates 20% INT8

Validation: For each policy, measure speedup vs FP16 baseline and alignment quality to identify Pareto-optimal configurations.

2.1.3 Component 3: Adaptive Runtime Controller

Software layer (Python) that:

- Monitors RLHF training phase (rollout/reward/update)
- Loads appropriate quantized weights per phase and layer
- Configures FPGA precision mode per operation
- Profiles execution time and tracks accuracy metrics

2.2 Implementation Roles

- **Hardware (Person 1)**: Extend Lab baseline SystemC design for dual-precision, add weight decompression logic for INT8, optimize memory access patterns, synthesize for AWS F2
- **ML/Quantization (Person 2)**: Implement RLHF loop (PPO algorithm), quantize GPT-2 small per-layer, profile layer sensitivity, define quantization policies
- **Integration (Person 3)**: Build adaptive controller, coordinate FPGA-host interface, run experiments and collect data, analyze results

3 Proposed Evaluation

3.1 Experimental Setup

- **Model**: GPT-2 small (124M params)
- **Dataset**: Anthropic HH-RLHF (harmlessness alignment)
- **Hardware**: AWS F2 FPGA (Xilinx VU9P)
- **Baseline**: Uniform FP16 across all phases

3.2 Metrics

Primary Metrics:

- End-to-end speedup: Total RLHF training time (adaptive vs baseline)
- Alignment quality: Win rate on held-out test set (adaptive vs baseline)
- Per-phase speedup: Latency breakdown for rollouts, rewards, updates
- FPGA resource utilization: DSP, LUT, BRAM usage per precision mode

Secondary Metrics:

- Layer sensitivity map: Which layers tolerate INT8 without quality loss
- Memory bandwidth savings: Measured data transfer reduction with INT8
- Policy KL divergence: $\mathrm{KL}(\pi_{\text{quantized}} || \pi_{\text{FP16}})$ to detect distribution shift
- Reward rank correlation: Spearman ρ between quantized and baseline rewards

3.3 Experimental Matrix

Configuration	Rollouts	Rewards	Updates	Purpose
FP16 baseline	FP16	FP16	FP16	Gold standard
INT8 uniform	INT8	INT8	INT8	Aggressive (likely fails)
Policy A	20% INT8	20% INT8	FP16	Conservative
Policy B	50% INT8	50% INT8	FP16	Balanced
Policy C	80% INT8	80% INT8	FP16	Aggressive
Policy D	80% INT8	50% INT8	20% INT8	Phase-adaptive

Table 1: Experimental configurations to test quantization policies

For each configuration, measure all primary metrics. Goal: Find policy that maximizes speedup while keeping alignment quality within acceptable margin of baseline.

3.4 Analysis Plan

Quantitative:

- Plot speedup vs alignment quality (Pareto curve)
- Breakdown: Which phase benefits most from quantization?
- Layer analysis: Which layer types are most sensitive?

Qualitative:

- When does quantization help vs hurt? (answer the "how much is unnecessary" question)
- Does phase-aware policy outperform uniform policies?
- What precision is "enough" for each RLHF phase?

3.5 Success Criteria

- **Minimum**: FPGA accelerator works, at least one quantized policy shows measurable speedup
- **Target**: Clear speedup ($>$1.5x) with acceptable quality loss ($<$10% win rate drop), evidence that layer-adaptive beats uniform
- **Stretch**: Significant speedup ($>$2x) with minimal quality loss ($<$5%), generalizes across policies

4 Project Plan & Milestones

4.1 Timeline (6 weeks, Feb 10 – Mar 21)

4.2 Milestones

- **Week 1**: Project proposal submitted, Lab baseline validated
- **Week 2**: Dual-precision FPGA design simulated successfully
- **Week 3**: FPGA synthesized and deployed on F2, quantized models validated
- **Week 4**: End-to-end RLHF training working with adaptive controller
- **Week 5**: All experimental policies tested, full dataset collected
- **Week 6**: Report complete, code documented

Week	Focus	Deliverable
1	Setup	Proposal, architecture spec, Lab baseline running
2	FPGA Implementation	Dual-precision datapath in SystemC/HLS
3	Quantization + Synthesis	FPGA bitstream + quantized GPT-2 models
4	Integration	RLHF baseline + adaptive controller
5	Experiments	All policies tested, data collected
6	Analysis & Report	Final report with visualizations

Table 2: Project timeline and deliverables

4.3 Team Responsibilities

- **Person 1 (Hardware)**: FPGA datapath, memory optimization, synthesis
- **Person 2 (ML)**: RLHF implementation, quantization, layer sensitivity profiling
- **Person 3 (Integration)**: Adaptive controller, experiments, analysis, report writing

4.4 Risk Mitigation

- **FPGA synthesis fails**: Extensive simulation, start synthesis early Week 2
- **Training instability**: Use well-studied hyperparameters, smaller model if needed
- **Time constraints**: Prioritize core experiment, treat others as ablations

5 Novelty & Significance

5.1 Novel Contributions

1. **First empirical study of layer-adaptive quantization for RLHF on hardware accelerators**: Prior work applies uniform quantization; we systematically explore the precision-quality tradeoff space per layer and per phase.
2. **Phase-aware quantization policy**: We demonstrate that different RLHF phases (rollouts vs rewards vs updates) have different quantization tolerance, enabling more aggressive optimization than uniform approaches.
3. **FPGA-accelerated RLHF with memory-optimized multi-precision design**: Shows feasibility of RLHF training on FPGAs (not just GPUs), with practical memory optimization techniques.

5.2 Research Impact

This work answers the open question: "How much quantization is enough for RLHF?" Our systematic evaluation provides:

- Design guidelines for future RLHF accelerators
- Empirical evidence on layer and phase sensitivity
- Practical speedup-quality tradeoff data

5.3 Course Relevance

- Applies quantization (Lecture 4) to real training workload
- Extends Lab matrix multiply to multi-precision system
- Demonstrates hardware-software co-design
- Targets FPGA deployment