

Real-Time Recurrent Qubit-State Discrimination on a Heterogeneous AI-Engine–FPGA Accelerator

Vu Le*

University of Massachusetts Amherst
Amherst, MA, USA
vdle@umass.edu

Yilun Xu

Lawrence Berkeley National
Laboratory
Berkeley, CA, USA
yilunxu@lbl.gov

Neel Vora

Lawrence Berkeley National
Laboratory
Berkeley, CA, USA
nrvora@lbl.gov

Gang Huang

Lawrence Berkeley National
Laboratory
Berkeley, CA, USA
ghuang@lbl.gov

Deepak Ganesan

University of Massachusetts Amherst
Amherst, MA, USA
dganesan@umass.edu

Phuc Nguyen

University of Massachusetts Amherst
Amherst, MA, USA
vp.nguyen@cs.umass.edu

Abstract

Qubit readout, or qubit state discrimination, remains one of the slowest and most error-prone operations in superconducting quantum processors, making fast, accurate, and hardware-efficient classification essential for scalable quantum computing. Conventional discriminators integrate the I/Q record into a single point, discarding evidence a relaxing qubit leaves behind; full-resolution feed-forward classifiers recover it but are too costly to deploy under this constraint. Recurrent classifiers retain full resolution far more cheaply, but their loop-carried hidden state cannot fit a single AI Engine (AIE) tile’s register capacity and spills to data memory, adding latency that grows with sequence length.

We present a hardware-aware LSTM accelerator for an AMD Versal device that resolves this by partitioning at the dependency boundary—-independent input projections on the AIE array, the loop-carried update in programmable logic (PL)—keeping pace with a 64 ns input cadence under a 1,000 ns budget. On the qubit affected by mid-readout state transitions, our design matches a full-resolution feed-forward network’s accuracy (96.59% vs. 96.37%) using 403× fewer parameters and roughly 103× fewer DSPs while cutting relative error by 68.2% over GMM, 43.6% over QubiCML, and 16.82% over MF-NN. Using 4 out of 400 AIE tiles and 17 DSP58, it needs 2.2–8.2× fewer DSP58 blocks than QubiCML and MF-NN, and the AIE–PL partition cuts exposed tail latency by 60.3% versus a pure-AIE mapping and DSP58 usage 31.1× versus an all-PL implementation. Across eight qubits, a compact variant reaches 96.78% mean accuracy, the strongest hardware-deployed operating point. Temporal resolution and recurrent-state placement must then be treated jointly when designing low-latency quantum readout accelerators.

1 Introduction

Qubit readout faces a basic timing trade-off: a longer window collects more signal but increases exposure to relaxation and

can improve separation between quantum states. However, it also increases the probability that an excited qubit relaxes during measurement. When this occurs, early samples reflect the prepared $|1\rangle$ state, while later samples increasingly resemble $|0\rangle$. Conventional classifiers integrate these samples into a single I/Q point, discarding when the transition occurred and potentially moving the readout toward the wrong decision region [13, 17, 21].

This creates a fundamental challenge for low-latency readout. A shorter window reduces exposure to relaxation but collects less signal. A longer window collects more signal, but integration can combine evidence from different states and make the resulting representation less reliable. Retaining the full temporal sequence avoids this information loss, but increases the classifier input from two integrated values to thousands of samples. Since signal acquisition and classification must fit within the same real-time feedback deadline, a longer readout window simultaneously produces more data and leaves less time to process it.

We observe this behavior in readout traces from eight superconducting qubits. For seven qubits, classifiers operating on an integrated I/Q point perform nearly as well as classifiers that retain the complete time series. One qubit behaves differently. On this qubit, which we refer to as Q3, a population of prepared $|1\rangle$ shots transitions toward $|0\rangle$ during readout. As the window grows, GMM accuracy decreases because integration increasingly mixes the early $|1\rangle$ evidence with the later $|0\rangle$ signal. A full-resolution classifier preserves this temporal evidence and improves accuracy from 89.27% with GMM and 93.95% with QubiCML to 96.37%.

Existing classifiers occupy opposite ends of the resulting accuracy–cost tradeoff. GMM [15] and QubiCML [32] are inexpensive because they consume only one integrated I/Q point, but they discard temporal information that matters on Q3. A full-resolution FCN [20] recovers this information by consuming all 2,816 real values in a 704 ns trace, but requires millions of parameters and a significant fraction

*Also with Lawrence Berkeley National Laboratory.

of the FPGA’s resources, making it impractical to deploy (Figure 1). The central question is therefore: how can we retain temporal information when it matters while meeting both a sub-microsecond latency target and a practical per-channel hardware budget?

Recurrent inference. Recurrent models offer a promising middle ground. An LSTM processes the readout incrementally and summarizes the samples observed so far in a fixed-size hidden state [14, 17]. It can therefore retain temporal structure without requiring a feed-forward network whose parameter count grows with the complete input record. In our case, the resulting full-resolution LSTM has $403\times$ fewer parameters than the FCN while matching its accuracy on Q3. Recurrence also lets classification overlap with acquisition instead of waiting for the complete readout record to arrive.

However, a compact model is not necessarily a fast or inexpensive hardware implementation. Every LSTM timestep depends on the state produced by the previous timestep, and the state update must complete before the next segment of the readout arrives. At our operating point, the 2 GSPS converter produces one 256-value input segment every 64 ns. A pure-AIE implementation cannot sustain this rate because the loop-carried state exceeds the available register capacity and spills to local memory. An all-PL implementation avoids this scheduling problem, but consumes 26.9% of the device’s DSP58 budget. Thus, the model is compact, but it cannot be implemented efficiently on either AIE or PL alone.

Our approach. Our key insight is that both temporal resolution and hardware placement should be chosen where they provide value. At the algorithm level, we use a full-resolution recurrent model for a qubit such as Q3, where readout timing affects the decision. For qubits that do not benefit from the complete trace, we use a compact recurrent model that preserves coarse temporal structure while reducing each timestep to one mean I/Q pair. This avoids imposing the cost of Q3’s classifier on every channel.

At the hardware level, we partition the LSTM according to its data dependencies. The four input projections are independent and execute in parallel on AIE tiles, while the loop-carried recurrent update and classifier remain in PL. This keeps the hidden state local and allows each state update to complete within the 64 ns input cadence. The AIE computation and PL recurrent update overlap as samples arrive, leaving only the final pipeline tail exposed after acquisition.

Technical challenges. Realizing this approach requires addressing two challenges. First, selecting a recurrent model cannot be based on accuracy alone. Hidden size, temporal resolution, and depth jointly determine the AIE computation, recurrent update rate, resource use, and end-to-end latency. We develop analytical feasibility bounds that capture these constraints and eliminate configurations unable to keep pace with the input stream or meet the 1,000 ns latency target.

This search identifies two $H = 8$ operating points: LSTM-heavy, which retains the complete readout record, and LSTM-compact, which uses one mean I/Q pair per timestep.

Second, AIE offload is beneficial only when the input projection is large enough to justify the AIE-PL crossing. For LSTM-heavy, moving the wide input projection to the AIE array substantially reduces PL resource use and produces a deployable design. For LSTM-compact, the input projection is small, and PL-only execution is faster. Hardware placement must therefore be selected together with the temporal representation rather than applied uniformly across models.

Results. On Q3, LSTM-heavy achieves 96.59% accuracy at a 704 ns readout duration, corresponding to a 68.2% relative reduction in error over GMM and 43.6% over QubiCML, and 16.82% over MF-NN [23]. It matches the full-resolution FCN’s accuracy with $403\times$ fewer parameters and uses $2.2\times$, $3.9\times$, and $8.2\times$ fewer DSP58 blocks than QubiCML, a prior LSTM, and MF-NN, respectively. The AIE-PL implementation completes in 854 ns, compared with 1,082 ns for the pure-AIE mapping, and reduces the exposed inference tail by 60.3%. Relative to an all-PL implementation of the same model, it reduces DSP58 use by $31.1\times$ and eliminates BRAM use.

The same mapping is not optimal for every model. The PL-only LSTM-compact implementation completes in 814 ns, compared with 864 ns for its AIE-PL implementation, and achieves the highest mean accuracy across the eight qubits at 96.78%. A per-channel deployment uses LSTM-heavy for Q3 and LSTM-compact for the other seven qubits, outperforming the FCN in readout fidelity on every qubit. These results show that temporal resolution and hardware placement are both per-model design decisions.

Contributions. The main contributions of this paper are:

- We characterize readout traces from eight superconducting qubits and show that the value of temporal resolution is channel-dependent. In particular, we identify a qubit for which mid-readout relaxation causes integration to discard evidence that remains recoverable in the time-resolved record.
- We develop a hardware-aware recurrent design-space exploration that jointly considers temporal resolution, hidden size, model depth, input-stream cadence, and end-to-end latency. The resulting compact and full-resolution operating points allow classifier complexity to follow the needs of each qubit.
- We develop a heterogeneous recurrent implementation that maps independent LSTM input projections to AIE tiles and keeps the loop-carried state update in PL. This dependency-aware partition sustains the input cadence while substantially reducing PL resource use.
- We implement and evaluate the design on an AMD Versal VCK190, achieving FCN-level accuracy on the qubit most affected by relaxation within an 854 ns

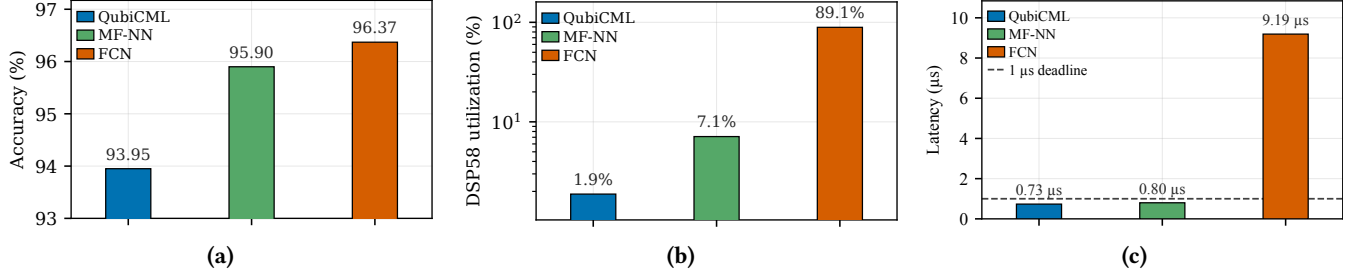


Figure 1. Comparison of single-qubit classifiers. (a) Accuracy on Q3, the qubit most affected by mid-readout relaxation. (b) DSP58 utilization. (c) Latency, including the 704 ns acquisition window, against the 1,000 ns accelerator target. No existing classifier is simultaneously accurate, efficient, and fast enough. FCN values are estimates.

accelerator latency, while a per-channel deployment provides high readout fidelity across all eight qubits. We confirm this latency on physical VCK190 hardware for the deployed LSTM-heavy AIE-PL mapping, measuring 860 ns end-to-end.

2 Background

2.1 Superconducting Qubit Readout

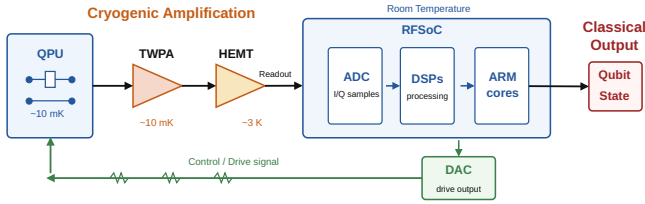


Figure 2. Schematic of a typical superconducting transmon qubit control and readout chain.

Figure 2 shows a typical readout pipeline in superconducting quantum platforms. Qubit state information is extracted via dispersive measurement: a microwave probe pulse traverses a resonator coupled to the qubit, acquiring state-dependent phase and amplitude shifts that are digitized into in-phase (I) and quadrature (Q) components [29, 33]. The resulting I/Q samples form a time series whose trajectory captures the temporal evolution of the readout signal, including transient state changes that may occur during measurement.

On the QubiC platform [33], a 2 GSPS sampling rate and finite measurement duration together produce a large volume of time-series data per shot. These traces are typically processed with matched filters or machine-learning classifiers on the control board or an external host, and this step is often the bottleneck: it is slow relative to the qubit coherence time and prone to misclassification. Readout accuracy depends on both the analog front end and this discriminator stage; this paper focuses on the latter, where the accuracy-latency trade-off we study is decided.

2.2 Temporal Readout Classification

Integration-based classifiers commonly reduce a readout record to a single point in the IQ plane and apply a statistical or feed-forward discriminator. Gaussian-mixture models [15] are attractive because they are simple and inexpensive, while feed-forward neural networks can learn more expressive decision boundaries. Both approaches, however, operate on the representation presented to them and cannot recover temporal information removed by integration.

A sequence-aware classifier instead processes the readout as a temporal trajectory. LSTMs [14] maintain a recurrent hidden state across timesteps and can therefore use the evolution of the measurement record to distinguish trajectories that have similar integrated IQ values. Prior work has demonstrated LSTM-based reconstruction of superconducting-qubit trajectories [19] and sequence-based qubit-state discrimination [17]. These results motivate temporal inference for readout, but do not imply that the full raw trajectory is necessary for every workload: temporal aggregation can trade input resolution for substantially lower model, hardware, and communication cost. Section 3 characterizes this trade-off across an eight-qubit workload before the hardware design is introduced.

3 The Case for Time-Resolved Readout

The core observation behind our design is that a single input representation does not serve all qubits. For a qubit such as Q1, conventional integration can reduce the full readout trace to two accumulated I/Q values while preserving the information needed for classification. For a qubit such as Q3, however, integration obscures when the state changes during measurement. Its classifier must retain more of the temporal record and consequently process a larger input. Thus, the temporal resolution required for qubit-state discrimination, and the hardware resources needed to process that input, are channel-dependent.

Figure 3 contrasts these two regimes. On Q1, the three prepared states remain well separated after integration, and

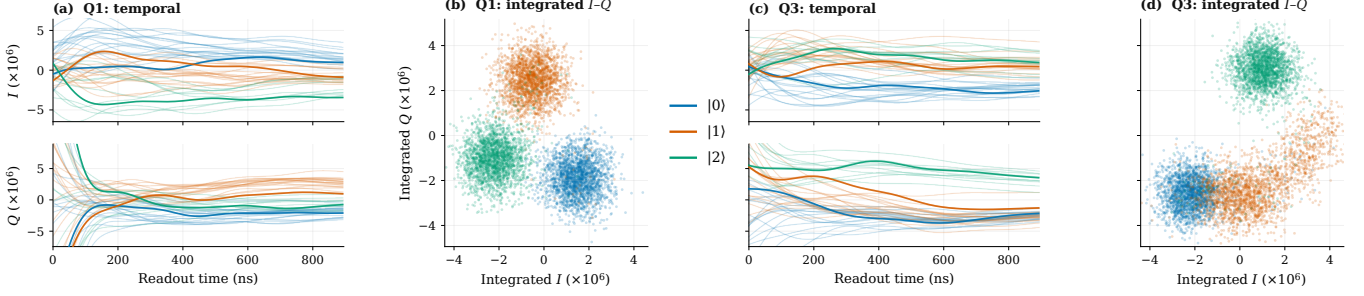


Figure 3. Readout trajectories for a representative well-behaved qubit (Q1) and the integration-limited qubit (Q3). (a),(c) Running I/Q trajectories; (b),(d) the corresponding integrated points. Q1 remains separable after integration. On Q3, a population of prepared $|1\rangle$ shots transitions toward $|0\rangle$ during readout, causing the integrated distributions to overlap. *Traces are bandpass-filtered for visual clarity only.*

retaining the complete temporal record provides little additional value. On Q3, a population of prepared $|1\rangle$ shots moves toward the $|0\rangle$ trajectory during readout, causing their integrated points to overlap with the $|0\rangle$ cluster. The early samples still contain evidence of the prepared state, but that evidence is lost when the complete record is reduced to one integrated I/Q point.

This comparison raises two questions. First, how much of the accuracy loss can be recovered by learning a better decision boundary over the same integrated input? Second, how much additional accuracy requires retaining temporal information?

We answer these questions by comparing classifiers along the input-resolution axis, from one integrated I/Q point to the complete readout trace. These comparisons make the case for selecting temporal resolution per channel and show how that choice changes the hardware workload. Section 6.2 later compares recurrent and feed-forward models after controlling the input representation more closely.

3.1 Experimental Setup and Comparison

We evaluate single-shot readout traces from eight fixed-frequency transmon qubits at a U.S. national laboratory, acquired with QubiC [33]. The dataset contains 25,000 shots per prepared state $|0\rangle$, $|1\rangle$, and $|2\rangle$. The ADC samples at 2 GSPS, so the 704 ns window used throughout the evaluation contains $S = 1408$ complex samples, or $2S = 2816$ interleaved I/Q values. The classifier consumes unfiltered post-DLO I/Q samples rescaled to the int16 range. We discard shots that fail heralded-state preparation [16] ($\sim 1\text{--}7\%$, by qubit) and report accuracy on the retained shots.

We use three classifiers to expose the accuracy-resolution tradeoff. GMM [15] and QubiCML [32] consume the same integrated I/Q point, or two real values, but use different decision functions. Their comparison shows what can be gained by learning a more expressive boundary without retaining additional temporal information. The FCN [20]

consumes the complete 2,816-value trace. We use it as a full-resolution accuracy reference, while noting that its millions of parameters and high hardware cost make it impractical for our target deployment. All classifiers use the training procedure described in Section 6.1.

3.2 Impact of Temporal Resolution

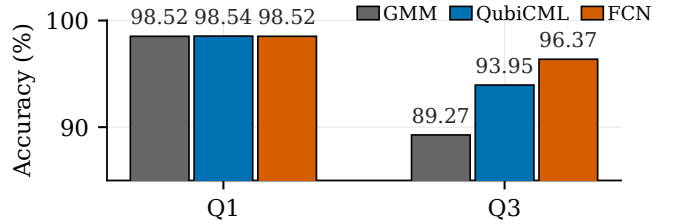


Figure 4. Readout accuracy on Q1 and Q3 at 704 ns. Retaining the complete temporal record provides little benefit on Q1 but substantially improves accuracy on Q3.

The accuracy results show the same two regimes (Figure 4). On Q1, classifier choice has little effect because integration preserves a separable representation. Q3 shows a different progression. At 704 ns, GMM reaches 89.27% accuracy. QubiCML improves this to 93.95% using a learned boundary over the same two integrated values. Thus, 4.68 points of the gap can be recovered without changing the input representation. The full-resolution FCN reference reaches 96.37%, indicating that another 2.42 points are available when the temporal record is retained. This gap between the integrated QubiCML and the full-resolution FCN shows that learning a better decision boundary alone is insufficient on Q3.

The readout-duration sweep provides a stronger test of this result. If integration only needed more signal, its accuracy should improve or eventually saturate as the window grows. On Q3, extending the window instead makes the integrated representation less reliable.

As shown in Figure 5, GMM accuracy is nearly flat through 512 ns at 92.53% and then falls to 88.99% by 768 ns. The longer

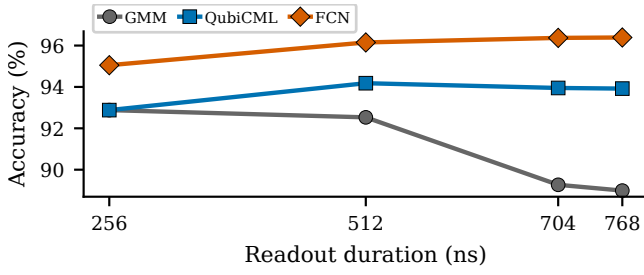


Figure 5. Accuracy versus readout duration on Q3. GMM declines as the window grows, QubiCML plateaus after 512 ns, and the full-resolution FCN continues to improve.

window makes the integrated statistic less discriminative because it mixes more of the later $|0\rangle$ -like signal into relaxing $|1\rangle$ shots. QubiCML recovers a better boundary in the integrated plane but plateaus once that representation has discarded the transition timing. The FCN continues to improve because its input retains both the early $|1\rangle$ evidence and the subsequent trajectory. Together with Figure 3, this shows that the problem is not simply insufficient signal or an inadequate decision boundary; on Q3, useful information is lost during integration.

Across the eight qubits, Q3 has the largest measured benefit from retaining the complete trace. The other seven channels do not exhibit the same continuing gain at full resolution (Table 1). The channels therefore require unequal temporal resolution: Q3 benefits from the complete trace, whereas the other seven can use a more compact representation.

3.3 Implications for Hardware-Algorithm Co-Design

The input representation should therefore be selected per channel rather than fixed across the device. A difficult channel such as Q3 should retain the complete temporal record, while other channels can use a compact representation without paying the cost of processing the full trace.

This choice directly changes the hardware workload: retaining more temporal detail increases the input dimension and the cost of processing each timestep. Hence, we need a hardware-aware approach that selects how much temporal information each channel retains while ensuring that the resulting classifier can sustain the input cadence and meet the latency and resource budgets.

4 Hardware-Aware Design-Space Exploration

Building on the observations in Section 3, our approach has two main components: selecting the temporal representation and recurrent-model configuration to match each channel, and selecting the hardware placement for the resulting workload. Both decisions must account for the input cadence, accelerator latency, and available hardware resources. This

section develops the analytical model and design-space exploration used to make these choices.

4.1 Target Platform and Execution Model

We explore these design choices for the AMD Versal VCK190, whose combination of AI Engines (AIEs) and programmable logic (PL) exposes the mapping tradeoff central to our approach. The device provides a 400-tile AIE array at 1.25 GHz connected to PL through PLIO/AXI4-Stream interfaces [4]. Each AIE tile provides a 512-bit vector datapath and 32 KB of local SRAM, supporting up to 32 MACs per cycle for int16 operands, while the PL fabric provides customizable datapaths. This asymmetry motivates the mapping used throughout the paper: the wide, independent $W_x x_t$ projections are mapped to AIE, while the small loop-carried recurrent update is implemented in PL. These platform characteristics determine where the LSTM computation can execute. Whether a mapping is feasible, however, also depends on how quickly readout samples arrive and when the classification result must be returned.

4.2 Streaming Interface and Design Objective

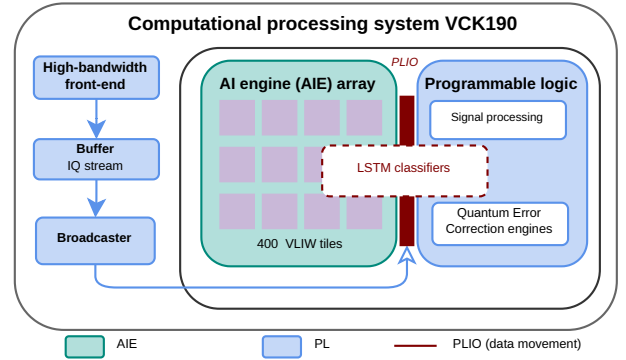


Figure 6. Target streaming architecture for real-time quantum-state classification on the AMD Versal VCK190. I/Q samples from the QubiC ZCU216 RFSoc enter the PL domain and are streamed to the AIE input projection engines through PLIO.

We show the target system in Figure 6. The quantum control system QubiC [33] digitizes the readout signal at 2 GSPS and streams complex I/Q samples toward the VCK190 via the high-bandwidth front-end, where PL forwards them to the AIE input-projection engines through PLIO as they arrive, without buffering the complete readout burst.

We report *accelerator latency* from the first I/Q sample leaving the converter interface to assertion of the valid class label. This excludes the analog front end, digital down-conversion, inter-board transfer, and controller return path, none of which is implemented in our setup. We use the 1,000 ns

target as an admission bound for the accelerator and reproduce the 2 GSPS input cadence offline with a paced replay buffer; no real-time cold-qubit experiments were performed.

Future RF-series Versal devices [5] integrate RF-ADCs/DACs with the same PL/AIE fabric, eliminating the inter-board communication our reported latency already excludes.

4.3 Recurrent Workload Decomposition

To determine how an LSTM maps to this streaming system, we first separate the computation that depends on the current input from the loop-carried state update. The LSTM forward pass [11, 14] at timestep t is

$$\begin{aligned} [\mathbf{i}_t \ \mathbf{f}_t \ \mathbf{g}_t \ \mathbf{o}_t]^\top &= [\sigma \ \sigma \ \tanh \ \sigma]^\top (W_x \mathbf{x}_t + W_h \mathbf{h}_{t-1} + \mathbf{b}), \\ \mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t, \quad \mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t). \end{aligned} \quad (1)$$

Here $\mathbf{i}_t$, $\mathbf{f}_t$, $\mathbf{g}_t$, and $\mathbf{o}_t$ are the input, forget, candidate, and output gates, respectively, while W_x and W_h stack their input and recurrent weight matrices. The equation exposes two computational terms central to our mapping. The input projection $W_x \mathbf{x}_t$ depends only on the current input, so the four gate-specific projections can execute in parallel as $\mathbf{x}_t$ arrives. The recurrent projection $W_h \mathbf{h}_{t-1}$ depends on the previous hidden state and therefore lies on the loop-carried dependency path.

For input dimension X and hidden dimension H , each gate's input projection requires HX MACs per timestep, while its recurrent projection requires H^2 . At the $H = 8$ operating point evaluated later, a raw 64 ns segment has $X = 256$, making the input projection $32\times$ larger than the recurrent projection. Reducing the same segment to one mean I/Q pair gives $X = 2$, making the recurrent projection four times larger than the input projection. The temporal representation therefore changes not only the input size, but also where the LSTM spends its work.

To simplify the loop-carried state update, we replace the standard sigmoid and tanh with piecewise-linear hardsigmoid and hardtanh nonlinearities that can be implemented as clamps:

$$\begin{aligned} \sigma_h(x) &= \text{clamp}\left(\frac{x}{4} + \frac{1}{2}, 0, 1\right), \\ \tanh_h(x) &= \text{clamp}(x, -1, 1). \end{aligned} \quad (2)$$

We also clamp the cell state to $[-1, 1]$, so the hidden-state update reduces to $\mathbf{h}_t = \mathbf{o}_t \odot \mathbf{c}_t$.

4.4 Analytical Hardware Model

The decomposition above allows us to determine whether a candidate LSTM can keep pace with the input stream. We model the AIE input-projection time, the PL recurrent-update time, and the exposed pipeline tail, then use these

terms to test both streaming feasibility and the 1,000 ns accelerator target. We use the following platform constants:

$$\begin{aligned} f_{\text{AIE}} &= 1.25 \text{ GHz}, & \text{MAC}_{\text{peak}} &= 32, \\ B_{\text{AIE}} &= 64 \text{ bit/cycle}. \end{aligned} \quad (3)$$

The 32-MAC/cycle peak follows from the 512-bit AIE vector datapath, while B_{AIE} follows from the two 32-bit input-stream ports [1, 4]. The nominal weight-load bandwidth is therefore eight times the activation-stream bandwidth.

4.4.1 Input-Projection Lower Bound. For one gate tile computing $\mathbf{z}_k = W_{ik} \mathbf{x}_t$, two independent constraints bound the projection time: compute requires at least $HX/\text{MAC}_{\text{peak}}$ AIE cycles, and input delivery requires at least X/B_{eff} cycles, where $B_{\text{eff}} = \min(B_{\text{AIE}}, B_{\text{link}})$ is the effective activation bandwidth in int16 values per AIE cycle, capped by either the native AIE path ($B_{\text{AIE}} = 4 \text{ int16/cycle}$) or a lower upstream link bandwidth limited by clock-domain crossing (CDC), B_{link} . Thus

$$T_{\text{Wx}}^{\text{floor}}(H, X) = \max\left(\frac{HX}{\text{MAC}_{\text{peak}}}, \frac{X}{B_{\text{eff}}}\right). \quad (4)$$

Because the four gate projections execute independently and in parallel, this single-gate bound also determines the AIE input projection cadence floor; it captures only fundamental compute and input-delivery costs, so implementation overheads can only increase the realized cadence.

The native AIE balance point is $H_{\text{AIE}}^* = \text{MAC}_{\text{peak}}/B_{\text{AIE}} = 32/4 = 8$: sustaining 32 int16 MACs/cycle requires 512 bits/cycle of weights, against a native input path of 64 bits/cycle. The system-level balance is $H_{\text{sys}}^* = \text{MAC}_{\text{peak}}/B_{\text{eff}}$. For $H < H_{\text{AIE}}^*$, a single gate projection is input-stream bound; in our mapping, $H = 4$ packs two independent gate projections per tile, using two AIE tiles for all four gates. For $H > H_{\text{AIE}}^*$, a single tile becomes compute-bound at the native interface; $H = 16$ therefore requires two tiles per gate to keep pace with the system stream. Thus $H_{\text{AIE}}^* = 8$ is an intrinsic tile property, while the realized system balance depends on the effective upstream bandwidth B_{eff} .

4.4.2 Streaming Feasibility. Let S denote the number of complex I/Q samples in the selected readout window and T the number of LSTM timesteps, so each timestep represents $B = S/T$ complex I/Q samples. For an ADC sampling at f_{ADC} , a timestep therefore arrives every $T_{\text{bin}} = B/f_{\text{ADC}}$; for our workload, $f_{\text{ADC}} = 2 \text{ GSPS}$, so $T_{\text{bin}} = 0.5B \text{ ns}$.

The AIE input projection is hidden by acquisition, and the PL recurrent update keeps pace with the stream, when

$$T_{\text{AIE}} T_{\text{Wx}}^{\text{floor}}(H, X) \leq T_{\text{bin}} \quad \text{and} \quad T_{\text{rec}}^{\text{step}}(H, N_L) \leq T_{\text{bin}}, \quad (5)$$

where $T_{\text{AIE}} = 0.8 \text{ ns}$ is the AIE clock period on the VCK190, N_L is the number of LSTM layers and $T_{\text{rec}}^{\text{step}}$ is the PL state-update time at the target 500 MHz clock.

When both stages sustain the input cadence, only the exposed tail is added to the acquisition time:

$$T_{\text{acc}}^{\text{est}}(H, T, N_L) = \frac{S}{f_{\text{ADC}}} + T_{\text{tail}}(H, T, N_L) \leq 1,000 \text{ ns}. \quad (6)$$

For the target architecture,

$$T_{\text{tail}}(H, T, N_L) = T_{\text{plio}} + T_{\text{rec}}^{\text{tail}}(H, T, N_L) + (N_L - 1)T_{\text{il}} + T_{\text{dense}}(N_L) + T_{\text{CDC}}^{\text{exposed}}, \quad (7)$$

where $T_{\text{CDC}}^{\text{exposed}}$ includes only CDC latency that remains exposed after the streaming stages have completed. For the deployed $H = 8$, one-layer datapath, the measured recurrent update is $T_{\text{rec}}^{\text{step}} = 31$ PL cycles = 62 ns, of which 13 cycles are overlapped with the input stream, $T_{\text{plio}} = 6$ ns, and $T_{\text{dense}} = 36$ ns. The 1,000 ns bound is an admission test: a candidate that exceeds it cannot meet the target even if excluded control-path stages consume no accelerator budget. Candidates that pass are implemented in hardware.

4.5 Pure-AIE Mapping Constraints

The input-projection balance does not by itself make the recurrent update efficient on AIE. A pure-AIE mapping must either localize the loop-carried state on a central tile or distribute it across tiles.

State localization. A central tile aggregates the four gate results and maintains $\mathbf{h}_t$ and $\mathbf{c}_t$ locally. For the implemented kernel, the architectural live state can be approximated by

$$N_{\text{live}}(H) = 2 \underbrace{\left\lceil \frac{4H}{8} \right\rceil}_{\text{bias}} + \underbrace{\left\lceil \frac{H}{4} \right\rceil}_{\mathbf{h}_t} + \underbrace{\left\lceil \frac{H}{4} \right\rceil}_{\mathbf{c}_t} + \underbrace{\left\lceil \frac{H}{8} \right\rceil}_{\mathbf{h}_{16}} + \underbrace{2}_{\text{wt. temp.}}. \quad (8)$$

Counts are in 128-bit vector registers, of which the AIE tile provides 16 [1]. At $H = 8$, the architectural state alone is approximated at 13 registers, with the weight temporaries adding 2 more for a predicted total of 15; the compiler assigns 14 of the 16 physical registers, closely matching this estimate. The pure-AIE kernel consequently spills to data memory: the compiler reports `__spill_DM_stack` and seven register-allocation moves, and the measured Π is 100 cycles versus a 64-cycle floor. At 1.25 GHz, this is 80 ns, exceeding the 64 ns input cadence. At $H = 4$, the same architectural state occupies only 7 registers.

State distribution. Distributing the recurrent update across AIE tiles reduces local register pressure but requires the previous hidden state to cross the interconnect at every timestep, making gate and state update kernels as well as scheduling a lot more complex. The communication therefore recurs T times for only H^2 recurrent MACs per gate per timestep. We did not implement this variant, so the comparison is analytical rather than measured. Pure-AIE mapping thus trades register pressure for repeated state movement,

motivating placement of the loop-carried state in PL while the independent input projections remain on AIE tiles.

4.6 Hardware-Constrained Design-Space Exploration

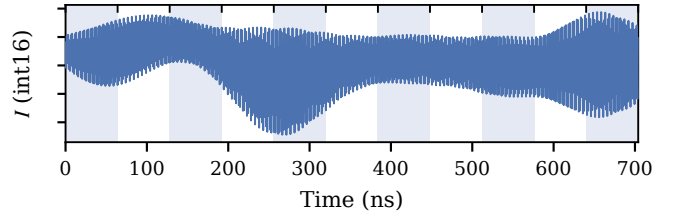


Figure 7. Raw Q3 trace (I component, int16-rescaled), with alternating shading marking successive $T=11$, $B=128$ -sample (64 ns) LSTM-heavy timesteps.

The preceding analysis yields three feasibility checks: the AIE input projection and PL recurrent update must each keep pace with the input stream, and the acquisition-plus-tail latency must remain within the 1,000 ns budget. We use these analytical tests to prune candidate models before training. We explore a refined search region over hidden dimension, temporal resolution, representation, and depth:

- Hidden dimension: $H \in \{4, 8, 16\}$.
- Temporal bin size: $B \in \{128, 256, 512\}$ complex I/Q samples per timestep.
- Temporal representation: $\mathcal{R} \in \{\text{raw}, \text{meanIQ}\}$, giving $X = 2B$ (raw) and $X = 2$ (meanIQ), respectively.
- Number of LSTM layers: $N_L \in \{1, 2\}$.

Recall S , the number of complex I/Q samples in the selected readout window (Sec. 4.4.2); the number of complete timesteps is $T = \lfloor S/B \rfloor$, with leftover samples discarded. For the primary 704 ns workload, the 2 GSPS input gives $S = 1408$ complex samples. The deployed configurations use $B = 128$, corresponding to a 64 ns timestep and $T = 11$. LSTM-heavy therefore consumes $X = 256$ real interleaved I/Q values per timestep, whereas LSTM-compact reduces the same 64 ns window to one mean I/Q pair, giving $X = 2$ while keeping the same 11-step sequence. Figure 7 visualizes this binning on a raw Q3 trace (I only; Q is binned identically).

We additionally verify that the resident gate weights fit within the 31 KB of tile-local memory available after reserving 1 KB for the stack: for a tile processing H_{tile} output rows, the resident weight footprint is $M_{\text{gate}}(H_{\text{tile}}, X, d) = H_{\text{tile}}Xd \leq 31,744$ bytes, where $d = 2$ bytes for the int16 weights used by the deployed models. All candidates in this refined search region satisfy this capacity bound, so local memory is a residency check rather than a binding pruning constraint.

The joint admission rule combining these four checks is therefore

$$\begin{aligned} \text{admit}(H, \mathcal{R}, T, N_L) \iff & T_{\text{AIE}} T_{W_X}^{\text{floor}}(H, X) \leq T_{\text{bin}} \\ & \wedge T_{\text{rec}}^{\text{step}}(H, N_L) \leq T_{\text{bin}} \\ & \wedge T_{\text{acc}}^{\text{est}}(H, T, N_L) \leq 1,000 \text{ ns} \\ & \wedge M_{\text{gate}}(H_{\text{tile}}, X, d) \leq 31,744 \text{ bytes} \end{aligned} \quad (9)$$

where $X = X(\mathcal{R}, B)$ and H_{tile} is determined by the gate-to-tile partitioning used for the candidate.

The resulting search exposes three regimes. With $B = 64$, the PL recurrent update is the limiting stage for $H = 8$ and cannot sustain the input cadence. At the deployed $B = 128$ point, both the AIE projection and recurrent update keep pace. For larger bins, the AIE input projection becomes the limiting stage even relative to the 100-cycle recurrent II of the spilled pure-AIE kernel; these candidates remain stream-feasible but provide lower accuracy than $B = 128$.

4.6.1 Search Outcome. The search identifies single-layer $H=8$ as the best balance between accuracy and hardware cost. A larger hidden dimension ($H=16$) doubles the AIE tiles required for input projection without improving Q3 accuracy; coarser temporal binning ($B=256, 512$) reduces accuracy while lowering projection cost; and a second recurrent layer remains cadence-feasible but adds a second recurrent engine and inter-layer relay cost with no accuracy benefit. Full ablation numbers are given in Appendix A. We therefore deploy single-layer $H=8$ designs: LSTM-compact ($X=2$) and LSTM-heavy ($X=256$).

5 Co-Designed LSTM Implementation

The hardware-aware design-space exploration identifies two operating points with $H = 8$: LSTM-heavy, which preserves the raw I/Q samples within each timestep, and LSTM-compact, which reduces each 64 ns timestep to one mean I/Q pair. Both use the same heterogeneous AIE-PL design; only the input-projection kernel changes.

The implementation partitions the LSTM at its data dependency boundary. The four independent input projections $z_k = W_{ik}x_t$ execute on four AIE tiles, while the loop-carried recurrent update and the final classifier remain in programmable logic.

5.1 AIE Input-Projection Engine

Each projection $z_k = W_{ik}x_t$ ($k \in \{i, f, g, o\}$) runs on its own AIE tile. For LSTM-heavy, $H = 8$ and $X = 256$ real interleaved int16 values are consumed per timestep. The analytical lower bound is 64 AIE cycles; the implemented kernel achieves 73 cycles, or 58.4 ns at 1.25 GHz, when driven at the native dual-stream peak of 4 int16 values/AIE cycle. The additional 9 cycles capture input delivery, data type conversion, and PLIO output setup omitted from the lower-bound model.

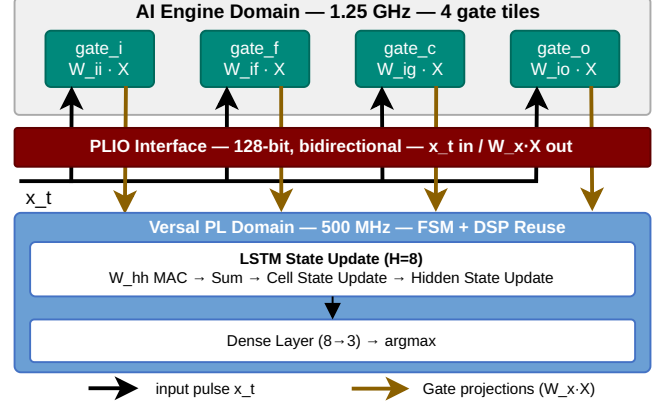


Figure 8. Architecture of the AIE-PL LSTM. Four AIE tiles compute the input projections $W_{ik}x_t$ in parallel; a PL state-update engine produces the three-class output.

The deployed 2 GSPS input arrives at 64 Gb/s, corresponding to one 256-value timestep every 64 ns. The 73-cycle kernel interval is therefore below the 64 ns arrival interval, so the input projection is stream-paced rather than compute-paced. FIFOs absorb the rate mismatch between the peak AIE throughput and the converter stream.

LSTM-compact’s AIE-PL mapping shares this datapath and recurrent backend, but its input-projection kernel consumes one mean I/Q pair ($X = 2$) per 64 ns timestep. A single 32-bit PLIO stream is sufficient for each tile.

5.2 PL Recurrent-State Engine

Gate results cross the PLIO streaming interface, are buffered across the clock-domain crossing, and are reassembled by a PL-side merge stage before the state update. The recurrent update and dense classifier are implemented as a finite-state controller coupled to a time-multiplexed DSP58 MAC datapath in PL at 500 MHz. The engine computes $W_{hk}h_{t-1}$, updates c_t and h_t , and the dense classifier head, while reusing DSP58 resources across these operations. Hardsigmoid and hardtanh are implemented in LUT logic.

At $H = 8$, the recurrent projection takes 13 PL cycles (26 ns), followed by 18 cycles (36 ns) for the state update, giving a 31-cycle (62 ns) steady-state update. The recurrent projection overlaps the streamed AIE input projection; hence the 18-cycle portion is the exposed PL critical path.

5.3 Streaming Overlap and Critical Path

AIE projection and PL recurrent processing execute concurrently as input streams through the pipeline (Fig. 9), so only the 18-cycle PL update (Section 5.2) is exposed at the state-update boundary for the deployed $H=8, T=11$ design.

Hardware-integration simulation reproduces the 64 ns gate cadence for timesteps 1–11. The final input leaves an exposed drain before the last gate result reaches PL; this drain

and the final PL state-update/output work form the latency tail. The deployed LSTM-heavy implementation therefore has 704 ns of acquisition plus a 150 ns exposed tail, for the 854 ns accelerator latency reported in Section 6. The corresponding pure-AIE implementation exposes a 378 ns tail.

The hybrid pipeline is deterministic: all datapaths are statically scheduled and contain no data-dependent control flow, so repeated shots produce the same accelerator latency.

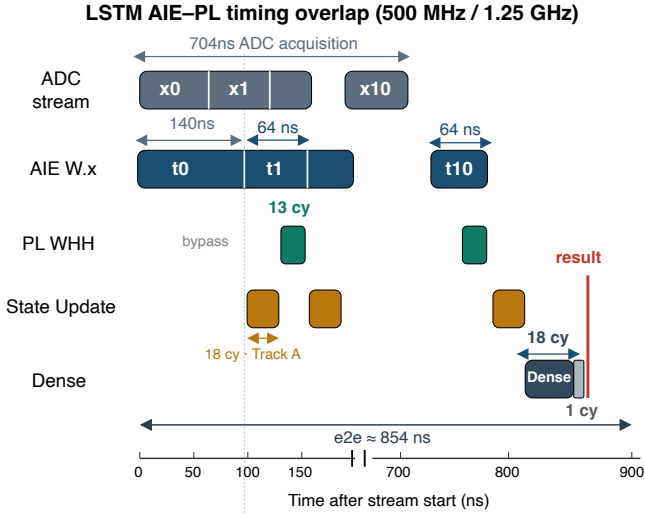


Figure 9. Streaming datapath overlap for the deployed $H = 8$, $T = 11$ implementation. AIE input projection overlaps PL recurrent processing at the 64 ns input cadence; the 31-cycle PL update has an 18-cycle exposed state-update portion.

5.4 Pure-AIE Baseline

For a fair comparison with the AIE-PL version, we instantiate a pure-AIE $H = 8$ design using the same four input-projection tiles and a central recurrent-state tile with the same hardtanh and hardsigmoid. The register-pressure-induced 100-cycle initiation interval is analyzed and measured in Section 4.5; its end-to-end latency is reported in Section 6. No PL resources are attributed to this baseline.

6 Evaluation

We evaluate our hardware-aware LSTM design across classification quality, hardware cost, and real-time latency. We focus on four questions: (1) does the selected recurrent model retain the temporal-information benefit on mid-readout state transition workloads; (2) how does its accuracy compare with the feed-forward and other machine learning-based classifiers; (3) what resource cost does the heterogeneous AIE-PL implementation require; and (4) how does the measured execution compare with the pure-AIE baseline and the sub-microsecond latency objective?

6.1 Experimental Setup

Model training. All learned models are implemented in PyTorch [25] and trained for 100 epochs with Adam optimizer [18], an initial learning rate of 10^{-4} , batch size 128, and exponential decay every 10 epochs. All classifiers use the same stratified 80/20 split and random seed. Our LSTMs use the hardware-friendly hardsigmoid and hardtanh nonlinearities described in Section 4.3 for both training and inference. Because the prior LSTM [17] has no hardware implementation, we synthesize its architecture with our own hard activations, trained with its reported GMM-confidence-weighted loss, for the resource and latency figures in Section 6.5; accuracy figures instead use its original soft activations. It also does not specify a bin width, so we use 64 ns bins for fair comparison with our recurrent models.

Hardware evaluation. We use aiesimulator [2] for AIE kernel timing and functional behavior, Verilator [28] for the PL datapath and AIE-PL integration checks, and QEMU [7] with the v++ [2] system link for end-to-end system execution and latency measurement. Synthesis and implementation use Vivado 2025.2 at a 500 MHz constraint for a fair latency comparison; resource utilization and WNS come from post-implementation reports, with post-synthesis figures flagged where used. The PL logic and all LSTM variants, including the prior LSTM [17], are written in Chisel [6] for one parameterized design; QubiCML [32] and MF-NN [23], which reduces each trace to nine matched-filter likelihood scores before a small feed-forward network refines the classification, are implemented with hls4ml [27] and hls [3]. All baselines are our own re-implementations at their published configurations, thus differences from published numbers are possible.

6.2 Impact on Readout Accuracy

Table 1. Single-shot readout accuracy (%) at 704 ns ($T=11$, FP32 evaluation). GMM^a [15], QubiCML [32], MF-NN [23], FCN^a [20], LSTM [17]; LSTM-h/LSTM-c denote LSTM-heavy/LSTM-compact. ^aaccuracy-only references, not synthesized. Full results with binomial standard errors are provided in Appendix E.

Qubit	GMM	QubiCML	MF-NN	FCN	LSTM	LSTM-h	LSTM-c
Q0	97.70	97.77	98.00	98.08	98.18	97.96	98.15
Q1	98.52	98.54	98.52	98.52	98.75	98.26	98.74
Q2	97.52	97.67	97.67	97.59	97.88	97.41	97.89
Q3	89.27	93.95	95.90	96.37	95.87	96.59	96.11
Q4	94.72	94.39	95.59	95.22	95.67	94.96	95.74
Q5	97.85	97.89	97.81	97.98	98.20	97.93	98.25
Q6	93.01	93.62	92.68	93.75	94.15	93.92	94.27
Q7	94.71	94.70	94.63	94.91	95.15	94.79	95.10
$\mu \pm \sigma$	95.41	96.07	96.35	96.55	96.73	96.48	96.78

Q3 is the qubit most affected by mid-readout relaxation (Section 3), so we use it as our central comparison case. GMM

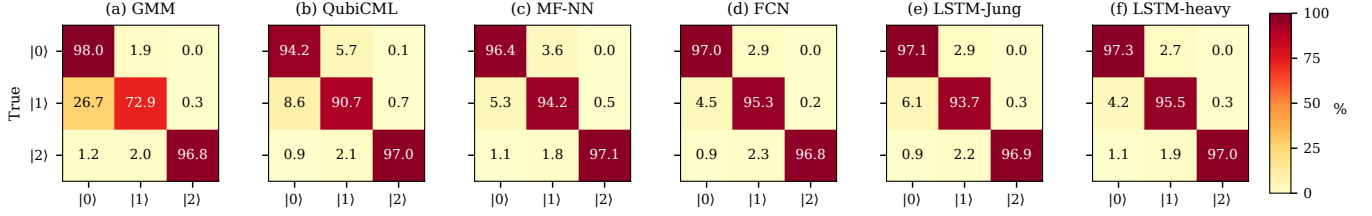


Figure 10. Row-normalized confusion matrices for Q3 at 704 ns. Cell values show the percentage of samples in each true class assigned to each predicted class. The dominant GMM error is $|1\rangle \rightarrow |0\rangle$ misclassification, which is substantially reduced by the full-resolution FCN and recurrent models.

and QubiCML, operating on an integrated point, reach 89.27% and 93.95%, respectively, while every structure-preserving model exceeds 95.8%; LSTM-heavy is highest at 96.59%, a 68.2% relative reduction in error over GMM, 43.6% over QubiCML, 16.82% over MF-NN and 6.06% over the FCN, though not significant after correction ($p = 0.06$).

Figure 10 shows why Q3 differs: GMM misassigns roughly a quarter of prepared $|1\rangle$ shots to $|0\rangle$, an error every structure-preserving model cuts to about 4%, while the $|2\rangle$ row remains largely unchanged. The recovered population is consistent with the mid-readout relaxation events in Section 3.

Across eight qubits, LSTM-compact achieves the highest mean accuracy (96.78%), a 29.9% relative reduction in error over GMM and 6.7% over the closest baseline, FCN, with QubiCML and MF-NN in between (18.1% and 11.8%), despite far fewer parameters than the FCN.

In addition to pooled accuracy, we report readout fidelity, a stricter, class-balanced metric: the geometric mean of per-state classification accuracy, $F = (P(0|0) \cdot P(1|1) \cdot P(2|2))^{1/3}$, computed from the row-normalized confusion matrix on the same frozen eval split. A per-channel deployment (LSTM-compact everywhere except Q3, using LSTM-heavy) outperforms the FCN in readout fidelity on every qubit (Table 2), by 0.07 to 0.54 points (0.29 on average).

Table 2. Deployed per-channel readout fidelity (%) vs. the FCN reference [20], at 704 ns. Deployed uses LSTM-compact on all channels except Q3, where LSTM-heavy is used.

	Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7
FCN*	98.07	98.52	97.59	96.36	95.19	97.96	93.64	94.83
Deployed	98.14	98.74	97.89	96.59	95.71	98.24	94.19	95.01

6.3 Q3 Accuracy–Resource Comparison

Figure 11 compares accuracy against DSP58 usage on Q3. DSP58 is the binding resource for every classifier evaluated, so this comparison determines hardware feasibility rather than latency alone. The resource gap is largest here: at 96.59% on Q3, LSTM-heavy needs $3.9\times$ fewer DSP58 blocks than the prior LSTM (95.87%), $8.2\times$ fewer than MF-NN (95.90%),

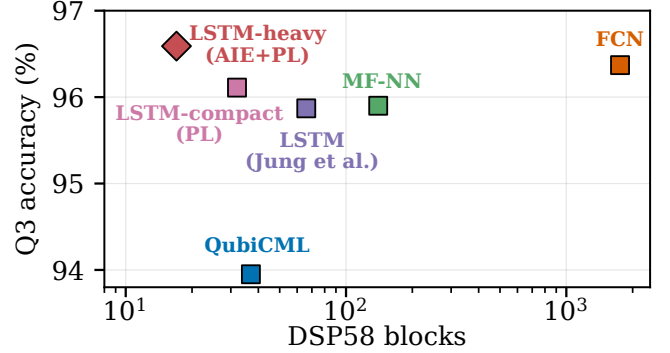


Figure 11. Accuracy on Q3 against DSP58 usage (log scale) for each classifier’s best deployable mapping.

and $2.2\times$ fewer than QubiCML. LSTM-compact trails LSTM-heavy by only 0.48 points (96.11%) while using the fewest DSP58 blocks of any classifier evaluated, including QubiCML. LSTM-heavy matches FCN’s 96.37%; LSTM-compact comes within 0.26 points, at a fraction of FCN’s estimated 89.1% DSP58 budget, consistent with prior reports that this architecture scales poorly on FPGA fabric [22]; full synthesis exhausted our host memory during elaboration, so we report an hls4ml-based analytical estimate using a per-layer ReuseFactor equal to each layer’s fan-in (2816/1000/500/250) instead; latency is estimated the same way, extrapolated from smaller synthesized networks to 9.19 μ s.

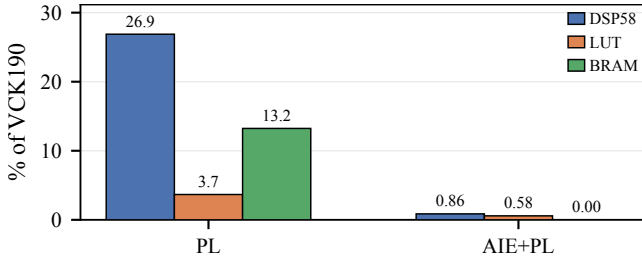
DSP58 is not the only axis worth checking: Table 3 reports LUT and BRAM usage alongside DSP58 and AI Engine tile count for every design. LSTM-heavy’s DSP58 and BRAM advantage does not extend to LUT, where it uses more than either baseline; we report the full resource profile rather than DSP58 alone, since Section 6.4 details how that DSP58 advantage is achieved.

6.4 PL vs. AIE+PL Resource Comparison

The AIE–PL LSTM-heavy mapping offloads its wide 8×256 input projection to 4 out of 400 AIE tiles, cutting DSP58 usage from 26.9% to 0.86% of the device and LUT usage from 3.7% to 0.58% (Figure 12), a $31.1\times$ reduction in DSP58 blocks and $6.4\times$ in LUTs, while eliminating all 128 BRAMs.

Table 3. Resource usage on the VCK190 for different classifiers in this paper.

Design	DSP58	LUT	BRAM	AIE tiles
QubiCML	37 (1.9%)	0.18%	0.0%	—
MF-NN	140 (7.1%)	0.42%	7.5%	—
FCN	1,753 (89.1%)	—	—	—
LSTM-compact	32 (1.6%)	0.52%	0.8%	0/400
LSTM-heavy	17 (0.86%)	0.58%	0.0%	4/400

**Figure 12.** LSTM-heavy resource use on the VCK190, as a fraction of device DSP58 and LUT capacity, for the PL-only and the AIE+PL implementation.

This is what lets the AIE offload bring a full-resolution recurrent model under budget on this device (Section 6.3). LSTM-compact already fits in PL at 32 DSP58 blocks, so we retain its PL-only mapping rather than offloading to AIE: with only $X=2$ input features per timestep, the projection is too small for the AIE round trip to pay for itself.

6.5 End-to-End Latency

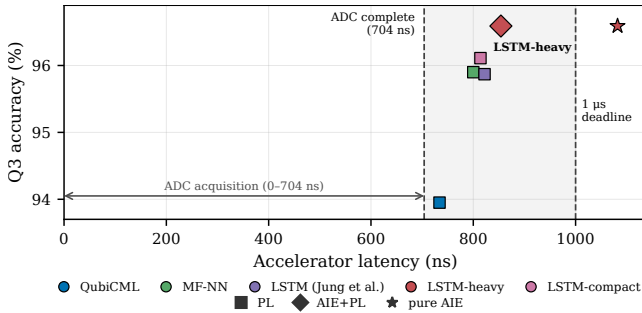
**Figure 13.** Measured accelerator latency under a 704-ns acquisition stream.

Figure 13 plots accelerator latency against mean accuracy for every evaluated classifier and mapping, relative to the 704 ns acquisition window and the 1,000 ns deadline. Table 4 compares the evaluated classifiers under the same 2 GSPS replay conditions (accelerator latency as defined in Section 4.2). As detailed in Section 5.3, the AIE-PL and pure-AIE mappings reach the same 96.59% accuracy on Q3

since both implement identical trained weights, but only the AIE-PL mapping’s shorter exposed tail meets the 1,000 ns deadline. On physical VCK190 hardware, this AIE-PL mapping measures 860 ns end-to-end for a representative single shot—within 0.7% of the 854 ns simulated figure above.

Table 4. Single-shot accelerator latency at the 704 ns operating point. Tail is accelerator latency minus the acquisition window; Δ is the tail reduction of each model’s deployable mapping over its alternative. [†]Measured on physical VCK190 hardware: 860 ns. [‡]Reference only.

Model	Mapping	Accel. (ns)	Tail (ns)	Δ (%)
QubiCML [32]	PL	734	30	—
MF-NN [23]	PL	800	96	—
LSTM [17]	PL	822	118	—
<i>This work</i>				
LSTM-compact	PL only	814	110	31.25
	AIE+PL	864	160	
LSTM-heavy	AIE+PL	854[†]	150	60.32
	Pure AIE	1082	378	
	PL only [‡]	792	88	

An all-PL LSTM-heavy implementation is nominally faster at 792 ns but misses timing closure at 500 MHz by a narrow margin ($WNS = -0.005$ ns); we use it only as the all-PL resource-comparison reference, not as a deployable operating point. For LSTM-compact, PL-only completes in 814 ns versus 864 ns for AIE+PL: with only $X=2$ input features per timestep, the AIE offload saves too little projection work to amortize the fixed AIE-PL crossing.

7 Related Work

Feed-forward inference for qubit readout. Vora et al. [32] demonstrated low-latency FPGA inference for qubit/qutrit discrimination—the QubiCML baseline we reimplement and evaluate in Section 6.3—while Butko et al. [8] extended feed-forward inference to the Versal AI Engine. Maurya et al. [22] reduced classifier complexity through matched filters and a small neural network to fit FPGA resource constraints, and Mude et al. [23] extended this approach to three-level readout. Concurrent work by Otting et al. [24] targets the same failure mode with a Mamba-based sequence model, reducing parameter count by 49.6% over the prior state of the art, but evaluates entirely in software. None of these establish a hardware mapping for a recurrent classifier’s loop-carried state; Published latencies across this body of work are also not directly comparable, since acquisition windows differ, so we reimplement selected baselines under the same 704 ns replay and measurement definition.

Recurrent inference on spatial accelerators. Prior AIE mappings such as CHARM [34] and MaxEVA [30] optimize

dense matrix multiplication for throughput, whereas Sapkas et al. [26] study latency-constrained GRU inference on Versal and keep recurrent matrix-vector operations AIE-resident, proposing a AIE-PL variant for faster result aggregation. Our partition is complementary: the independent input projections execute on AIE tiles, while the loop-carried recurrent state remains in PL—the reverse of their assignment.

Hardware-aware recurrent model selection. Hardware-aware NAS methods such as ProxylessNAS[9] and MnasNet[31] use learned or measured cost models to navigate large architecture spaces. Recurrent FPGA accelerators such as ESE[12] and ML-to-FPGA toolkits like hls4ml[10, 27] instead target fixed architectures, trading throughput for resource use through compression or reuse-factor scheduling.

8 Conclusion

In this paper, we show that recurrent inference resists spatial acceleration at small hidden sizes because register pressure, not arithmetic throughput, becomes the bottleneck—our $H=8$ pure-AIE kernel spills to data memory and settles at a 100-cycle initiation interval against a 64-cycle floor—and we resolve this by placing the loop-carried state in PL and the four independent input projections on AIE tiles. This partition cuts LSTM-heavy’s exposed tail from 378 to 150 ns and its accelerator latency from 1,082 to 854 ns (860 ns measured on VCK190), inside the 1,000 ns budget, while reducing PL DSP58 usage by 96.8% and LUT usage by 84.3% and eliminating BRAM use, using 4/400 AI Engine tiles. Against prior baselines, LSTM-heavy uses 2.2–8.2× fewer DSP58 blocks than QubiCML, the prior LSTM, and MF-NN, and matches a full-resolution FCN’s accuracy on Q3 (96.59% vs. 96.37%) with roughly 403× fewer parameters; LSTM-compact outperforms QubiCML and MF-NN in accuracy on every qubit. Results are otherwise obtained offline via simulation, using a paced replay buffer, with no real-time cold-qubit experiments; integrating the accelerator into a live control loop and extending this partition to multiple simultaneous channels, where AIE tiles and PLIO interfaces become shared resources, are natural next steps.

Acknowledgments

We would like to thank Mark Rollins, Jose Manuel Monsalve Diaz and Paul Hartke (Advanced Micro Devices, Inc.) for discussions on the implementation and help with debugging.

Claude (Anthropic) was used to assist with editing, condensing, and revising portions of this manuscript’s text.

References

- [1] Advanced Micro Devices, Inc. *AI Engine Kernel and Graph Programming Guide (UG1079)*. AMD, 2025. Version 2025.2.
- [2] Advanced Micro Devices, Inc. *AI Engine Tools and Flows User Guide (UG1076)*. AMD, 2025. Version 2025.2.
- [3] Advanced Micro Devices, Inc. *Vitis High-Level Synthesis User Guide*. AMD, 2025. Version v2025.2.
- [4] Advanced Micro Devices, Inc. *Versal Architecture and Product Data Sheet: Overview (DS950)*. AMD, June 2026. v2.10.
- [5] AMD. Amd versal rf series. <https://www.amd.com/en/products/adaptive-socs-and-fpgas/versal/rf-series.html>. Accessed: Aug. 10, 2026.
- [6] Jonathan Bachrach, Huy Vo, Brian Richards, Yunsup Lee, Andrew Waterman, Rimas Avižienis, John Wawrzynek, and Krste Asanović. Chisel: constructing hardware in a scala embedded language. In *Proceedings of the 49th annual design automation conference*, pages 1216–1225, 2012.
- [7] Fabrice Bellard. Qemu, a fast and portable dynamic translator. In *Proceedings of the Annual Conference on USENIX Annual Technical Conference, ATEC '05*, page 41, USA, 2005. USENIX Association.
- [8] Anastasiia Butko, Artem Marisov, David I Santiago, and Irfan Siddiqi. Quantum state discrimination enhanced by FPGA-based AI Engine technology. *arXiv preprint arXiv:2601.08213*, 2026.
- [9] Han Cai, Ligeng Zhu, and Song Han. ProxylessNAS: Direct neural architecture search on target task and hardware. In *International Conference on Learning Representations*, 2019.
- [10] Javier Duarte, Song Han, Philip Harris, Sergo Jindariani, Edward Kreinar, Benjamin Kreis, Jennifer Ngadiuba, Maurizio Pierini, Ryan Rivera, Nhan Tran, and Zhenbin Wu. Fast inference of deep neural networks in FPGAs for particle physics. *Journal of Instrumentation*, 13(07):P07027, 2018.
- [11] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471, 2000.
- [12] Song Han, Junlong Kang, Huizi Mao, Yiming Hu, Xin Li, Yubin Li, Dongliang Xie, Hong Luo, Song Yao, Yu Wang, Huazhong Yang, and William J Dally. ESE: Efficient speech recognition engine with sparse LSTM on FPGA. In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 75–84. ACM, 2017.
- [13] S. Hazra, W. Dai, T. Connolly, P. D. Kurilovich, Z. Wang, L. Frunzio, and M. H. Devoret. Benchmarking the readout of a superconducting qubit for repeated measurements. *Phys. Rev. Lett.*, 134:100601, Mar 2025.
- [14] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.
- [15] Evan Jeffrey, Daniel Sank, J Y Mutus, T C White, Julian Kelly, R Barends, Y Chen, Z Chen, B Chiaro, A Dunsworth, A Megrant, P J J O’Malley, C Neill, P Roushan, A Vainsencher, J Wenner, A N Cleland, and John M Martinis. Fast accurate state measurement with superconducting qubits. *Physical Review Letters*, 112(19):190504, 2014.
- [16] J. E. Johnson, C. Macklin, D. H. Slichter, R. Vijay, E. B. Weingarten, John Clarke, and I. Siddiqi. Heralded state preparation in a superconducting qubit. *Phys. Rev. Lett.*, 109:050506, Aug 2012.
- [17] Samuel Jung, Neel Vora, Akel Hashim, Yilun Xu, and Gang Huang. Time-series based quantum state discrimination. *arXiv preprint arXiv:2601.19057*, 2026.
- [18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [19] G Koolstra, N Stevenson, S Barzili, L Burns, K Siva, S Greenfield, W Livingston, A Hashim, R K Naik, J M Kreikebaum, K P O’Brien, D I Santiago, J Dressel, and I Siddiqi. Monitoring fast superconducting qubit dynamics using a neural network. *Physical Review X*, 12(3):031017, 2022.
- [20] Benjamin Lienhard, Antti Vepsäläinen, Luke C.G. Govia, Cole R. Hoffer, Jack Y. Qiu, Diego Ristè, Matthew Ware, David Kim, Roni Winik, Alexander Melville, Bethany Niedzielski, Jonilyn Yoder, Guilhem J. Ribeill, Thomas A. Ohki, Hari K. Krovi, Terry P. Orlando, Simon Gustavsson, and William D. Oliver. Deep-neural-network discrimination of multiplexed superconducting-qubit states. *Phys. Rev. Appl.*, 17:014024, Jan 2022.

- [21] Piero Luchi, Paolo E Trevisanuto, Alessandro Roggero, Jonathan L DuBois, Yaniv J Rosen, Francesco Turro, Valentina Amitrano, and Francesco Pederiva. Enhancing qubit readout with autoencoders. *Physical Review Applied*, 20(1):014045, 2023.
- [22] Satvik Maurya, Chaithanya Naik Mude, William D Oliver, Benjamin Lienhard, and Swamit Tannu. Scaling qubit readout with hardware efficient machine learning architectures. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, pages 1–13, 2023.
- [23] Chaithanya Naik Mude, Satvik Maurya, Benjamin Lienhard, and Swamit Tannu. Efficient and scalable architectures for multi-level superconducting qubit readout. In *2025 62nd ACM/IEEE Design Automation Conference (DAC)*, pages 1–7. IEEE, 2025.
- [24] Luca Otting, Xiaorang Guo, Emmanouil Giortamis, Benjamin Lienhard, Pramod Bhatotia, and Martin Schulz. Multi-stage mamba-based architecture for fast and scalable superconducting qubit readout. *arXiv preprint arXiv:2607.11668*, 2026.
- [25] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.
- [26] M Sapkas, A Triossi, and M Zanetti. A latency-constrained, gated recurrent unit (GRU) implementation in the Versal AI Engine. *arXiv preprint arXiv:2511.15626*, 2025.
- [27] Jan-Frederik Schulte, Benjamin Ramhorst, Chang Sun, Jovan Mitrevski, Nicolò Ghielmetti, Enrico Lupi, Dimitrios Danopoulos, Vladimir Lončar, Javier Duarte, David Burnette, Lauri Laatu, Stylianos Tzelepis, Konstantinos Axiotis, Quentin Berthet, Haoyan Wang, Paul White, Suleyman Demirsoy, Marco Colombo, Thea Klæboe Aarrestad, Sioni Summers, Maurizio Pierini, Giuseppe Di Guglielmo, Jennifer Ngadiuba, Javier Campos, Benjamin Hawks, Abhijith Gandrakota, Farah Fahim, Nhan Tran, George A Constantinides, Zhiqiang Que, Wayne Luk, Alexander Tapper, Duc Hoang, Noah Paladino, Philip C. Harris, Bo-Cheng Lai, Manuel Valentin, Ryan Forelli, Seda Oğrenci, Lino Gerlach, Rian Brooks Flynn, Mia Liu, Daniel Diaz, Elham E. Koda, Melissa Quinlan, Russell Marroquin Solares, Santosh Parajuli, Mark S. Neubauer, Christian Herwig, Ho Fung Tsoi, Dylan Rankin, Shih-Chieh Hsu, and Scott Hauck. hls4ml: A flexible, open source platform for deep learning acceleration on reconfigurable hardware. *ACM Trans. Reconfigurable Technol. Syst.*, 19(2), May 2026.
- [28] Wilson Snyder, Paul Wasson, Duane Galbi, Geza Lore, et al. Verilator: Open-source systemverilog simulator and lint system. <https://verilator.org>, 2026. Available from Veripool.
- [29] Leandro Stefanazzi, Kenneth Treptow, Neal Wilcer, Chris Stoughton, Collin Bradford, Sho Uemura, Silvia Zorzetti, Sara Sussman, Salvatore Montella, Shefali Saxena, Horacio Arnaldi, Ankur Agrawal, Helin Zhang, Chunyang Ding, Andrew Houck, David I Schuster, and Gustavo Cancelo. The QICK (quantum instrumentation control kit): Readout and control for qubits and detectors. *Review of Scientific Instruments*, 93(4), 2022.
- [30] Endri Taka, Aman Arora, Kai-Chiang Wu, and Diana Marculescu. Maxeva: Maximizing the efficiency of matrix multiplication on versal ai engine. *arXiv preprint arXiv:2311.04980*, 2023.
- [31] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. MnasNet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2820–2828, 2019.
- [32] Neel Vora, Yilun Xu, Akel Hashim, Neelay Fruitwala, Nam Nguyen, Noah Goss, Jan Balewski, Birgitta Whaley, Irfan Siddiqi, Vp Nguyen, and Gang Huang. ML-enabled fpga framework for fast quantum state

discrimination in mid-circuit measurement regimes. In *Proceedings of the Great Lakes Symposium on VLSI 2026*, GLSVLSI '26, page 836–842, New York, NY, USA, 2026. Association for Computing Machinery.

- [33] Yilun Xu, Gang Huang, Neelay Fruitwala, Abhi Rajagopala, Ravi K Naik, Kasra Nowrouzi, David I Santiago, and Irfan Siddiqi. QubiC 2.0: An extensible open-source qubit control system capable of mid-circuit measurement and feed-forward. *arXiv preprint arXiv:2309.10333*, 2023.
- [34] Jinming Zhuang, Jason Lau, Hanchen Ye, Zhuoping Yang, Yubo Du, Jack Lo, Kristof Denolf, Stephen Neuendorffer, Alex Jones, Jingtong Hu, Deming Chen, Jason Cong, and Peipei Zhou. Charm: Composing heterogeneous accelerators for matrix multiply on versal acap architecture. In *Proceedings of the 2023 ACM/SIGDA International Symposium on Field Programmable Gate Arrays, FPGA '23*, page 153–164, New York, NY, USA, 2023. Association for Computing Machinery.

A Hardware-Aware Search Outcome

Table 5 shows representative outcomes at the 704 ns operating point: $H=16$ ties $H=8$ at 96.59% while doubling AIE tiles to eight; wider timesteps ($B=256, 512$, which repartition the raw sequence into fewer, larger steps without discarding samples) lower projection cost but cost 0.04–0.17 accuracy points; a second recurrent layer adds 576 parameters for no accuracy gain (96.58% vs. 96.59%); and a narrower timestep ($B=64$) is rejected outright, as the PL recurrent update cannot keep pace with the resulting cadence. We therefore deploy only the two highlighted configurations.

Table 5. Representative outcomes of the hardware feasibility tests at the 704 ns operating point. B denotes complex I/Q samples per timestep and X denotes real interleaved I/Q values presented to the LSTM at each timestep.

H	B	X	T	N_L	Q3 Acc.	Outcome
8	128	2	11	1	96.11%	LSTM-compact
8	128	256	11	1	96.59%	LSTM-heavy
8	256	512	5	1	96.55%	admitted; not deployed
8	512	1024	2	1	96.42%	admitted; not deployed
8	128	256	11	2	96.58%	admitted; not deployed
16	128	256	11	1	96.59%	admitted with 8 AIE tiles; not deployed
8	64	128	22	1	–	rejected: PL recurrent cadence

B Ablation Study

B.1 Impact of Input Quantization

The results throughout this paper (e.g., Table 1) evaluate FP32 models on I/Q samples already rescaled to the int16 range (Section 3), matching QubiC’s native fixed-point ADC output. Unlike Section B.3, which quantizes the model itself, here the model stays FP32 and only the input representation changes. Table 6 compares FP32 and int16 inputs for LSTM-heavy and a separately trained FCN reference.

Table 6. Single-shot accuracy (%) at 704 ns (raw). Δ = int16 – fp32 (percentage points). Mean is across 8 qubits.

qubit	LSTM-heavy			FCN		
	fp32	int16	Δ	fp32	int16	Δ
Q0	97.97	97.97	+0.00	98.10	98.09	-0.01
Q1	98.26	98.25	-0.01	98.51	98.52	+0.01
Q2	97.41	97.31	-0.10	97.61	97.61	+0.00
Q3	96.59	96.60	+0.01	96.35	96.36	+0.01
Q4	94.97	95.07	+0.10	95.31	95.31	+0.00
Q5	97.95	97.75	-0.20	97.97	97.98	+0.01
Q6	93.91	93.94	+0.03	93.77	93.78	+0.01
Q7	94.83	94.68	-0.15	94.91	94.91	-0.01
μ	96.49	96.45	-0.04	96.57	96.57	+0.00

The per-qubit accuracy change stays within 0.20 percentage points for either model, and the mean shifts by at most 0.04 points; minor differences from Table 1’s numbers for either model reflect ordinary training variance across separate runs, not the input change under test.

B.2 Hard Activation versus Soft Activation

The mean difference is +0.03 points for LSTM-heavy and -0.04 for LSTM-compact, an order of magnitude below the mean accuracy differences the paper’s conclusions rest on, so the hardware-friendly activations are free in accuracy terms on average. Q4 is the largest single-qubit exception (+0.29 for LSTM-heavy). The LSTM-heavy/LSTM-compact comparisons elsewhere in this paper are therefore unaffected by activation choice, since both models are trained and evaluated with the same hard activations throughout.

Table 7. Activation ablation at 704 ns ($T=11$). Hard activations (piecewise-linear sigmoid and hardtanh) versus true σ and tanh. Δ is soft minus hard (percentage points).

Qubit	LSTM-heavy			LSTM-compact		
	Hard	Soft	Δ	Hard	Soft	Δ
Q0	97.96	97.98	+0.02	98.15	98.16	+0.01
Q1	98.26	98.31	+0.05	98.74	98.71	-0.03
Q2	97.41	97.48	+0.07	97.89	97.83	-0.06
Q3	96.59	96.52	-0.07	96.11	96.06	-0.05
Q4	94.96	95.25	+0.29	95.74	95.68	-0.06
Q5	97.93	97.92	-0.01	98.25	98.20	-0.05
Q6	93.92	93.94	+0.02	94.27	94.21	-0.06
Q7	94.79	94.68	-0.11	95.10	95.09	-0.01
Mean	96.48	96.51	+0.03	96.78	96.74	-0.04

B.3 Impact of Quantization

The accuracy results above use FP32 evaluation of models trained with the hardware-faithful hardsigmoid and hardtanh functions. The deployed datapath instead uses int16 arithmetic (Q13 weights, Q15 activations); we quantize only our deployed models, since baselines elsewhere in this paper are not part of our hardware deployment. Table 8 compares int16 accuracy against FP32 on the same eight-qubit test sets.

Table 8. Float32 versus deployed int16 (Q13 weights, Q15 activations) accuracy (%).

Qubit	LSTM-heavy			LSTM-compact		
	FP32	int16	Δ	FP32	int16	Δ
Q0	97.96	97.95	-0.01	98.15	98.14	-0.01
Q1	98.26	98.10	-0.16	98.74	98.73	-0.01
Q2	97.41	97.41	+0.00	97.89	97.90	+0.01
Q3	96.59	96.57	-0.02	96.11	96.05	-0.06
Q4	94.96	94.80	-0.16	95.74	95.70	-0.04
Q5	97.93	97.85	-0.08	98.25	98.24	-0.01
Q6	93.92	93.90	-0.02	94.27	94.27	+0.00
Q7	94.79	94.86	+0.07	95.10	95.11	+0.01
Mean	96.48	96.43	-0.05	96.78	96.77	-0.01

Quantization has negligible impact: mean accuracy drops by at most 0.05 points, and the largest single-qubit loss is 0.16 points (Q1, Q4, LSTM-heavy). The temporal-resolution and Q3 findings reported above therefore hold for the deployed int16 representation.

C Bandwidth–Accuracy Tradeoff: Temporal Compression on Q3

Reducing per-shot data volume allows more qubit streams to share a fixed interboard bandwidth. We test this by pooling the raw Q3 I/Q stream by a factor P before it reaches LSTM-heavy’s fixed 128-feature timestep—unlike the timestep-width sweep in Section 4.6.1, pooling here averages raw samples together, so information is destroyed rather than reorganized. As P grows, fewer timesteps are needed to cover the 704 ns window ($T=11$ at $P=1$ down to $T=2$ at $P=5$), and accuracy stays within 0.18 points of deployed throughout (Figure 14). This robustness makes LSTM-heavy suitable for Q3, where raw resolution matters most (Section 6.3); a contrasting failure mode for LSTM-compact is given in Appendix D.

D Temporal Binning on LSTM-compact

Section C shows that LSTM-heavy tolerates pooling of the raw Q3 stream because pooling only stretches timestep width, leaving per-feature averaging fixed. LSTM-compact has no such fixed per-feature width: each timestep reduces directly to a single mean over B raw samples, so B sets the per-timestep noise floor rather than timestep width alone. Accuracy is nearly flat through $B=64$ (96.03%, within 0.08 points

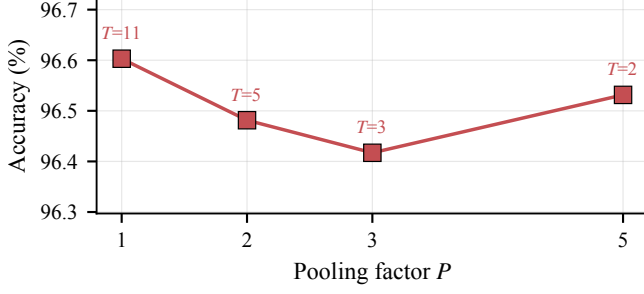


Figure 14. LSTM-heavy accuracy on Q3 vs. pooling factor B on the fixed 128-sample timestep, at 704 ns. Labels report the resulting number of timesteps T .

of deployed), drops to 95.35% at $B=32$ (0.76 points below deployed), and collapses to 86.56% at $B=16$ (Figure 15), where too few raw samples remain in each average for the added temporal resolution (T rising to 88) to compensate.

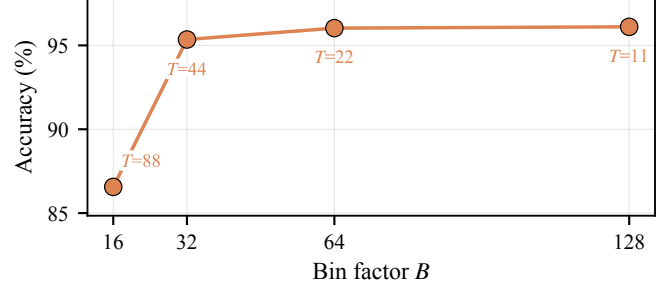


Figure 15. LSTM-compact accuracy on Q3 vs. mean-IQ averaging window B , at 704 ns. Labels report the resulting number of timesteps T .

E Accuracy Across Readout Windows

Accuracy improves substantially from 256 to 512 ns and continues improving through 704 ns, after which returns collapse: extending to 768 ns adds only 0.06 points to mean LSTM-compact accuracy (0.07 for LSTM-heavy), and on Q3, LSTM-heavy moves from 96.59% to 96.60%. We adopt 704 ns as the operating point for the hardware evaluation, since it captures essentially all of the accuracy available from a longer window while returning 64 ns of the 1 μ s control budget and shortening each shot’s exposure to relaxation during measurement.

Table 9. Single-shot readout accuracy (%) at 256 ns. Conventions as in Table 1; *accuracy-only, never synthesized.

Qubit	GMM [15]*	QubiCML [32]	MF-NN [23]	FCN* [20]	LSTM [17]	LSTM-heavy	LSTM-compact
Q0	97.21±0.14	97.06±0.14	97.11±0.14	97.19±0.14	97.35±0.13	97.16±0.14	97.34±0.13
Q1	87.78±0.27	87.75±0.27	88.40±0.26	88.15±0.27	88.46±0.26	87.99±0.27	88.59±0.26
Q2	92.88±0.22	92.73±0.22	93.17±0.21	93.01±0.22	93.19±0.21	92.53±0.22	93.31±0.21
Q3	92.88±0.22	92.88±0.22	93.74±0.20	95.05±0.18	93.84±0.20	95.29±0.18	93.95±0.20
Q4	63.91±0.40	64.50±0.40	66.44±0.39	66.10±0.39	66.33±0.39	66.11±0.39	66.86±0.39
Q5	91.41±0.23	91.05±0.24	91.84±0.23	91.58±0.23	92.03±0.22	91.44±0.23	92.13±0.22
Q6	87.70±0.27	87.60±0.27	88.06±0.27	88.01±0.27	88.08±0.27	87.87±0.27	88.01±0.27
Q7	94.52±0.19	94.44±0.19	94.23±0.19	94.48±0.19	94.60±0.19	94.25±0.19	94.60±0.19
$\mu \pm \sigma$	88.54±9.78	88.50±9.55	89.12±9.02	89.20±9.24	89.24±9.12	89.08±9.21	89.35±8.97

Table 10. Single-shot readout accuracy (%) at 512 ns. Conventions as in Table 1; *accuracy-only, never synthesized.

Qubit	GMM [15]*	QubiCML [32]	MF-NN [23]	FCN* [20]	LSTM [17]	LSTM-heavy	LSTM-compact
Q0	98.04±0.11	97.98±0.12	97.82±0.12	98.09±0.11	98.17±0.11	97.96±0.12	98.17±0.11
Q1	97.70±0.12	97.73±0.12	97.75±0.12	97.73±0.12	98.05±0.11	97.49±0.13	97.99±0.12
Q2	97.35±0.14	97.30±0.14	97.31±0.14	97.21±0.14	97.51±0.13	97.08±0.14	97.50±0.13
Q3	92.53±0.22	94.18±0.20	95.72±0.17	96.15±0.16	95.74±0.17	96.37±0.16	95.90±0.17
Q4	89.03±0.26	88.78±0.26	91.28±0.23	90.57±0.24	91.26±0.24	90.45±0.24	91.27±0.24
Q5	97.49±0.13	97.35±0.13	97.49±0.13	97.60±0.13	97.75±0.12	97.45±0.13	97.76±0.12
Q6	92.48±0.22	92.76±0.21	92.50±0.22	92.94±0.21	93.08±0.21	92.94±0.21	93.14±0.21
Q7	94.81±0.18	94.89±0.18	94.64±0.18	94.94±0.18	95.10±0.18	94.94±0.18	95.13±0.18
$\mu \pm \sigma$	94.93±3.09	95.12±2.99	95.56±2.38	95.65±2.51	95.83±2.39	95.59±2.48	95.86±2.38

Table 11. Single-shot readout accuracy (%) at 768 ns. Conventions as in Table 1; *accuracy-only, never synthesized.

Qubit	GMM [15]*	QubiCML [32]	MF-NN [23]	FCN* [20]	LSTM [17]	LSTM-heavy	LSTM-compact
Q0	97.58±0.13	97.68±0.12	97.97±0.12	98.07±0.11	98.18±0.11	97.93±0.12	98.16±0.11
Q1	98.52±0.10	98.53±0.10	98.50±0.10	98.59±0.10	98.75±0.09	98.24±0.11	98.75±0.09
Q2	97.52±0.13	97.71±0.13	97.74±0.13	97.63±0.13	97.92±0.12	97.53±0.13	97.98±0.12
Q3	88.99±0.26	93.92±0.20	95.88±0.17	96.39±0.16	95.88±0.17	96.60±0.15	96.14±0.16
Q4	95.24±0.18	95.03±0.18	96.12±0.16	95.82±0.17	96.11±0.16	95.46±0.17	96.16±0.16
Q5	97.85±0.12	97.92±0.12	97.87±0.12	97.99±0.12	98.20±0.11	97.87±0.12	98.25±0.11
Q6	92.88±0.21	93.46±0.20	92.83±0.21	93.84±0.20	94.18±0.19	94.04±0.19	94.25±0.19
Q7	94.71±0.18	94.70±0.18	94.63±0.18	94.90±0.18	95.14±0.18	94.73±0.18	95.07±0.18
$\mu \pm \sigma$	95.41±3.02	96.12±1.91	96.44±1.84	96.65±1.59	96.80±1.58	96.55±1.51	96.84±1.56