Degree Project in Embedded Systems

Second cycle, 30 credits

Design of a Dual-Scale Quantized SSM Inference Accelerator for NoC-Based Systems

A case study on Quamba-based AI accelerator

TONG LEI

Design of a Dual-Scale Quantized SSM Inference Accelerator for NoC-Based Systems

A case study on Quamba-based AI accelerator

TONG LEI

Master's Programme, Embedded Systems, 120 credits
Date: June 21, 2026

Supervisor: Yizhi Chen
Examiner: Ahmed Hemani
School of Electrical Engineering and Computer Science
Swedish title: Design av en dubbelskalig kvantiserad SSM-inferensaccelerator för NoC-baserade system
Swedish subtitle: En fallstudie om Quamba-baserad AI-accelerator

© 2026 Tong Lei

Abstract

State Space Models (SSMs) such as Mamba offer linear-time sequence modeling, which makes them suitable for deployment on hardware accelerators. Such accelerators benefit from low communication overhead together with high numerical accuracy, a concern that is especially pronounced on NoC-based accelerators where the width of each transmitted value bounds both the network bandwidth and the router area. The difficulty is that the primary input tensor u contains a small fraction of large-magnitude outliers. Standard INT8 quantization clips these outliers, and resulting errors compound across the recurrent hidden-state update, whereas widening the format to INT9 or INT11 preserves them but by inflating the payload width. This thesis proposes a flag-bit-driven Dual-Scale (DS) dequantization frontend for an SSM processing element (PE), which preserves outliers while keeping a 9-bit payload. The method operates in two stages: a 1-bit scale flag is assigned per element offline during calibration, and this flag then drives a combinational multiplexer at inference time. We implement the design as synthesizable RTL and obtain synthesis results with a 22nm standard-cell library via Synopsys Design Compiler.

In the first stage, offline calibration assigns each element of u a 1-bit flag. The flag selects between a fine-grained inlier scale and a wide-range outlier scale, so the outlier magnitude is recovered at dequantization rather than clipped at quantization. In the second stage, the same flag drives a 2-to-1 multiplexer at inference time. The delay of this multiplexer is absorbed within the existing INT8-to-FP32 conversion, so the design adds no pipeline stages. All dual-scale logic is confined to the u channel, while every other tensor remains on a single-scale INT8 path. The PE is verified against a bit-accurate Python golden model, and the synthesized design is then evaluated for accuracy, area, and timing.

At $\alpha = 0.99$, the DS scheme matches the accuracy of the wider INT9 and INT11 formats in both RMSE and maximum absolute error, while using only a 9-bit payload. Relative to the INT8-Clip baseline, it reduces the peak outlier error by 69%. The Dual-Scale logic adds $144\mu m^2$ of area, which corresponds to a 0.30% overhead on the PE, and all configurations meet timing closure at 1 GHz. A system-level analysis shows that the single PE is compute-bound, so the narrower DS payload reaches INT11-level accuracy without lowering throughput. These results indicate that ASIC combinational logic enables per-element scale selection at negligible cost, achieving the accuracy of wider uniform formats at a narrower payload width.

Keywords

Artificial Intelligence, Large Language Models, State Space Models, Mamba, Dual-Scale Quantization, Post-Training Quantization, Hardware Accelerator, RTL Design, Network-on-Chip

Sammanfattning

Tillståndsrymdsmodeller (SSM) som Mamba erbjuder linjär tidssekvensmodellering, vilket gör dem lämpliga för driftsättning på hårdvaruacceleratorer. Sådana acceleratorer drar nytta av låg kommunikationsoverhead tillsammans med hög numerisk noggrannhet, ett problem som är särskilt uttalat på NoC-baserade acceleratorer där bredden på varje överfört värde begränsar både nätverksbandbredden och routerområdet. Svårigheten är att den primära ingångstensorn u innehåller en liten andel av extremvärden med stor magnitud. Standard INT8-kvantisering begränsar dessa extremvärden, och resulterande fel ökar över den återkommande uppdateringen av dolda tillstånd, medan en utvidgning av formatet till INT9 eller INT11 bevarar dem men genom att blåsa upp nyttolastens bredd. Denna avhandling föreslår ett flaggbitsdrivet Dual-Scale (DS) dekvantiseringsgränssnitt för ett SSM-processningselement (PE), vilket bevarar extremvärden samtidigt som en 9-bitars nyttolast hålls. Metoden fungerar i två steg: en 1-bitars skalflagga tilldelas per element offline under kalibrering, och denna flagga driver sedan en kombinatorisk multiplexor vid inferenstid. Vi implementerar designen som syntetiserbar RTL och erhåller syntesresultat med ett 22nm standardcellsbibliotek via Synopsys Design Compiler.

I det första steget tilldelar offlinekalibrering varje element i u en 1-bits flagga. Flaggan väljer mellan en finkornig inlier-skala och en brett intervall av outlier-skala, så outlier-magnituden återvinns vid dekvantisering snarare än klipps vid kvantisering. I det andra steget driver samma flagga en 2-till-1-multiplexerare vid inferenstid. Fördröjningen för denna multiplexerare absorberas inom den befintliga INT8-till-FP32-konverteringen, så designen lägger inte till några pipeline-steg. All dubbelskalig logik är begränsad till u-kanalen, medan alla andra tensorer förblir på en enskalig INT8-väg. PE verifieras mot en bit-noggrann Python-gyllene modell, och den syntetiserade designen utvärderas sedan för noggrannhet, area och timing.

Vid $\alpha = 0.99$ matchar DS-schemat noggrannheten hos de bredare INT9- och INT11-formaten i både RMSE och maximalt absolut fel, samtidigt som det endast använder en 9-bitars nyttolast. I förhållande till INT8-Clip-baslinjen minskar det toppfelet med 69%. Dual-Scale-logiken lägger till $144\mu^2$ i area, vilket motsvarar en overhead på 0,30% på PE, och alla konfigurationer uppfyller tidsstängningen vid 1 GHz. En systemnivåanalys visar att den enda PE:n är beräkningsbunden, så den smalare DS-nyttolasten når INT11-nivånoggrannhet utan att sänka genomströmningen. Dessa resultat indikerar att ASIC-kombinationslogik möjliggör skalningsval per element till försumbar

kostnad, vilket uppnår noggrannheten hos bredare enhetliga format vid en smalare nyttolastbredd.

Nyckelord

Artificial Intelligence, Large Language Models, Tillståndsrymdsmodeller, Mamba, Dual-Scale Quantization, Post-Training Quantization, Hårdvaruaccelerator, RTL Design, Network-on-Chip

Acknowledgments

I would like to thank my supervisor, Yizhi Chen, for shaping the direction of this thesis work and helping me focus its scope on dual-scale quantization design. Yizhi is truly a responsible mentor in both the projects and the thesis, who is very good at coming up with innovative ideas and proposing implementation plans. From last fall until now, his guidance on my academic work was invaluable. I am also grateful to my examiner, Prof. Ahmed Hemani, for overseeing this thesis.

I sincerely thank my co-author and close friend Ruichi Han, for collaborating on the first paper I have ever contributed to as a third author. It has been a monumental event in my entire life which I would always keep in my heart.

I would also like to thank my friends Xina Luo, Yurun Chao, Ziyin Yang, Yuting Cui, Xingyi Gou, Nake Zhao, Yang Ou, Jiayi Luo, Ratchanon Leungon, Ivy Liu, Reid Zemek, Joel Frezghi Tesfazghi, Sampan Manna, Anishik Gupta. I wouldn't have persisted through these two years of study abroad without their kindness and help, both in life and on campus.

Finally, the greatest gratitude is presented to my parents. They have been supporting me for my study at KTH, not only financially, but also wholeheartedly for me to broaden my horizon and pursue an overseas Master's degree. I love them as I always did and I always will.

Stockholm, June 2026
Tong Lei

Contents

1 Introduction **1**
1.1 Background . . . 1
1.2 Problem . . . 2
1.2.1 Original problem and definition . . . 2
1.2.2 Research Question . . . 2
1.2.3 Scientific and engineering issues . . . 3
1.3 Purpose . . . 3
1.4 Goals . . . 3
1.5 Research Methodology . . . 4
1.6 Delimitations . . . 4
1.7 Structure of the thesis . . . 5
1.8 Declaration of Generative AI usage . . . 5

2 Background **7**
2.1 From General-Purpose Processors to Custom Silicon . . . 7
2.1.1 Why GPUs Are Not Enough: The Memory Wall . . . 7
2.1.2 ASIC Accelerators for Neural Networks . . . 9
2.1.3 Network-on-Chip as the Interconnect Fabric . . . 9
2.1.4 Performance Analysis: The Roofline Model . . . 9
2.2 State Space Models . . . 10
2.2.1 Limitations of Transformers and the Case for SSMs . . . 10
2.2.2 The SSM Recurrence: Continuous to Discrete . . . 11
2.3 Quantization for Neural Network Inference . . . 12
2.3.1 Basics: Why Quantize? . . . 12
2.3.2 Post-Training Quantization and Scale Factors . . . 12
2.3.3 The Outlier Problem in SSMs . . . 13
2.4 Related work . . . 15
2.4.1 Algorithmic Quantization Recipes for SSMs . . . 15
2.4.2 Hardware Accelerators for SSMs . . . 16

2.4.3 The Gap in the Literature . . . 18
2.5 Summary . . . 18

3 Methodology **19**
3.1 Dual-Scale Quantization Methodology . . . 21
3.1.1 The Outlier Problem and Evaluated Schemes . . . 21
3.1.2 Mathematical Formulation . . . 23
3.1.3 Numerical Accuracy Validation . . . 24
3.2 Hardware Design Framework . . . 26
3.2.1 Processing Element Microarchitecture . . . 26
3.2.2 System-Level NoC Integration . . . 27
3.3 Evaluation Methodology . . . 28

4 Implementation Details **31**
4.1 Baseline SSM Compute Core . . . 31
4.1.1 Micro-Pipelined State Recurrence . . . 32
4.2 Complete RTL Architecture . . . 34
4.3 Dual-Scale Hardware Implementation . . . 36
4.4 Parametrized Design Space for Evaluation . . . 37
4.5 System-Level Integration Modeling . . . 38

5 Experiment Results **41**
5.1 Experimental Setup . . . 41
5.2 Numerical Accuracy Evaluation . . . 42
5.3 Error Propagation in SSMs . . . 49
5.4 Hyperparameter Robustness . . . 53
5.5 Hardware Area and Timing Analysis . . . 55
5.6 System-Level Throughput and NoC Bandwidth Analysis . . . 60
5.7 Discussion . . . 63
5.8 Summary . . . 65

6 Conclusions and Future work **69**
6.1 Conclusions . . . 69
6.2 Future work . . . 70
6.3 Reflections . . . 71

References **73**

A Algorithms **81**
A.1 Dual-Scale Offline Calibration 82
A.2 SSM Processing Element: Per-Token Inference 83
A.3 Validity Propagation via `vld_pipe` Shift Register 85
A.4 Parameterized Synthesis Sweep for Fair Area and Timing Comparison 86

List of Figures

2.1 Conceptual Comparison of Dynamic Scale Selection Overhead between GPU and ASIC 8
2.2 Structural Comparison between Transformer and SSM 10
2.3 Quantization Error Compounding over Timesteps in SSM recurrence 14

3.1 Overview of the Methods 19
3.2 Decision Boundaries and Scale Ranges of the Evaluated Quantization Schemes 22

4.1 Finite-State-Machine Control Flow of the SSM Compute Core 32
4.2 Overall Hardware Design Framework of the SSM Compute Core 33
4.3 Datapath of the ST_STATE_LOOP FSM State 34
4.4 Detailed RTL Architecture of the SSM Compute Core 35

5.1 Per-Element Quantization Error Distribution at $\alpha = 0.99$ with 200 seeds 43
5.2 Accuracy Comparison across Quantization Schemes in terms of RMSE at $\alpha = 0.99$ 44
5.3 Accuracy Comparison across Quantization Schemes in terms of MaxAE at $\alpha = 0.99$ 44
5.4 Cumulative distribution function of relative quantization errors at $\alpha = 0.99$ 46
5.5 Absolute and Relative Error Distributions across Evaluated Quantization Schemes 48
5.6 Residual Error per Sample sorted by Ideal Magnitude 49
5.7 Mean L2 Error of the Hidden State h_t over 64 timesteps across varying alpha thresholds 51
5.8 Mean Absolute Error of the Output Projection y_t over 64 timesteps across varying alpha thresholds 52

5.9 Box Plot Distribution of RMSE Variance across Different Calibration Percentiles 54
5.10 Total Area Comparison & Logic Overhead at 1GHz, 22nm technology node 56
5.11 Hierarchical Area Breakdown across Evaluated Quantization Schemes 57
5.12 Pareto Frontier of Single-PE Area Overhead versus Quantization Error (RMSE) 58
5.13 Roofline of a single SSM-PE across quantization schemes and state dimensions with $F_{PE} = 1$ GHz 61
5.14 Achieved Throughput against NoC per-port Bandwidth across Flit Widths 62
5.15 Summary of accuracy, area, robustness, and bandwidth efficiency across the evaluated schemes 66

List of Tables

5.1 Summary of Experimental Parameters 42
5.2 Timing closure results across evaluated configurations. 58

List of acronyms and abbreviations

ASIC	Application-Specific Integrated Circuit
DRAM	Dynamic Random-Access Memory
DS	Dual-Scale
FPGA	Field-Programmable Gate Array
FSM	Finite-State Machine
GPU	Graphics Processing Unit
II	Initiation Interval
LLM	Large Language Model
MaxAE	Maximum Absolute Error
MUX	Multiplexer
NoC	Network-on-Chip
PE	Processing Element
PTQ	Post-Training Quantization
RMSE	Root-Mean-Square Error
RTL	Register-Transfer Level
SiLU	Sigmoid Linear Unit
SSM	State Space Model

Chapter 1

Introduction

The rapid scaling of Large Language Models (LLMs) has driven the need for highly efficient hardware architectures capable of processing extensive context windows [1, 2, 3]. While State Space Models (SSMs) offer a promising linear-time alternative to traditional Transformers, their large hidden-state activations must still be streamed between the many distributed processing elements of a Network-on-Chip (NoC) accelerator, introducing stringent area and bandwidth constraints on activation transport [4, 3]. This chapter introduces the context of SSM inference, defines the critical challenge of quantization outliers inherent to recurrent models, and outlines the research objectives and contributions of this thesis in developing a hardware-efficient Dual-Scale (DS) dequantization frontend that bridges the gap between low-bandwidth network communication and high-precision SSM computation [5, 6].

1.1 Background

The rapid evolution of artificial intelligence has been largely driven by Transformer models; however, their self-attention mechanisms suffer from quadratic computational and memory complexity with respect to sequence length [3]. To overcome this quadratic scaling barrier for long context tasks, the Mamba architecture employs a selective state space model (SSM) in which hidden states are recurrently updated using input-dependent parameters [4]. This formulation enables linear-time sequence modeling, avoiding the quadratic computational complexity of traditional Transformers [4, 3]. Recent advancements further reformulate this recurrence using structured matrix operations to improve general hardware efficiency and theoretical scalability [5]. Despite these algorithmic improvements, deploying SSMs

at scale requires highly parallel, distributed hardware. NoC accelerators are typically employed to provide the necessary computation density, but they operate under strict physical area and communication bandwidth constraints, making data compression techniques like quantization mandatory [7, 8].

1.2 Problem

1.2.1 Original problem and definition

To enable efficient deployment on resource-constrained platforms, SSM accelerators typically employ Post-Training Quantization (PTQ) formats, such as INT8, which require custom hardware dequantization to floating-point formats for the core SSM computation [7, 6, 9]. However, outliers in SSM recurrent computations severely degrade standard uniform INT8 quantization, leading to severe accuracy degradation [6, 10]. The recurrent nature of SSMs introduces a reliability challenge: quantization errors in the hidden state propagate across subsequent time steps and accumulate through the sequence [4, 6]. Specifically, standard uniform quantization clips extreme outlier activations, leading to substantial output deviations [11].

1.2.2 Research Question

The central research question of this thesis is:

Can a flag-bit-driven dual-scale dequantization frontend, implemented as a combinational ASIC datapath, achieve accuracy parity with wider uniform bit-width formats while incurring negligible silicon area overhead and zero additional pipeline latency?

This question concerns whether the structurally critical outliers that uniform clipping discards can be preserved directly in hardware, by selecting between two quantization scales per element rather than widening the numerical format uniformly. This question is worth answering because the dominant alternative, increasing the global bit-width, inflates both the on-chip area of the dequantization datapath and the payload that must traverse the NoC [12]; a scale-selection mechanism that adds neither would allow accurate low-bit SSM inference at the cost of a single control bit. The central challenge is that such per-element scale selection must be resolved without stalling the pipeline or adding arithmetic stages, since the SSM processing element is latency-sensitive and area-constrained [8]. This thesis addresses the question through the four sub-goals defined in Section 1.4, which decompose

it into accuracy validation, RTL implementation, hardware cost evaluation, and system-level analysis.

1.2.3 Scientific and engineering issues

This problem is fundamentally driven by conflicting scientific and engineering challenges. From a scientific standpoint, the non-linear outlier distribution of SSM activations cannot be effectively captured by linear, static quantization boundaries. A single uniform scale cannot simultaneously provide fine-grained resolution for the majority of small activations and a wide dynamic range for extreme outliers. From an engineering standpoint, handling these outliers by simply increasing the global bit-width or utilizing complex floating-point architectures incurs unacceptable Hardware Area and NoC Communication overheads. High-precision formats saturate the limited network routing bandwidth, while complex dynamic-range datapaths exceed the stringent area and power budgets of customized Processing Elements (PEs) [13, 14, 15].

1.3 Purpose

The primary purpose of this thesis is to design, implement and evaluate a hardware-efficient quantization microarchitecture that mitigates the SSM outlier problem without incurring the area and bandwidth costs associated with higher bit-widths. By bridging the gap between mathematical quantization algorithms and physical circuit design, this research aims to demonstrate that targeted, non-linear hardware interventions can improve the accuracy-area trade-off of SSMs on custom silicon relative to uniform-precision quantization.

1.4 Goals

The goal of this project is to design and evaluate a customized hardware quantization datapath that resolves the SSM outlier problem without the area and bandwidth penalties of wider uniform formats. This has been divided into the following four specific sub-goals:

1. Subgoal 1: Design and verify a baseline RTL compute core tailored specifically for SSM inference;

2. Subgoal 2: Implement a Dual-Scale dequantization hardware frontend that dynamically switches scaling factors at runtime based on a generated flag bit, converting compressed integer network payloads into FP32 formats for internal computation;

3. Subgoal 3: Evaluate the area and timing overhead of the Dual-Scale logic using industry-standard logic synthesis tools;

4. Subgoal 4: Model the system-level throughput and NoC bandwidth trade-offs to confirm the macroscopic efficiency of the proposed design.

1.5 Research Methodology

This thesis employs a rigorous hardware-software co-design approach. The methodology begins with software data profiling to analyze activation distributions and validate the dual-scale mathematical concept using a bit-accurate Python Golden Model. This is followed by a hardware implementation phase, where the PEs are designed in SystemVerilog RTL. The hardware costs and timing feasibility are then evaluated through top-down logic synthesis targeting a 1GHz frequency on a standard technology node. Finally, the synthesized hardware metrics are integrated into an analytical system model to evaluate computation and communication trade-offs within a NoC topology.

1.6 Delimitations

To maintain a focused scope, specific delimitations are applied to this research. The RTL implementation focuses on a single PE and the hardware dequantization frontend. Consequently, system-level NoC communication aspects, such as router contention, network topology limits and global latency, are evaluated via analytical modeling rather than cycle-accurate multi-node RTL simulation. Dual-Scale optimization is restricted to tensor u to match the outlier distribution in Quamba. Furthermore, the thesis assumes that the forward-pass quantization occurs prior to PE ingestion. Therefore, the hardware evaluation focuses on the area cost of the INT-to-FP32 dequantization frontend and the payload-width implications of receiving compressed activations; all subsequent internal SSM state calculations are executed in standard FP32 to ensure mathematical stability.

1.7 Structure of the thesis

Chapter 2 reviews the background and related work concerning SSM architectures, hardware accelerators, and neural network quantization. Chapter 3 outlines the research methodology, including the dual-scale algorithmic design and the hardware evaluation framework. Chapter 4 details the RTL implementation of both the baseline SSM compute core and the proposed Dual-Scale dequantization frontend. Chapter 5 presents the experimental results and analysis, evaluating the design across numerical accuracy, hardware area and timing, and system-level throughput. Finally, Chapter 6 concludes the thesis, acknowledges current limitations, and outlines directions for future work.

1.8 Declaration of Generative AI usage

In accordance with the academic guidelines, I hereby declare that generative AI tools were utilized during the thesis work strictly for language polishing and stylistic refinement. The AI was neither used to generate novel academic ideas, conduct literature reviews, now perform data analysis. All concepts, methodologies, and conclusions are entirely my original work, and I maintain full accountability for the integrity of this document.

Chapter 2

Background

This chapter establishes the technical foundation required to understand the research presented in this thesis. It begins by examining why general-purpose processors are insufficient for large-scale neural network inference, and how custom silicon accelerators organized around PEs and NoC interconnects address this gap. It then introduces the mathematical structure of State Space Models and explains why their recurrent nature makes them particularly sensitive to quantization noise. Finally, it reviews the principles of neural network quantization, characterizes the outlier problem specific to SSM activations, and surveys recent related work to identify the open hardware challenge that motivates this thesis.

2.1 From General-Purpose Processors to Custom Silicon

2.1.1 Why GPUs Are Not Enough: The Memory Wall

Graphics Processing Units (GPUs) have become the de facto platform for training large neural networks, owing to their massively parallel architecture of thousands of compute cores. However, the autoregressive inference workload of modern language models exposes a fundamental mismatch: each forward pass generates a single output token, and strict latency limits keep throughput-oriented GPUs far from their peak utilization, often below half of their achievable throughput under response-time constraints [1].

The deeper bottleneck is the *memory wall*. Autoregressive inference requires loading the full model parameter set from off-chip DRAM on every generated token. For a model with billions of parameters, this implies

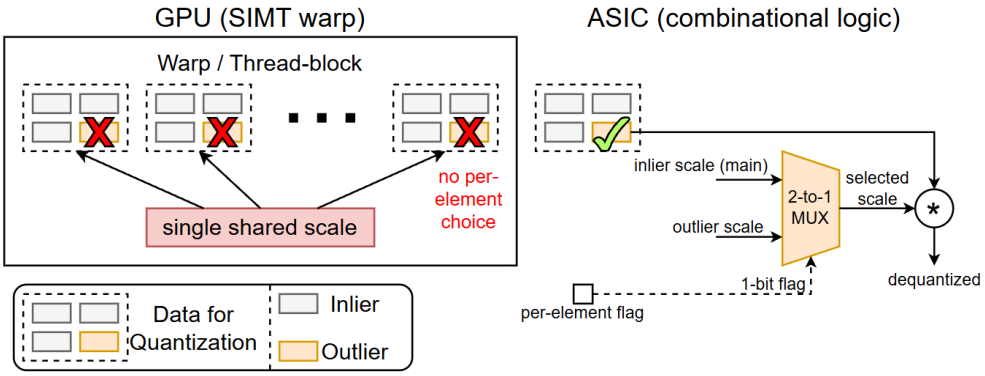

Figure 2.1: Conceptual Comparison of Dynamic Scale Selection Overhead between GPU and ASIC

transferring gigabytes of data per token, regardless of how fast the arithmetic units operate. The ratio of computation to data movement – the arithmetic intensity – is therefore extremely low, placing GPU-based SSM inference firmly in the memory-bound regime of the Roofline model [16, 17].

Beyond bandwidth, the CUDA programming model imposes a structural constraint relevant to quantization. GPU threads execute in groups of 32, called warps. Divergent control flow within a warp, such as applying different scaling factors to different elements, causes serialization and significant throughput loss [18]. This constraint makes per-element dynamic scale switching, which is central to the dual-scale approach proposed in this thesis, prohibitively expensive on GPU hardware. Each PE executes a fixed, tailored datapath with no generality overhead, so control decisions such as data-dependent scale selection can be embedded directly in the datapath rather than incurring the branch-divergence penalty they would on a SIMT architecture. Figure 2.1 contrasts the warp divergence penalty of GPU-based dynamic quantization with the proposed zero-latency ASIC multiplexer.

This per-element divergence is precisely why software outlier recipes for SSMs remain GPU-oriented and avoid true per-element precision switching; recent SSM quantization frameworks such as SSDi8 [9] instead reformulate the computation to keep a uniform integer path, rather than branching per element.

Custom Application-Specific Integrated Circuits (ASICs) eliminate these constraints by design. Each PE executes a fixed, tailored datapath with no generality overhead, so control decisions such as data-dependent scale selection can be embedded directly in the datapath rather than incurring the branch-divergence penalty they would on a SIMT architecture. Furthermore,

on-chip scratchpad memories can be sized and organized to match the exact data reuse patterns of the target model, replacing costly DRAM accesses with local register reads [19].

2.1.2 ASIC Accelerators for Neural Networks

The deployment of Large Language Models (LLMs) requires massive computational throughput and memory bandwidth, which traditional von Neumann architectures cannot efficiently supply [1]. Consequently, modern AI accelerators utilize spatial architectures consisting of numerous PEs arranged in two-dimensional arrays. This spatial layout allows data to be reused locally within PE registers and scratchpad buffers, significantly minimizing the energy-intensive and high-latency accesses to off-chip DRAM [19].

2.1.3 Network-on-Chip as the Interconnect Fabric

To manage the immense data flow between these distributed PEs and global memory, accelerators rely on NoC interconnects, often utilizing mesh topologies, to provide scalable, high-bandwidth communication [12]. Within these NoC architectures, data packets traverse through distributed routers composed of multi-stage buffers and crossbar switches. Crucially, the dynamic power consumption, routing latency, and silicon area of a NoC are directly proportional to the data payload bit-width [12]. Therefore, transmitting uncompressed 16-bit or 32-bit floating-point data across the network rapidly saturates the available routing bandwidth and inflates physical layout congestion, making rigorous data compression and quantization at the network level mandatory [7, 20].

2.1.4 Performance Analysis: The Roofline Model

The performance bounds of these NoC-based spatial accelerators are classically analyzed using the Roofline model, which maps system throughput against arithmetic intensity – defined as the ratio of floating-point operations per byte of memory accessed [16]. Under this paradigm, workloads are either ”compute-bound” (limited by the peak operations per second of the PEs) or ”memory-bound” (limited by the NoC and external memory bandwidth) [16, 1]. Because LLMs and SSMs possess large parameter counts and rely on continuous parameter streaming, they are commonly expected to exhibit low arithmetic intensity and to operate in the memory-bound region [3]. Whether

this expectation holds for a specific PE design depends on its dataflow, as examined in Chapter 5.

2.2 State Space Models

2.2.1 Limitations of Transformers and the Case for SSMs

While Transformers have dominated sequence modeling, their self-attention mechanism imposes an $O(N^2)$ computational and memory complexity relative to the sequence length N, severely limiting their applicability in extreme long-context scenarios [3]. The Mamba architecture addresses this by employing a selective state space model (SSM) in which hidden states are recurrently updated using input-dependent parameters, enabling $O(N)$ linear-time sequence modeling [4]. Figure 2.2 illustrates the structural difference between the all-to-all attention mechanism of Transformers and the linear sequential state-passing of SSMs.

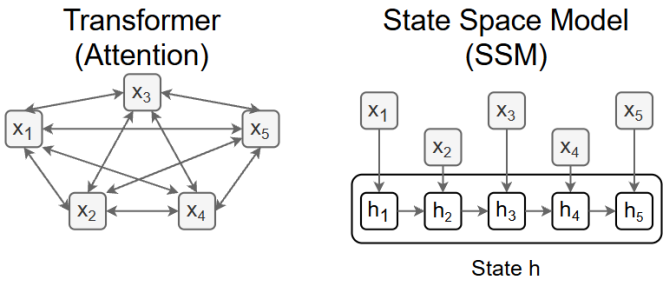

Figure 2.2: Structural Comparison between Transformer and SSM

Recent advancements further reformulate this recurrence using structured matrix operations, known as State Space Duality (SSD), to improve hardware efficiency and parallelization [5]. The modern state space model lineage began with the Structured State Space sequence model (S4) [21], which demonstrated that a linear state-space recurrence could match attention on long-range benchmarks at subquadratic cost. Mamba then introduced an input-dependent selective-scan mechanism [4], and Mamba-2 subsequently recast this recurrence through the State Space Duality (SSD) framework to expose matrix-multiplication parallelism [5]. The practical relevance of SSMs is reinforced by their adoption beyond language. Vision Mamba [22] applies bidirectional SSMs to image representation, and Jamba [23] integrates Mamba blocks into a large-scale hybrid Transformer.

2.2.2 The SSM Recurrence: Continuous to Discrete

Mathematically, SSMs are grounded in continuous-time differential equations that map a 1-dimensional input sequence $x(t) \in \mathbb{R}$ to an output $y(t) \in \mathbb{R}$ through an N-dimensional latent state $h(t) \in \mathbb{R}^N$. This dynamic system is formalized by the following linear ordinary differential equations (ODEs):

$$h'(t) = \mathbf{A}h(t) + \mathbf{B}x(t) \tag{2.1}$$

$$y(t) = \mathbf{C}h(t) \tag{2.2}$$

where $\mathbf{A} \in \mathbb{R}^{N\times N}$ represents the state evolution matrix, while $\mathbf{B} \in \mathbb{R}^{N\times 1}$ and $\mathbf{C} \in \mathbb{R}^{1\times N}$ are the input and output projection matrices, respectively [4]. To run on digital hardware, this continuous system is discretized with a per-step size Δ, transforming the continuous matrices $\mathbf{A}$ and $\mathbf{B}$ into their discrete counterparts $\bar{\mathbf{A}}$ and $\bar{\mathbf{B}}$, yielding the foundational recurrence:

$$h_t = \bar{\mathbf{A}}h_{t-1} + \bar{\mathbf{B}}x_t \tag{2.3}$$

Because h_t is updated sequentially, the SSM acts as a dynamic, compressed memory buffer whose contents evolve token by token [5].

The Mamba architecture specializes this generic recurrence with input-dependent parameters, a data-dependent step size, a skip connection, and an output gate [4]. Throughout this thesis, the discrete input x_t is the input-projection tensor denoted u_t – the activation transmitted in compressed integer form across the NoC and dequantized at the PE boundary, making it the locus of the quantization problem addressed here. The effective step size is obtained through a softplus activation, $\Delta_{sp} = \mathrm{softplus}(\Delta + b_\Delta)$, and the discrete state-evolution coefficient for state dimension n is

$$\bar{A}_n = \exp(-\exp(A_n)\,\Delta_{sp}). \tag{2.4}$$

The per-dimension hidden state is then updated as

$$h_{t,n} = \bar{A}_n\, h_{t-1,n} + \Delta_{sp}\, u_t, \tag{2.5}$$

and the gated output token is formed by the output projection, a D-weighted skip path, and a SiLU gate driven by the auxiliary input z_t:

$$y_t = \Big(\sum_{n=1}^{N} B_n C_n\, h_{t,n} + D\, u_t\Big) \cdot \mathrm{SiLU}(z_t). \tag{2.6}$$

In this formulation the input matrix B is applied within the output

projection of Equation 2.6 rather than per-timestep inside the state update of Equation 2.5; this is a deliberate compute-core simplification whose scope and justification are stated in Section 1.6. Equations 2.4–2.6 define the exact datapath synthesized in Chapter 4, and u_t is the single tensor onto which the Dual-Scale dequantization frontend is applied.

2.3 Quantization for Neural Network Inference

2.3.1 Basics: Why Quantize?

Quantization reduces the numerical precision of a neural network's weights and activations from floating-point to low-bit integer formats, lowering both memory footprint and the bandwidth required to move data between memory and compute units [24, 25]. Quantization methods fall into two broad categories: quantization-aware training (QAT), which retrains the model to adapt to low precision, and post-training quantization (PTQ), which calibrates a fixed model offline without gradient updates [24]. Because retraining large language and state space models is prohibitively expensive, this thesis adopts the PTQ setting, in which the quantization scale is derived from activation statistics and applied directly at inference time. This choice is not without cost: for structured SSMs, quantization-aware training has been shown to tolerate substantially more aggressive precision reduction than PTQ, with the recurrent kernel parameters identified as the hardest to quantize [26]. PTQ is nonetheless adopted here because it avoids the prohibitive retraining cost, shifting the burden of outlier handling entirely onto the inference-time dequantization datapath.

2.3.2 Post-Training Quantization and Scale Factors

To alleviate the NoC bandwidth constraints outlined by the Roofline model, Neural Network Quantization maps high-precision floating-point values to lower-precision integer formats [24]. In standard uniform symmetric PTQ, a floating-point tensor $\mathbf{X}$ is mapped to a b-bit integer format $\mathbf{X}_q$. The quantization process is mathematically defined by the clipping threshold X_{max} and the resulting scaling factor S:

$$S = \frac{X_{max}}{2^{b-1} - 1} \tag{2.7}$$

$$\mathbf{X}_q = \text{Round}\left(\text{Clip}\left(\frac{\mathbf{X}}{S}, -(2^{b-1}-1), 2^{b-1}-1\right)\right) \tag{2.8}$$

Crucially, to preserve the stability of the $\bar{\mathbf{A}}h_{t-1}$ recurrence shown in Equation 2.3, the core SSM accumulation must be performed in high-precision floating-point (FP32). Therefore, the hardware focus of this thesis is isolated to the network-to-compute boundary, implementing the inverse dequantization function:

$$\tilde{\mathbf{X}} = \mathbf{X}_q \times S \tag{2.9}$$

where $\tilde{\mathbf{X}}$ is the restored pseudo-floating-point tensor inside the PE. The quantization noise $\mathbf{E} = \mathbf{X} - \tilde{\mathbf{X}}$ is strictly dictated by the initial choice of the threshold X_{max} [24].

The choice of X_{max}, and more generally the handling of activation outliers, has become the central challenge in low-bit LLM quantization. Dettmers et al. [27] first showed that transformer activations contain systematic high-magnitude outliers that dominate quantization error, and proposed isolating them in a separate higher-precision path. Subsequent work reduced this overhead through offline transformations: SmoothQuant [28] migrates the quantization difficulty from activations to weights, AWQ [29] protects a small fraction of salient channels identified from the activation distribution, and GPTQ [30] minimizes layer-wise reconstruction error during weight quantization. These methods are predominantly software-oriented and target GPU execution, where per-element precision selection incurs branching overhead. This thesis instead resolves the outlier problem at the hardware dequantization boundary, where a custom datapath can apply per-element scale selection at no pipeline cost. A complementary hardware line encodes outliers directly in the datapath: OliVe pairs each outlier with a sacrificial neighbour [31], while per-element precision tags select an encoding width at the array boundary in Spark [32]. The dual-scale frontend proposed here belongs to this hardware-encoding family, but unlike these schemes, confines selection to a single flag at the dequantization stage.

2.3.3 The Outlier Problem in SSMs

Applying this dequantization to the Mamba input tensor u (i.e. taking $X = u$ in Equation 2.9), the accuracy relies entirely on the choice of the clipping threshold X_{max}. Modern LLMs and SSMs exhibit a well-documented phenomenon where activation tensors—specifically the input projection tensor u—develop extreme, large-magnitude outlier distributions

containing rare but structurally critical outliers [27, 6, 33]. A fundamental trade-off arises: utilizing the absolute maximum value (absmax) for X_{max} preserves these large outliers but drastically inflates the scale S, thereby destroying the quantization resolution for the dense majority of smaller inlier activations [24].

Conversely, employing percentile-based clipping (e.g., setting X_{max} to the 99.9th percentile) provides excellent, fine-grained resolution for the dense regions of the distribution but forcefully truncates the magnitude of extreme outlier activations [6, 24]. While standard convolutional networks can often absorb localized clipping noise [24], truncating these massive activations in SSMs injects immediate numerical errors into the discrete input u. The recurrent sequential dependency introduces a severe mathematical vulnerability: any numerical errors introduced into the input u at a specific time step are recursively multiplied by the transition matrix $\bar{A}$ in subsequent iterations, causing deviations to accumulate and compound across the sequence length [6]. As visually summarized in Figure 2.3, quantization error injected into the discrete input are recursively amplified across subsequent hidden state updates along the sequence, degrading the generative fidelity of the model and necessitating non-linear hardware interventions [11].

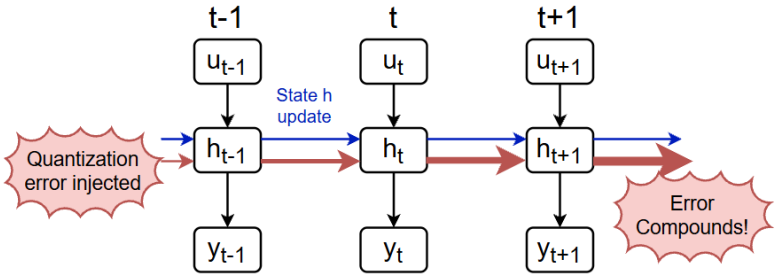

Figure 2.3: Quantization Error Compounding over Timesteps in SSM recurrence

This recurrent amplification distinguishes SSM quantization from transformer quantization: Pierro and Abreu [34] identified persistent outlier channels specific to recurrent Mamba activations, and the Quamba line of work [6, 11] confirmed that input-dependent parameters require dedicated outlier handling to preserve generative fidelity.

2.4 Related work

Addressing the unique computational patterns and quantization vulnerabilities of State Space Models requires synergistic innovation across both software algorithms and hardware microarchitectures. While extensive literature exists for Transformer-based accelerators, research targeting the precise hardware-software co-design of SSMs is a rapidly emerging field.

2.4.1 Algorithmic Quantization Recipes for SSMs

To tackle the severe accuracy degradation caused by naive uniform quantization in Mamba models, Chiang et al. [6] introduced Quamba, a highly specialized post-training quantization recipe. The authors identified that standard activation tensors—particularly the input projection tensor u—exhibit small-fraction large-magnitude outlier distributions. To recover precision for the dense inlier majority, Quamba suppresses the maximum input values using a percentile-based threshold, trading away fidelity in the outlier tail in exchange for a finer quantization step over the bulk of the distribution. For the SSM output activations, it instead applies a Hadamard transform to map values into an outlier-free space prior to quantization. While Quamba establishes that u is the critical tensor and demonstrates strong end-to-end accuracy in PyTorch simulations, its treatment of input outliers is fundamentally lossy: the clipped extreme values are never recovered. The question of whether these structurally critical outliers can instead be preserved, rather than discarded, without widening the uniform bit-width remains open and is precisely the gap this thesis addresses.

Building upon the foundation of the original Quamba, Chiang et al. [11] proposed Quamba2, which extends the recipe to the Mamba-2 architecture by sorting and clustering activation channels to enable per-group 8-bit quantization of the linear recurrence inputs. This work further solidifies the consensus that SSM recurrent pathways are exceptionally fragile to quantization noise. Additionally, Chen and Hemani [35] explored Quamba-SE, investigating soft-edge quantizers designed to better capture the non-linear activation boundaries of SSMs without rigid clipping thresholds. Parallel efforts extend SSM quantization to new domains and bit-widths: SSDi8 [9] maintains a persistent INT8 path for the State Space Duality formulation by decoupling element-wise from matrix operations, while quantized Vision Mamba variants [33, 36, 37] address hidden-state outliers in image tasks. These confirm that outlier-induced error propagation is intrinsic to the

recurrence rather than specific to one model or modality.

Beyond scale-based clipping, a parallel line of work removes outliers through computational invariance. QuaRot [38] applies Hadamard rotations to eliminate outlier features before 4-bit quantization, and MambaQuant [10] adapts variance-aligned rotation methods to the Mamba family. These rotation-based schemes require additional online transform hardware, whereas the dual-scale approach studied here confines all outlier handling to a combinational selection within the existing dequantization path.

While these algorithmic advancements provide critical theoretical baselines, they inherently ignore the physical area and timing implications of their non-linear quantization functions when mapped to Application-Specific Integrated Circuits (ASICs).

2.4.2 Hardware Accelerators for SSMs

In parallel to algorithmic efforts, recent research has explored dedicated hardware architectures designed to execute the sequential SSM operations efficiently. Li et al. [13] proposed MARCA, a reconfigurable accelerator optimized for both the element-wise operations and the core recurrent loops of Mamba. MARCA addresses the low arithmetic intensity of SSMs by utilizing specialized memory hierarchies to maximize the on-chip reuse of the transition matrix $\bar{A}$. However, while MARCA improves the baseline PE efficiency, it assumes standard uniform data ingestion and does not provide specialized datapaths for mitigating the extreme quantization outliers inherent to compressed NoC payloads.

Focusing on highly parallelized structures, Raja et al. [14] demonstrated EpochCore, an architecture that leverages systolic-array-based designs to execute the structured matrix operations of SSMs at scale. Similarly, Wei et al. [7] introduced LightMamba, which pioneers a mixed-precision algorithm-hardware co-design implemented on FPGA platforms. LightMamba successfully demonstrates that tailoring the compute datapath to specific bit-widths can yield significant energy efficiency gains. Furthermore, Wang et al. [39] developed FastMamba, an FPGA accelerator tailored for Mamba-2 that utilizes power-of-two quantization schemes alongside hardware-friendly nonlinear approximations for the SiLU and Exp functions. The space of dedicated SSM accelerators has since expanded rapidly: eMamba [8] presents an end-to-end edge framework evaluated down to a 22nm technology node, a fully quantized Mamba-2 FPGA accelerator targets the two RMSNorm layers via segmented quantization [40], and further reconfigurable and FPGA

designs target sparsity, speculative decoding, token efficiency, and vision workloads [15, 17, 20, 41, 42]. Across this body of work, however, the activation path is ingested at a uniform precision; none introduces a per-element scale-selection mechanism at the dequantization boundary.

Outlier-aware hardware has also been explored outside the SSM domain. OLAccel [43] processes the rare high-magnitude activations on a dedicated high-precision processing element separate from the dense low-precision array, while OverQ [44] exploits multiplexing logic within a systolic array to store outliers without a separate compute unit. The dual-scale frontend in this thesis is closest in spirit to the latter, but applies combinational multiplexing specifically at the INT-to-FP32 dequantization stage of an SSM PE, so that outlier handling adds neither a separate datapath nor a pipeline stage. Beyond these two, a broader hardware-outlier lineage has emerged. Early schemes store outliers separately in higher precision [45]; later designs encode them locally through victim pairing [31] or index-pair encoding [46]; and the most recent retain them through microscaling or fine-grained mixed precision with per-block metadata [47, 48, 49]. A second branch couples outlier-aware quantization with the memory hierarchy, preserving critical values in higher-precision storage [50]. All of these incur either a separate datapath, a sparse or metadata side-channel, or a variable-length decode. The dual-scale frontend deliberately avoids all three, resolving outlier handling with a single combinational flag inside the existing dequantization path.

Taken together, these two research threads remain largely disjoint. Algorithmic SSM quantization recipes [6, 11, 9, 33] characterize where outliers arise and how lossy clipping degrades the recurrence, but stop at PyTorch-level validation and do not report any synthesized hardware cost. Conversely, dedicated SSM accelerators [13, 15, 14, 7, 39, 8, 40] optimize the compute datapath but ingest activations at a uniform precision, inheriting the very clipping-versus-bandwidth dilemma the algorithmic work exposes. The general-purpose hardware-outlier encoders [31, 32, 45, 47] that could bridge this divide were designed for feed-forward Transformer and CNN workloads, where a localized encoding error does not propagate. None has been evaluated on the recurrent SSM datapath, where a single mishandled outlier compounds across timesteps. The specific combination this thesis targets – per-element outlier preservation, realized as a synthesized combinational frontend, on a recurrent SSM PE – therefore sits in an unaddressed intersection.

2.4.3 The Gap in the Literature

Despite the significant advancements presented by the SSM accelerators and outlier-aware encoders surveyed above, a critical gap remains at the intersection of system-level communication and dynamic outlier handling. Existing accelerators typically default to uniform lower-precision formats (e.g., standard INT8) which, as proven by Quamba, corrupts the hidden state, or they revert to uniform higher-precision formats (e.g., INT9 or INT11), which increases the NoC routing bandwidth and external DRAM traffic. Implementing per-element outlier preservation—specifically, dynamically routing data through a dual-scale dequantization frontend based on an implicit flag—has not been physically realized and evaluated for its area and timing overhead. This gap motivates the hardware-software co-design approach proposed in this thesis.

2.5 Summary

This chapter has established three fields of knowledge necessary for this thesis. First, the physical constraints of NoC-based spatial accelerators were analyzed. SSM inference is commonly expected to operate in the memory-bound regime [3], making data compression at the network level a system-level necessity rather than an optimization. Second, the sensitivity of the SSM recurrence to quantization error was analyzed. The recurrent hidden-state update is a source of inherent numerical fragility, because an error injected at the input boundary compound across sequential timesteps rather than remaining localized. Third, standard PTQ faces an irresolvable trade-off when applied to SSM activations that contain a small fraction of large-magnitude outliers. A single uniform scale cannot simultaneously provide high resolution for the dense inlier majority and sufficient dynamic range for these outliers.

The related work survey confirmed a consistent gap. Algorithmic recipes such as Quamba have established that input outliers in tensor u dominate SSM quantization error, and dedicated accelerators such as MARCA and LightMamba have advanced efficiency. However, no existing work has physically implemented and evaluated a dynamic, flag-driven dual-scale dequantization frontend at the RTL level and assessed its area overhead through logic synthesis.

Chapter 3

Methodology

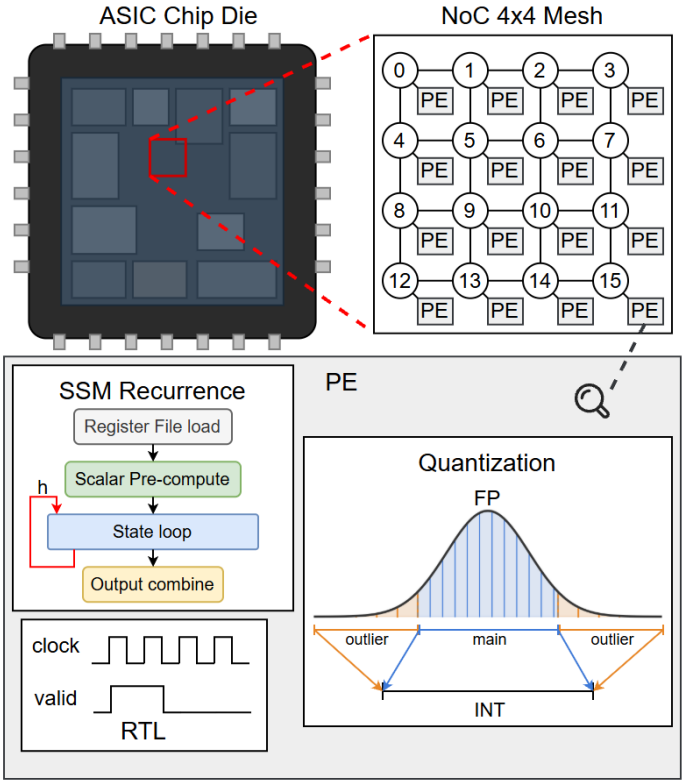

Figure 3.1: Overview of the Methods

This chapter presents the hardware-software co-design methodology employed to bridge algorithmic quantization theory and physical circuit implementation. The research is organized into four sequential phases: software data profiling, mathematical foundation of the Dual-Scale strategy,

RTL implementation and logic synthesis, and system-level NoC modeling. The four phases, which are described in the sections that follow, establish a rigorous evaluation chain from numerical accuracy to silicon area and network throughput.

The central methodological challenge of this thesis is to co-design an algorithm and a hardware datapath that are mutually constrained: the algorithm should be quantizable to INT8 with minimum accuracy loss, and the hardware should implement the dequantization logic within the existing pipeline timing budget. The recurrence of the hidden state in SSM compounds the quantization error with the timesteps, which is an algorithmic constraint. Yet there is the bandwidth of the NoC 4×4 Mesh as the hardware constraint. These two constraints make it necessary for the co-design of software and hardware in this thesis. Figure 3.1 demonstrates the full picture of this double-constraint research method.

The software-side methodology is motivated by the empirical finding in Quamba [6] that outlier activations in SSMs are concentrated exclusively in tensor u, justifying the asymmetric treatment of u relative to all other tensors. Building on this finding, the Dual-Scale quantization scheme is formulated mathematically. Then the scheme is validated via a bit-accurate `np.float32` Python model, producing the dequantization error benchmarks used as the accuracy baseline throughout this thesis.

The hardware-side methodology translates the validated quantization scheme into a synthesizable RTL implementation, evaluated against a physical 1GHz timing closure target and verified through cycle-accurate simulation. The baseline SSM compute core is implemented in SystemVerilog, with a hybrid-pipelined architecture driven by FSM. The Dual-Scale quantization changes the dequantization frontend of tensor u by adding a MUX and a flag register. RTL is then validated through bit-comparison between the SV testbench output and the Python golden model output. In Figure 3.1, the RTL waveform represents the waveform validation environment. In this phase, Synopsys Design Compiler is used in 22nm technology node at a target frequency of 1GHz, producing the hierarchical area report, timing closure report and per-token cycle count.

Finally, the standalone PE synthesis metrics are lifted into a system-level computation-communication model, quantifying the throughput benefit of INT8 payload compression across a 16-PE Constellation NoC fabric. As in Figure 3.1, a 16-PE Constellation fabric organized as a 4×4 mesh, in which each node comprises one SSM compute PE and one packet router, is the evaluation target. A Globally Asynchronous Locally Synchronous(GALS)

abstraction decouples the PE compute clock domain from the NoC routing domain, enabling independent analysis of computation and communication bottlenecks. The per-PE cycle count is combined with the per-link payload width of each quantization scheme to derive aggregate system throughput in Tokens/Second. The key comparison dimension is network bandwidth consumption: the proposed DS scheme transmits a 9-bit packed word per element, identical in width to INT9 but narrower than INT11, directly reducing the communication load on each router link. These throughput figures are then plotted against the sequence-level accuracy results to construct a Pareto frontier, identifying the operating point that maximizes accuracy per unit of network bandwidth. This analysis bridges the per-PE area and timing results of Chapter 4 with the system-level conclusions of Chapter 5.

3.1 Dual-Scale Quantization Methodology

3.1.1 The Outlier Problem and Evaluated Schemes

Per-tensor PTQ, which applies a single scaling factor uniformly across all elements of a tensor, is the dominant INT8 compression strategy for deployment efficiency [6]. However, activation profiling of the Mamba architecture reveals that this uniform approach is fundamentally ill-suited for the primary input tensor u: its distribution contains a small fraction of elements, or outliers, carrying magnitudes several times larger than the bulk of the distribution. This asymmetry is exclusive to u; the state transition parameters (A, B, C) and the step size (Δ) maintain relatively bounded, near-Gaussian distributions that can be accurately represented by a single absmax scale.

The conflict between inliers and outliers in u creates an irresolvable tension for any single-scale scheme. An absmax scale ($S = X_{max}/(2^{b-1} - 1)$) accommodates the outlier range but wastes dynamic range on inliers, coarsening their quantization step size and elevating rounding noise across the majority of values. A percentile-clipping scale ($S = X_{\alpha}/(2^{b-1} - 1)$) restores inlier resolution but saturates the outliers, injecting large truncation errors. Because u feeds directly into the recurrent hidden-state update (Equation 2.5), any quantization error in u propagates and accumulates across sequential steps, a sensitivity analyzed in Chapter 2.

This structural asymmetry motivates an asymmetric dequantization strategy: rather than elevating the entire datapath to a higher precision format, the hardware exclusively equips the u channel with a dual-scale frontend while

leaving all other tensor channels on standard single-scale INT8 paths. The key insight enabling this approach is that the inlier/outlier decision for each element of u can be pre-computed offline during calibration and encoded as a single flag bit, eliminating any on-the-fly comparison logic from the critical path. As detailed in Chapter 4, this flag-driven multiplexer is absorbed within the existing INT8-to-FP32 conversion latency, contributing zero additional pipeline stages.

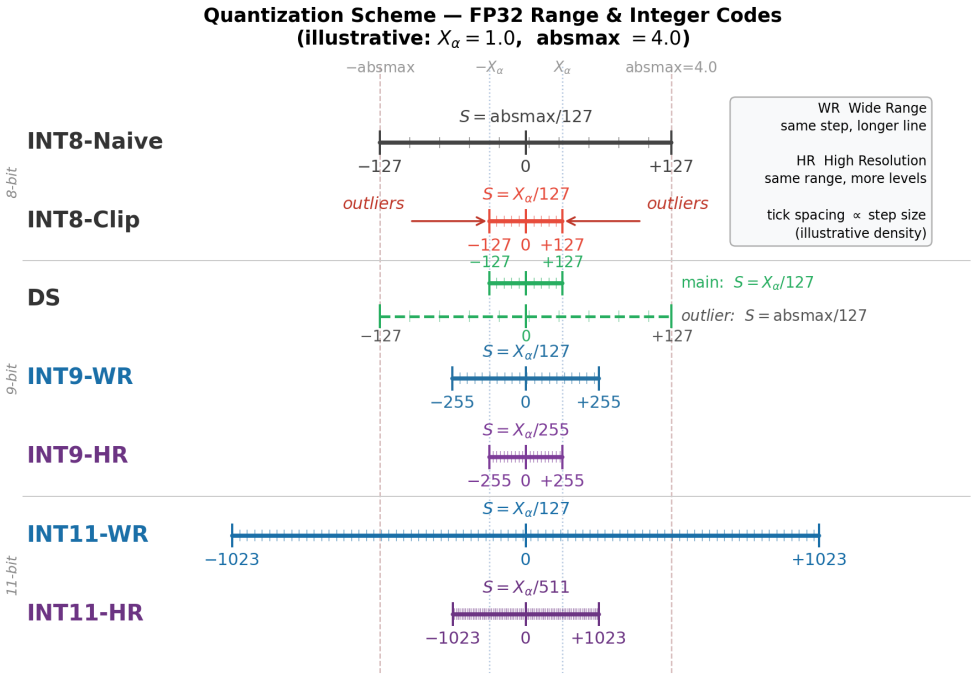

Figure 3.2: Decision Boundaries and Scale Ranges of the Evaluated Quantization Schemes

The trade-off illustrated in Figure 3.2 motivates the family of dequantization schemes evaluated in this thesis. Six schemes are compared, all sharing an identical frozen FP32 compute backend so that any difference reflects only the input representation:

- **INT8-Clip**: uniform 8-bit quantization with an α-percentile scale; recovers inlier resolution but clips outliers. The standard low-bit baseline.
- **Dual-Scale (DS)**: the proposed 9-bit scheme (8-bit payload plus a 1-bit flag) selecting per element between a percentile (inlier) scale and an absmax (outlier) scale.

- **INT9/INT11, each in Wide-Range (WR) and High-Resolution (HR)**: uniform higher-bit-width baselines constructed in this thesis as controlled comparators. All four use the same α-percentile scale base ($X_\alpha/127$) but spend their extra bits differently: a WR scheme keeps the INT8-Clip step size and widens the representable range (to $\pm 2X_\alpha$ for INT9-WR and $\pm 8X_\alpha$ for INT11-WR); an HR scheme keeps the $\pm X_\alpha$ range and refines the step instead ($X_\alpha/255$ for INT9-HR and $X_\alpha/511$ for INT11-HR), so it improves inlier resolution but still clips outliers exactly like INT8-Clip.

A seventh configuration, **INT8-Naive** (uniform 8-bit with an absmax scale), serves only as a theoretical reference in the error decomposition of Section 3.1.3. Because it wastes inlier resolution and is never a practical deployment choice, it is excluded from the result-level accuracy, robustness, and area and timing comparisons.

The following subsection formalizes the Dual-Scale strategy mathematically, and its accuracy is quantified against these baselines in Section 3.1.

3.1.2 Mathematical Formulation

To protect the outliers in tensor u without resorting to wider bit-width such as INT9 or INT11, this thesis formulates a flag-bit-driven Dual-Scale (DS) quantization scheme. The core concept decouples the dynamic range of the tensor from its fundamental quantization resolution by partitioning each element into one of two non-overlapping domains: an inlier region covering the α-percentile of the distribution, and an outlier region covering the remaining small fraction. Both scale factors are pre-computed offline during a one-time post-training calibration pass and stored as FP32 constants in the PE's configuration registers, imposing no runtime overhead on the inference datapath.

During offline calibration on a representative dataset, two distinct scale factors are derived for the tensor u. The main scale S_{main} is computed from the α-th percentile threshold (X_α), maximizing quantization step resolution within the inlier domain $[-X_\alpha, X_\alpha]$. The outlier scale, $S_{outlier}$, is derived from the absolute maximum (X_{max}), ensuring that no element is clipped regardless of magnitude:

$$S_{main} = \frac{X_\alpha}{2^{b-1} - 1} \tag{3.1}$$

$$S_{outlier} = \frac{X_{max}}{2^{b-1} - 1} \tag{3.2}$$

where b denotes the payload bit-width (e.g., 8 bits for INT8), and $2^{b-1} - 1 = 127$ is the maximum representable magnitude of a signed b-bit integer. The calibration percentile α is a tunable hyperparameter; the sensitivity of accuracy to its choice is evaluated empirically in Section 5.2.

To enable per-element scale selection, a 1-bit flag (f_i) is assigned to each element u_i during offline quantization. The flag encodes which domain the element belongs to:

$$f_i = \begin{cases} 0, & \text{if } |u_i| \leq X_\alpha \\ 1, & \text{if } |u_i| > X_\alpha \end{cases} \tag{3.3}$$

The flag is computed once during calibration and stored alongside the quantized payload. At inference time, f_i travels within the corresponding INT8 value $u_{q,i}$ as a single packed word, adding one bit to the network payload without modifying the integer representation itself.

Consequently, the network payload transmitted to the hardware accelerator consists of the compressed integer value $u_{q,i}$ accompanied by its corresponding 1-bit flag f_i, functioning as a 9-bit packed data type. Inside the PE, the dequantization reconstruction function $R(u_{q,i}, f_i)$ is mathematically defined as a conditional multiplication:

$$R(u_{q,i}, f_i) = u_{q,i} \times \begin{cases} S_{main}, & \text{if } f_i = 0 \\ S_{outlier}, & \text{if } f_i = 1 \end{cases} \tag{3.4}$$

This formulation offloads the outlier-detection burden to the offline calibration stage, leaving the on-chip critical path free of any magnitude comparison or branching logic. The pre-computed flag bit serves as a direct hardware control signal, driving a lightweight multiplexer whose propagation delay is absorbed within the existing INT8-to-FP32 conversion latency, as demonstrated in Chapter 4. Critically, this is strictly superior to simply widening the payload to INT9: a uniform INT9 scheme adds one bit of resolution globally but cannot simultaneously provide fine-grained resolution for inliers and full dynamic range for outliers, which is a trade-off that DS resolves by dedicating the extra bit exclusively to domain selection.

3.1.3 Numerical Accuracy Validation

To validate the Dual-Scale formulation prior to hardware implementation, a software-level accuracy benchmark compares DS against INT8-Naive and INT8-Clip using a bit-accurate `np.float32` Python golden model. Per-element dequantization error $|u_i - \hat{u}_i|$ is evaluated as the primary metric,

reporting both mean absolute error (MAE) and per-bin mean error across the magnitude range. The benchmark is conducted over multiple calibration seeds at a fixed percentile and bit-width, with the resulting per-element error evaluated as described in Chapter 5. The remainder of this section establishes the theoretical basis for these measurements by decomposing the dequantization error of each scheme analytically.

The three schemes differ structurally in how quantization error is distributed between the inlier and outlier regions, which can be made explicit by decomposing the MAE into two additive terms.

Define the INT8 quantizer as $Q(v,s) = clip(round(v/s), -127, 127)$, and the dequantized reconstruction under scale s as $\hat{u}_i^{(s)} = Q(u_i, s) \cdot s$. For INT8-Naive, $s_{naive} = X_{max}/127$ applies a uniform coarse step across all elements, producing pure rounding error with no clipping.

$$\text{MAE}_{\text{naive}} = \frac{1}{N} \sum_{i=1}^{N} \left| u_i - \hat{u}_i^{(s_{naive})} \right| \tag{3.5}$$

The step size is coarse because $X_{max} \gg X_\alpha$ for small-fraction large-magnitude outlier distributions.

INT8-Clip uses $s_{clip} = X_\alpha/127$, recovering finer inlier resolution at the cost of a hard clipping penalty for outliers(any element exceeding X_α):

$$\text{MAE}_{\text{clip}} = \frac{1}{N} \left[\underbrace{\sum_{|u_i| \le X_\alpha} \left| u_i - \hat{u}_i^{(s_{clip})} \right|}_{\text{fine rounding}} + \underbrace{\sum_{|u_i| > X_\alpha} |u_i - X_\alpha \, \text{sgn}(u_i)|}_{\text{clipping error}} \right] \tag{3.6}$$

The DS scheme uses $s_{main} = X_\alpha/127$ and $s_{out} = X_{max}/127$, with the flag f_i selecting the scale per element. This eliminates the clipping term entirely while preserving fine inlier resolution.

$$\text{MAE}_{\text{DS}} = \frac{1}{N} \left[\underbrace{\sum_{|u_i| \le X_\alpha} \left| u_i - \hat{u}_i^{(s_{main})} \right|}_{\text{fine rounding}} + \underbrace{\sum_{|u_i| > X_\alpha} \left| u_i - \hat{u}_i^{(s_{out})} \right|}_{\text{coarser rounding, no clipping}} \right] \tag{3.7}$$

The decomposition reveals that DS strictly dominates INT8-Clip in the outlier term, with bounded rounding versus unbounded clipping; and marginally improves over INT8-Naive in the inlier term, since $s_{main} < s_{naive}$

yields a finer quantization step within $[-X_\alpha, X_\alpha]$.

Among the higher-bit-width baselines introduced in Section 3.1.1, INT9-WR serves as the primary same-bandwidth comparator at 9-bit payload, providing a fair baseline for bandwidth-normalized evaluation.

3.2 Hardware Design Framework

The hardware evaluation framework spans two levels of abstraction: the Processing Element (PE) microarchitecture, which establishes the per-token compute latency and dequantization area overhead; and the system-level NoC integration, which situates these per-PE metrics within a realistic communication model. The RTL is synthesized targeting a 1GHz clock frequency with 22nm technology node. The two subsections below describe the architectural decisions at each level and establish the evaluation constraints that flow into Chapter 4.

3.2.1 Processing Element Microarchitecture

The SSM compute core PE is structured around a hybrid execution model that reflects the two distinct computational characters of the Mamba recurrence. Operations with no cross-iteration data dependence, specifically the input dequantization, softplus discretization and SiLU gating functions, are implemented as a fully pipelined datapath, where a new token can be accepted every cycle once the pipeline is filled. A centralized FSM controller partitions each token's computation into four sequential phases: ST_IDLE (awaiting valid input), ST_SCALAR (31 cycles, pipelined scalar pre-computation), ST_STATE_LOOP (79 cycles, micro-pipelined state recurrence), and ST_COMBINE (9 cycles, output accumulation and gating), yielding a total per-token latency of 119 cycles.

The state recurrence operation ($h_t = \bar{A}h_{t-1} + \Delta u$) introduces a loop-carried dependency: iteration n cannot begin until the FP32 accumulator write from iteration $n-1$ has resolved. Within the DesignWare FP32_Add IP requiring 3 pipeline stages at 1GHz, the Initiation Interval (II) of the micro-pipeline is set to 3 cycles. Over $d_{state} = 16$ iterations, this accounts for $16 \times 3 = 48$ active launch cycles; the remaining 31 cycles cover pipeline drain and final accumulator writeback. This cycle budget defines the temporal boundary within which the Dual-Scale dequantization frontend must complete without extending the 1ns critical path constraint.

3.2.2 System-Level NoC Integration

The system-level evaluation adopts a GALS theoretical model in which the PE compute clock domain(F_{PE}, synthesized at 1GHz) and the NoC routing domain(F_{NoC}) are treated as independent, asynchronously coupled subsystems. This decoupling is architecturally motivated: the PE's FP32 arithmetic pipeline and the NoC's link-layer arbitration operate on fundamentally different timing paths, and conflating them into a single clock domain would pessimistically constrain the achievable PE frequency. Under the GALS abstraction, cross-domain latency is modeled as a fixed synchronization overhead, leaving computation and communication throughput independently optimizable.

The target system is a 16-PE Constellation fabric organized as a 4×4 mesh, with each node comprising one SSM compute PE and one packet router. Two system-level quantities govern the design trade-off evaluated in Chapter 5: (1) the per-router area footprint(in μm^2), which establishes the silicon budget available to the PE relative to the total die; (2) the per-link packet bandwidth, which determines how the payload bit-width of each quantization scheme (9-bit DS, 9-bit INT9, 11-bit INT11) translates into network-level throughput. Plotting normalized bandwidth against quantization accuracy across all schemes yields the Pareto frontier used to identify the optimal operating point of the accelerator.

The computation-communication balance of the accelerator is characterized through two complementary analytical views: a roofline model and a bandwidth-saturation sweep. Both rest on the per-token operational intensity, defined as the number of FP32 operations executed per byte transferred between the PE and the NoC. Because each PE retains its recurrent state h in a local register bank, the state never traverses the network; only the streaming inputs and the output token are counted as PE-to-NoC traffic. The roofline places this intensity on the horizontal axis against attainable throughput, with a horizontal compute roof set by the PE throughput (FP32 operations per token divided by the per-token cycle interval at F_{PE}) and a sloped communication roof set by the NoC per-port bandwidth (flit width $\times$ routing frequency). The saturation view instead sweeps the NoC bandwidth directly and reports the achieved throughput, which rises linearly while communication-bound and clamps at the compute ceiling once compute-bound. Both models are analytical: the compute roof is anchored by an RTL-simulated per-token cycle count, and the communication roof is derived from the synthesized NoC interface parameters.

3.3 Evaluation Methodology

The evaluation framework quantifies the proposed architecture across three primary dimensions that form a sequential validation chain: numerical accuracy, hardware area and timing metrics, and system-level throughput analysis.

Numerical accuracy is evaluated at two levels. At the element level, per-element dequantization error $|u_i - \hat{u}_i|$ is measured against a bit-accurate `np.float32` Python golden model across 100 calibration seeds ($N = 1600$ data points), reporting MAE and per-bin mean error to isolate the outlier region $|u_i| > X_\alpha$. At the sequence level, RMSE and Maximum Absolute Error (MaxAE) of the final output token y are computed relative to an unquantized FP32 reference, confirming that the DS mechanism preserves fidelity in the outlier tail where INT8-Clip saturates. Both evaluations are conducted across all six quantization schemes shown in Figure 3.2 to establish a complete accuracy Pareto baseline.

Micro-architectural hardware costs are assessed from Synopsys Design Compiler synthesis reports targeting 22nm technology node at 1GHz, with the evaluation focused on synthesized area (μm^2) and timing closure. A hierarchical area breakdown isolates the synthesized gate-count area of the dequantization frontend – specifically the Scale MUX and flag register introduced by the Dual-Scale extension – from the static FP32 computational backend, attributing area differences precisely to the quantization scheme under test. Timing closure analysis verifies that the critical path remains within the 1ns constraint; any violation would necessitate an additional pipeline register at the MUX output, which would be reported as a one-cycle latency penalty in the per-token cycle count.

System-level performance is evaluated by integrating the per-PE cycle count with per-link bandwidth parameters derived from the GALS NoC model to compute aggregate throughput in Tokens/Second. The primary analysis instrument is a two-dimensional Pareto curve plotting normalized network bandwidth consumption – parameterized over INT8-DS, INT9-WR, INT9-HR, INT11-WR, INT11-HR – against the corresponding sequence-level accuracy, identifying the scheme that maximizes accuracy per unit bandwidth. INT8-Naive is excluded as it serves only as a theoretical reference. The Pareto analysis constitutes the principal evidence for the efficiency claim of the Dual-Scale approach at the system level.

The validity of all experimental comparison is based on two methodological safeguards. First, functional correctness is established through a bit-

accurate Python golden model that defines the intended numerical behavior of the SystemVerilog datapath including quantization rounding, integer overflow handling, and IEEE 754 FP32 arithmetic, without abstracting these effects as in standard PyTorch inference. RTL simulation outputs are cross-validated element-by-element against golden model arrays across 1000 random seeds prior to synthesis; exact element-wise agreement between the two is required before any synthesis result is accepted. Second, the validity of area and timing comparisons across quantization schemes is enforced by a parameterized, top-down synthesis methodology in which the shared FP32 backend is structurally frozen between experiments. Only the `INPUT_WIDTH` parameter of the `INTx_to_FP32_DW` conversion module is varied, ensuring that any difference reported in the synthesized area or timing is attributable exclusively to the input representation width and not to EDA-driven restructuring of the shared arithmetic units.

Chapter 4

Implementation Details

This chapter presents the Register Transfer Level(RTL) implementation of the SSM accelerator, progressing from the baseline compute core to Dual-Scale extension and the system-level integration model. Section 4.1 describes the FSM-controlled hybrid pipeline architecture, detailing the scalar pre-computation path, the micro-pipelined state recurrence, and the timing constraints that govern the complete 119-cycle per-token budget. Section 4.3 introduces the Dual-Scale modification, isolating the precise structural change to the dequantization frontend and characterizing its area cost relative to the frozen FP32 backend. Section 4.4 formalizes the parameterized synthesis methodology used to ensure fair comparisons across bit-width configurations. Finally, Section 4.5 defines the analytical model that translates per-PE cycle counts into system-level throughput within the 16-PE Constellation NoC.

4.1 Baseline SSM Compute Core

The top-level coordination of the baseline SSM computation core relies on a centralized hardware finite state machine(FSM) that partitions each token's computation into four temporally ordered phases, as illustrated in Figure 4.1. On reset, the PE occupies ST_IDLE and asserts `ready_out`, signaling availability to the network interface. Upon receiving a valid input handshake `valid_in`=1, the FSM transitions immediately into ST_SCALAR, where all channel-wise dequantization, softplus discretization and SiLU gating of z are executed as a fully pipelined, dependency-free datapath.

After a fixed 31-cycle interval (`CYC_SCALAR_DONE` = 31) timed by the internal `fsm_timer` counter, the FSM advances to ST_STATE_LOOP to process the recurrent state update across all $d_{state} = 16$ dimensions, spending

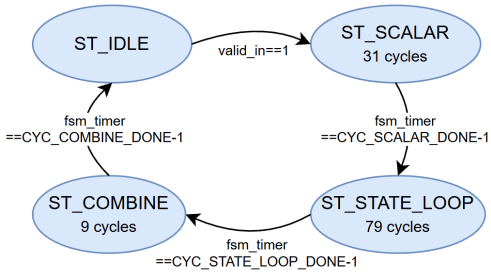

Figure 4.1: Finite-State-Machine Control Flow of the SSM Compute Core

79 cycles (`CYC_STATE_DONE` = 79). A final 9-cycle ST_COMBINE state phase(`CYC_COMBINE_DONE` = 9) accumulates the output projection and applies optional SiLU gating before the FSM returns to ST_IDLE, yielding a deterministic per-token latency of $31 + 79 + 9 = 119$ clock cycles.

The dataflow controlled by this FSM is shown in Figure 4.2. During ST_SCALAR, three parallel paths process the streaming inputs concurrently: the Δ path applies dequantization, adds `delta_bias`, and evaluates softplus to produce `dt_sp`; the u path dequantizes and scales u to FP32, then multiplies by `dt_sp` and `D` to form `dt_u` and `Du` separately; and the `z` path computes SiLU(`z`) (`silu_z`) for output gating. All four scalar results (`dt_sp`, `dt_u`, `Du` and `silu_z`) are latched simultaneously at cycle 31 into dedicated holding registers, from which they are consumed during subsequent phases. The state loop fires one iteration every 3 cycles (`loop_fire` = `ii_counter` == 0), serializing 16 state-dimension updates through the shared FP32 datapath. The DS variant inserts the `u_flag`-controlled Scale MUX exclusively in the u path of ST_SCALAR, highlighted in the orange sidebar of Figure 4.2; no other path is modified.

4.1.1 Micro-Pipelined State Recurrence

The arithmetic organization of the state-update phase is shown in Figure 4.3, which presents a conceptual three-stage view of the micro-pipeline that executes once per state dimension per token.

Stage 1 handles parameter fetch and pre-computation. INT8 values for $A[n]$, $B[n]$, and $C[n]$ are read from the local register file indexed by the `loop_idx`, converted to FP32, and multiplied by their respective scale factors to produce `A_fp32`, `B_fp32`, and `C_fp32`. The discretized state-transition coefficient $\bar{A}_n = exp(-exp(A_n) \cdot \Delta_{sp})$ is computed through a two-stage exponential chain (depth 23 cycles from launch). Concurrently,

`BC_product`$= B_n \cdot C_n$ is computed along the shorter 9-cycle path, and a 17-stage alignment register synchronizes it with Stage 2's output.

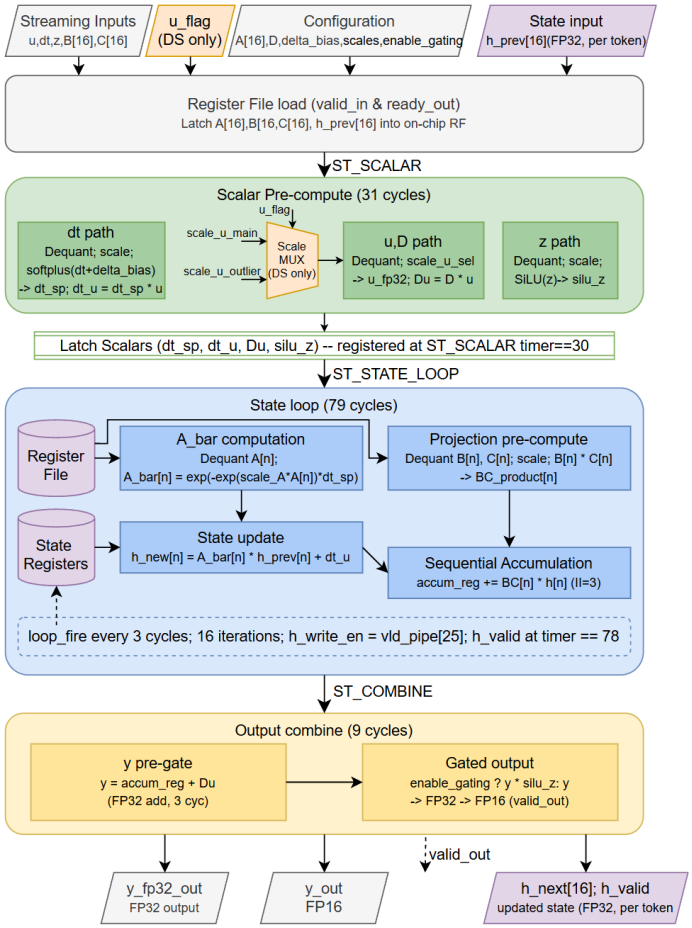

Figure 4.2: Overall Hardware Design Framework of the SSM Compute Core

Stage 2 executes the primary recurrence via a DesignWare `FP32_MAC`:$h_{new} = \bar{A}_n \times h_{prev}[n] + \Delta u$, where $h_{prev}[n]$ is the previous hidden state read from the state register file after a 23-stage delay alignment, and Δu is the latched scalar `dt_u` from ST_SCALAR. The updated state $h_{new}[n]$ is written back to the register file at `vld_pipe[25]`.

Stage 3 computes the output projection $BCh[n] = B_n C_n h_{new}[n]$ and accumulates the result into `accum_reg` via a `FP32_Add` module. Because this adder introduces a 3-cycle pipeline depth, and each iteration reads `accum_reg` as an operand, a loop-carried dependency exists: each new iteration must be issued no sooner than 3 cycles after the preceding one, enforcing $II = 3$. A bypass path `acc_vld_pipe[2]` forwards the in-flight

adder output when the register has not yet been updated, resolving the read-after-write hazard without stalling.

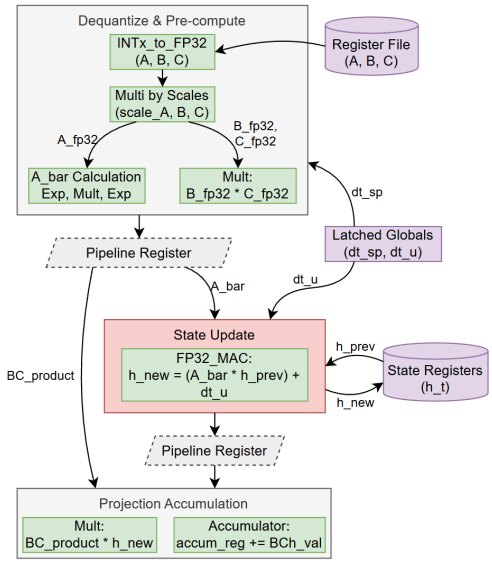

Figure 4.3: Datapath of the ST_STATE_LOOP FSM State

4.2 Complete RTL Architecture

Figure 4.4 presents the complete RTL datapath of the baseline SSM compute core, organized into three spatially distinct regions corresponding to the three active FSM phases. The FSM controller on the left governs all state transitions and generates the control signals that coordinate dataflow across the regions. Below paragraphs trace the datapath from input to output, following the top-to-bottom spatial layout of the figure.

The upper region of Figure 4.4 demonstrates the ST_SCALAR datapath, which processes all token-level scalar quantities in parallel during the first 31 cycles. Each input channel enters through a dedicated DesignWare `INT8_to_FP32` converter followed by a `FP32_Mult` scale stage. The Δ path adds `delta_bias_fp32` to the scaled result and computes softplus, which determines the 31-cycle scalar timing budget. The output of softplus block `dt_sp` is immediately written into the Latched Scalar register bank under the condition `current_state==ST_SCALAR & & fsm_timer == CYC_SCALAR_DONE-1`. The u and D paths pass through 22-stage pipeline alignment registers (`U_D_DELAY=22`) to arrive at cycle 27 in

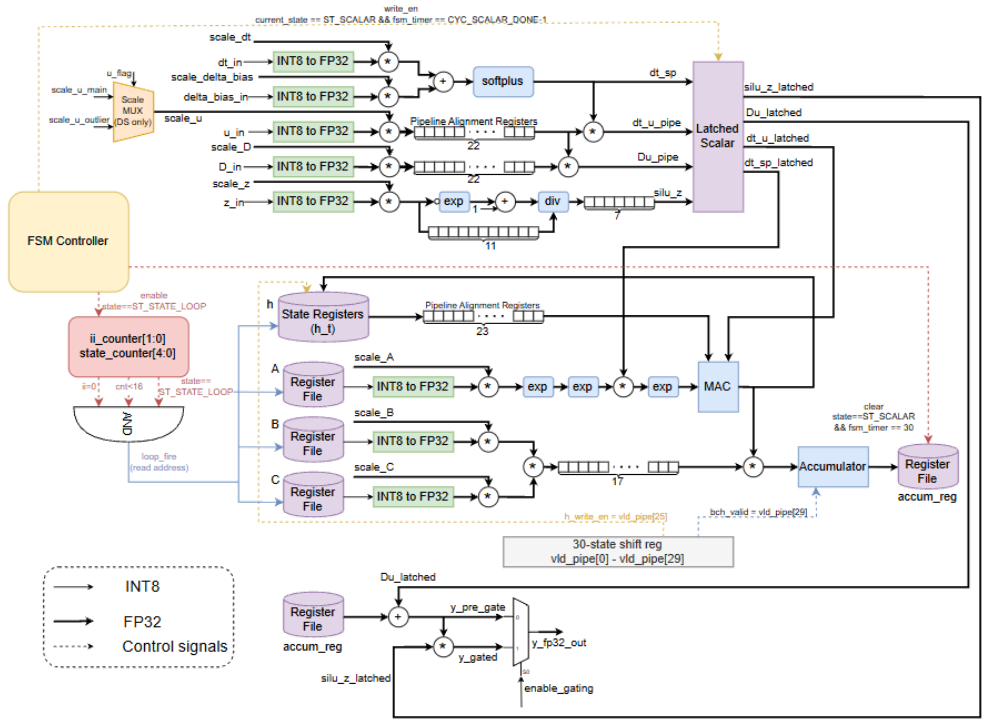

Figure 4.4: Detailed RTL Architecture of the SSM Compute Core

synchrony with `dt_sp`, at which point the two `FP32_Mult` stages compute `dt_u` and `Du` respectively. The z path implements SiLU via an exp-add-div chain of 11-cycle datapath followed by a 7-stage post-alignment shift register (`SILU_DELAY=7`), with `silu_z` completing at cycle 31. All four results – `dt_sp`, `dt_u`, `Du`, and `silu_z` – are latched simultaneously into the Latched Scalar register bank at the end of ST_SCALAR and held for the duration of the subsequent states.

The central region implements the ST_STATE_LOOP datapath, which is activated when the FSM transitions out of ST_SCALAR. The loop execution is governed by a two-level counter structure on the left: `ii_counter[1:0]` cycles through values 0-2 to enforce the $II = 3$ initiation interval, and `state_counter[4:0]` tracks the current state dimension index from 0 to 15. A combinational AND gate fires *loop_fire* – which doubles as the register file read address – only when `ii_counter==0` and `state_counter<16` and `state==ST_STATE_LOOP`, ensuring exactly one iteration launch per 3-cycle slot. On each `loop_fire`, the A, B and C register files supply their INT8 values for dimension n; these are converted to FP32 and scaled before diverging into two parallel chains. The A chain computes $\bar{A}_n =$

$exp(-exp(A_n) \cdot \Delta_{sp})$ through a four-stage sequence of `exp-neg-mult-exp` operations, feeding into the `FP32_MAC` together with `h_prev[n]` drawn from the State Registers via a 23-stage pipeline alignment register. The B and C chain computes $B_n \cdot C_n$ via a single `FP32_Mult`, and a 17-stage alignment register synchronizes this product with the `FP32_Mult` and `Accumulator` stages. The accumulated result is stored in `accum_reg`, which is cleared at `state==ST_SCALAR & & fsm_timer==30` to prepare for the next token.

Coordinating the delayed write-back of h_{new} and the gating of the accumulator requires precise cycle-level control that cannot be derived directly from the FSM state alone, since the relevant operations complete many cycles after `loop_fire`. This is handled by a 30-stage shift register, `vld_pipe[0:29]`, which propagates a validity token in lockstep with the pipeline depth. `vld_pipe[0]` is loaded with `loop_fire` on each iteration launch; the token arrives at `vld_pipe[25]` exactly when $h_{new}[n]$ is available at the Stage Register write port, asserting `h_write_en`. When the BCh product is ready at the Accumulator input, `vld_pipe[29]` asserts `bch_valid`. This distributed validity propagation mechanism decouples the control timing from the FSM counter arithmetic, eliminating the need for additional per-stage counters and ensuring hazard-free operation across all 16 iterations.

The lower region of Figure 4.4 implements the ST_COMBINE datapath, which assembles the final output from the quantities latched during the preceding two states. The Register File holding `accum_reg` supplies the accumulated projection sum, which is added to `Du_latched` via a `FP32_Add` to form `y_pre_gate`. An output multiplexer then selects between `y_pre_gate` and `y_gated = y_pre_gate × silu_z_latched` under the `enable_gating` control signal, producing `y_fp32_out`. This gating mechanism implements the selective output projection of the Mamba architecture, where the z channel modulates the output amplitude token-by-token. The full ST_COMBINE state completes in 9 cycles, accounting for the `FP32_Add`(3 cycles), the `FP32_Mult` for gating (4 cycles), and the final output register stage (2 cycles).

4.3 Dual-Scale Hardware Implementation

The Dual-Scale (DS) extension introduces a single structural modification to the baseline PE: the u channel dequantization frontend is augmented with a 2-to-1 scale multiplexer driven by a 1-bit `u_flag` input. All other

channels, A, B, C, Δ, D, z, retain their original single-scale INT8 paths without modification, and the entire FP32 arithmetic backend is structurally frozen. This confinement is a deliberate design choice: since outlier activations in the Mamba architecture are empirically concentrated in u [6], extending multi-scale dequantization to other tensors would incur area cost without accuracy benefit.

The critical implementation insight is that this multiplexer imposes zero additional pipeline stages. As established in Section 3.2, the INT8-to-FP32 conversion and scale multiplication together consume 5 cycles before the softplus chain needs their output. The 2-to-1 MUX selecting between `scale_main` and `scale_outlier`, which are both pre-loaded into FP32 configuration registers at layer initialization, is a purely combinational operation whose propagation delay is absorbed within this existing 5-cycle budget. By construction, the MUX therefore adds no pipeline stage to the dequantization frontend, and whether this holds under synthesis is verified in Chapter 5. This zero-latency-cost property is precisely the architectural advantage that an ASIC implementation provides over GPU-based quantization: CUDA thread-block execution imposes a uniform dequantization kernel across all tensor elements, precluding per-element scale selection without runtime branching overhead; the ASIC MUX achieves the same per-element selectivity entirely in combinational hardware, parallel with the existing converter delay.

4.4 Parametrized Design Space for Evaluation

To ensure a fair and reproducible evaluation across quantization configurations, the hardware architecture adopts a strictly parametrized input front-end. The top-level parameter INPUT_WIDTH, defined in the module header of ssm_compute_core_fsm and propagated to all submodules, governs the bit width of every signed integer input port: `u_in`, `dt_in`, `B_in`, `C_in`, `z_in`, `A_in`, `D_in`, and delta_bias_in are all declared as logic signed [INPUT_WIDTH-1:0]. The INTx_to_FP32_DW conversion primitive reads this parameter directly to size its internal sign-extension and binary-to-IEEE-754 conversion logic. This parameterization ensures that the input conversion overhead scales strictly with bit-width, without requiring any manual re-instantiation of the conversion module.

Critically, all FP32 arithmetic units downstream of the conversion stage

— the DesignWare Mult, Add, MAC, Exp, Softplus, and Div instances — are declared with fixed DATA_WIDTH of 32 bits and remain completely unchanged across synthesis runs. This means that when Design Compiler synthesizes the INT9 or INT11 variants, the only logic that differs from the INT8 baseline is the front-end conversion circuitry and the input port widths. The FP32 backend is structurally frozen, so the EDA tool cannot restructure or re-optimize those paths differently between runs. Consequently, any area difference reported between configurations is strictly attributable to the cost of the wider input representation, and the comparison provides a direct measurement of the silicon overhead of expanding uniform quantization bit width relative to the lightweight multiplexer approach of the DS design. The frozen backend constraint is enforced at the synthesis script level, ensuring that Design Compiler treats the shared FP32 units as structurally immutable across all runs. The reported area figures in Chapter 5 are therefore directly comparable, with any difference attributable solely to the input representation width.

4.5 System-Level Integration Modeling

The system-level model integrates the synthesized per-PE cycle count with the NoC fabric parameters to derive aggregate throughput, treating the 16-PE Constellation as a GALS system in which computation and communication are independently bounded. The evaluation target is a 4×4 mesh in which each node comprises one SSM compute PE and one packet router, totaling 16 compute nodes and 16 routers. A GALS abstraction decouples the PE clock domain ($F_{PE} = 1GHz$, synthesized) from the NoC routing domain, with cross-domain latency treated as a fixed synchronization overhead. The per-PE compute time is $T_{compute} = 119$ cycles$/F_{PE} = 119$ ns; the communication time per token is $T_{comm} =$ payload_bits/link_bandwidth. The system bottleneck is determined by $T_{token} = \max(T_{compute}, T_{comm})$, which identifies whether the system operates in a compute-bound or bandwidth-bound regime.

System throughput in Tokens/Second is given by Throughput $= F_{PE}/T_{token}$. The payload width varies by scheme: INT8-Clip transmits 8 bits per element; DS and both INT9 variants transmit 9 bits (INT8 payload + `u_flag` for DS, or a full 9-bit integer for INT9); INT11 variants transmit 11 bits. Normalizing against the INT8 baseline gives $B_{norm} = w_{scheme}/8$, yielding DS = INT9 = 1.125× and INT11 = 1.375×. Critically, DS and INT9 share identical B_{norm}, so any accuracy difference between them demonstrated in Section 5.2 is a pure algorithmic advantage at zero additional bandwidth

cost. The per-token compute and communication times defined here directly parameterize the roofline and bandwidth-saturation analyses in Section 5.6.

Chapter 5

Experiment Results

This chapter evaluates the proposed Dual-Scale (DS) hardware architecture against INT8-Clip and uniform higher bit-width formats across four dimensions: single-step quantization accuracy, recurrent error propagation, hyperparameter robustness, and hardware area and timing. Section 5.2 establishes single-step accuracy at $\alpha = 0.99$, demonstrating that DS suppresses peak outlier errors while matching INT9-WR and INT11-WR in RMSE at 9-bit payload cost. Section 5.3 tracks error accumulation across 64-step SSM recurrences. Section 5.4 evaluates robustness across $\alpha \in 0.90, 0.95, 0.99, 0.9995$. Section 5.5 presents synthesis-level area and timing results, and Section 5.6 translates these into system-level throughput and Pareto optimality within the 16-PE NoC.

5.1 Experimental Setup

Before presenting the detailed evaluation of the proposed architecture, this section outlines the specific experimental configurations, environmental parameters, and toolchains utilized to generate the results. The evaluation spans both software-level algorithmic fidelity and hardware-level physical implementation. To ensure the reproducibility and fairness of all comparative analyses, the hardware baselines and the proposed Dual-Scale architecture were evaluated under identical environmental constraints.

A comprehensive summary of the key experimental parameters is provided in Table 5.1.

The numerical evaluations use a bit-accurate `np.float32` Python golden model as described in Section 3.3. Activation statistics follow the Mamba-130M Quamba inference distributions [6], with $d_{state} = 16$. Unless stated

Table 5.1: Summary of Experimental Parameters

Parameter	**Value**
Algorithmic Configuration	
SSM State Dimension (d_{state})	16
Calibration Percentile (α)	0.90, 0.95, 0.99, 0.9995
Evaluated Quantization Schemes	INT8-Clip, DS, INT9-WR/HR, INT11-WR/HR
Statistical Seeds	200 (accuracy) / 1000 (error propagation)
Hardware Synthesis	
Technology Node	22nm
Target Clock Frequency	1.0 GHz
System Model	
NoC Topology	4×4 Constellation Mesh
Total Nodes	16 PE + 16 Routers
Clock Architecture	GALS (F_{PE} = 1 GHz)

otherwise, Section 5.2 uses $\alpha = 0.99$ with 200 seeds; Section 5.3 uses 1000 seeds over 64 timesteps with $\alpha \in 0.95, 0.99, 0.9995$; Section 5.4 uses 200 seeds with $\alpha \in 0.90, 0.95, 0.99, 0.9995$. Hardware synthesis follows the parameterized methodology of Section 4.4 under 22nm technology node at 1.0 GHz using Synopsys Design Compiler. The DS scheme is benchmarked against INT8-Clip, INT9-WR, INT9-HR, INT11-WR and INT11-HR across all experiments.

5.2 Numerical Accuracy Evaluation

To isolate the numerical degradation caused strictly by data compression, the hardware outputs of various quantization schemes were benchmarked against an ideal, unquantized FP32 baseline. The following evaluation utilizes a fixed, conservative calibration percentile of $\alpha = 0.99$ over 200 random seeds to ensure statistical significance. This section proves that the proposed hardware-efficient Dual-Scale approach matches or exceeds the error-management capabilities of significantly higher bit-width formats.

The experimental results in Figure 5.1 confirm the theoretical decomposition. The upper panel shows the magnitude distribution of u, with the inlier

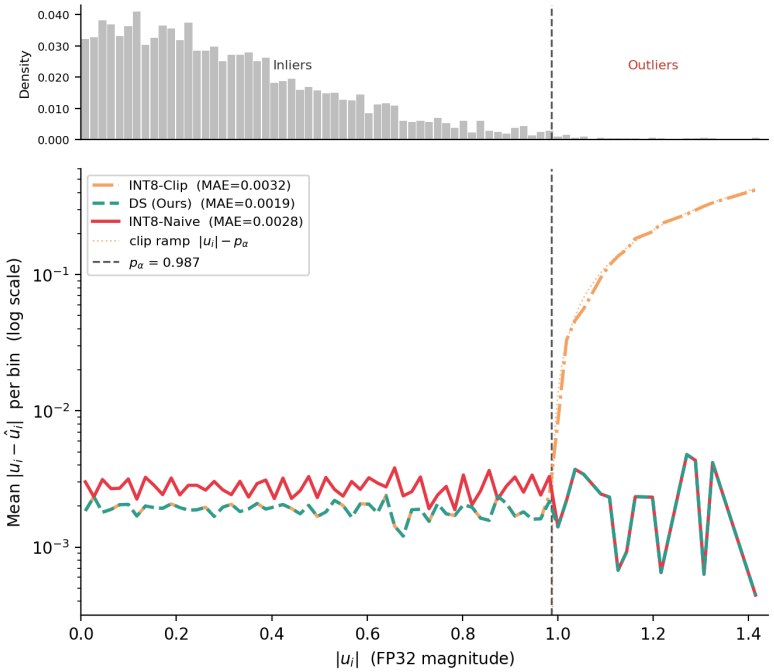

Figure 5.1: Per-Element Quantization Error Distribution at $\alpha = 0.99$ with 200 seeds

and outlier regions separated by the calibration threshold $X_\alpha = 0.987$; the heavy tail beyond X_α contains only 1% elements. The lower panel reports the per-bin mean dequantization error on a logarithmic scale for the three INT8-equivalent schemes. DS achieves the lowest overall error (MAE= 0.0019), compared to 0.0028 for INT8-Naive and 0.0032 for INT8-Clip.

The two panels reveal that DS adopts the better-performing scale in each region. Within the inlier region, both DS and INT8-Clip apply the fine percentile-derived scale ($s_{main} = X_\alpha/127$, which lies below INT8-Naive whose coarser absmax scale enlarges the rounding step. Within the outlier region, the relationship reverses: both DS and INT8-Naive apply the absmax-derived scale that represents large magnitudes without clipping, whereas INT8-Clip diverges sharply above X_α, tracking the linear clip-ramp reference $|u_i| - X_\alpha$. This is the exact failure mode that injects bias into the recurrent hidden state h_t. By matching INT8-Clip on inliers and INT8-Naive on outliers, DS combines the lower error of each scheme in the region where that scheme is strong, achieving the best result across the full magnitude range.

The bar charts presented in Figure 5.2 and Figure 5.3 provide a comprehensive summary of the quantization accuracy across two primary

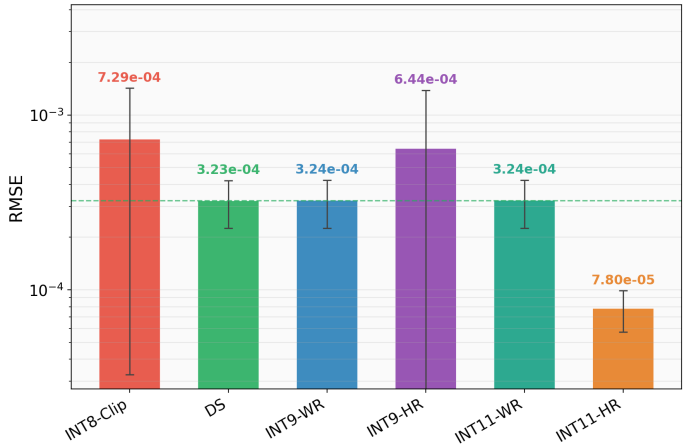

Figure 5.2: Accuracy Comparison across Quantization Schemes in terms of RMSE at $\alpha = 0.99$

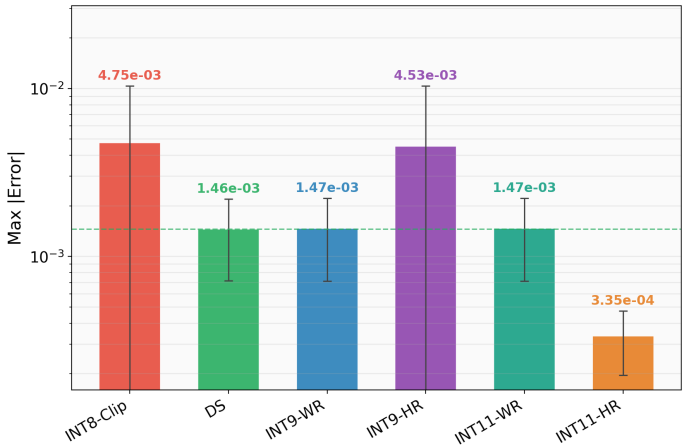

Figure 5.3: Accuracy Comparison across Quantization Schemes in terms of MaxAE at $\alpha = 0.99$

metrics: Root Mean Square Error (RMSE) and Maximum Absolute Error (MaxAE). The data contrasts the baseline INT8 with clipping (INT8-Clip), the proposed Dual-Scale (DS) configuration (which operates as a 9-bit packed payload), and standard uniform higher-precision formats including 9-bit and 11-bit configurations in both Wide Range (WR) and High Resolution (HR) modes. The results clearly illustrate that INT8-Clip suffers from a severe maximum error penalty, reaching a MaxAE of 4.75×10^{-3}. In contrast, the DS scheme successfully suppresses this MaxAE to 1.46×10^{-3}, a 69% reduction relative to INT8-Clip. DS matches both INT9-WR and INT11-WR in MaxAE while using a 9-bit rather than 11-bit payload. The advantage of DS is therefore not higher absolute accuracy, but achieving the same peak-error suppression at minimum payload, where the uniform INT8-Clip baseline at 1 bit less collapses. INT11-HR attains the lowest RMSE in this single-step setting (7.80×10^{-5}, but it does so by concentrating all resolution within a narrow inlier range, at the cost of dynamic range and an 11-bit payload. This resolution-range trade-off causes its accuracy to collapse once the calibration percentile tightens, as analyzed in Sections 5.3 and 5.4. Among the schemes that remain stable across calibration settings, DS attains the best accuracy at the smallest payload.

Analyzing these statistical metrics reveals the inefficiency of uniform bit-width scaling for outlier suppression. Allocating an extra bit to transition from INT8-Clip to INT9-WR lowers the maximum absolute error from 4.75×10^{-3} to 1.47×10^{-3}, because the wider integer grid stops clipping the outliers. At this point the MaxAE has effectively saturated: the INT9-WR (1.47×10^{-3}), the INT11-WR (1.47×10^{-3}) and DS (1.46×10^{-3}) are all indistinguishable, since none of them clips. What separates their performance is payload width and not accuracy. The DS scheme reaches this same floor by spending its extra bit as a per-element scale-selection control flag rather than as a uniform numerical range. The value of DS is therefore not a lower MaxAE than the wide-range formats, but reaching their shared accuracy floor at the minimum payload.

Figure 5.4 illustrates the Cumulative Distribution Function (CDF) of the relative quantization errors, plotted on a logarithmic scale. This graph traces the percentage of the total processed samples that fall below a specific relative error threshold. The red curve representing the baseline INT8-Clip formulation is noticeably shifted to the right, indicating that a noticeable portion of the data experiences relative degradation compared to the ideal FP32 values. When observing the green curve representing the DS architecture, it shifts to the left of INT8-Clip across the bulk of the

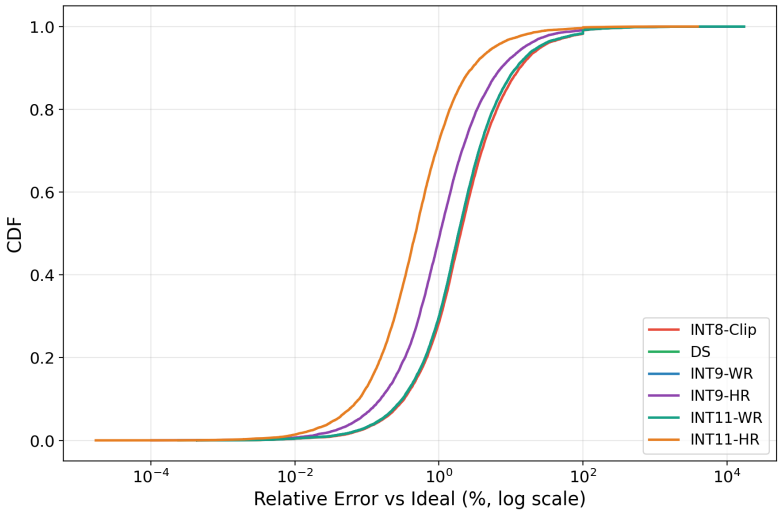

Figure 5.4: Cumulative distribution function of relative quantization errors at $\alpha = 0.99$

distribution and tracks the wide-range INT9-WR and INT11-WR closely. Across most of the distribution the DS, INT9-WR and INT11-WR curves nearly coincide, confirming that the three schemes handle the dense inlier majority with comparable relative accuracy. The distinction does not appear prominently on this relative-error axis because the outliers that DS protects, although they carry the largest absolute errors under INT8-Clip, represent only about 1% of elements and do not dominate a relative-error CDF. Their effect is instead captured by the MaxAE metric in Figure 5.3 and by the residual-error analysis in Figure 5.6. INT9-HR lies slightly to the left, while INT11-HR sits furthest left, but at the cost of 2.84% area overhead.

The CDF plot therefore confirms that DS preserves the inlier-region fidelity of the percentile-clipped schemes while additionally protecting the outlier tail, rather than trading one for the other. By setting the calibration percentile to $\alpha = 0.99$, the main scale is tightly optimized to provide maximum resolution for the vast majority of the data points, which explains why the lower-tail of the CDF performs so well. Simultaneously, the 1-bit flag mechanism guarantees that the remaining 1% of extreme values do not suffer catastrophic truncation. This curve definitively proves that dynamic, runtime scale-switching in hardware is a vastly superior strategy for managing small-fraction large-magnitude outlier distributions compared to static, linear resolution upgrades, allowing the system to emulate the performance of a wide-range 11-bit bus using significantly fewer physical wires.

The histograms provided in Figure 5.5 offer a granular view of the error frequency, displaying the count of samples against their absolute and relative error magnitudes. The topleft two subplots, representing the INT8-Clip baseline, exhibits a distinctly problematic shape: while the majority of errors are clustered near zero, a sparse but extended ”long tail” stretches far along the x-axis, with absolute errors reaching beyond 0.0025. This tail is the direct visual manifestation of outlier clipping. In the top-middle two subplots, the Dual-Scale (DS) histogram demonstrates a profound structural correction. The entire distribution is concentrated near the origin, with the extreme error tail removed. While the High Resolution (HR) configurations (INT9-HR and INT11-HR) show extremely tight distributions, they do so at the cost of severely restricted dynamic ranges, which will later be shown to fail catastrophically under different hyperparameters.

From an analytical perspective, these histograms expose the core vulnerability of standard quantization when applied to State Space Models. The presence of the long error tail in the INT8-Clip data indicates that while 99% of the tensor is quantized with acceptable precision, the most structurally significant values—the outliers—are being arbitrarily crushed by the rigid hardware boundaries. The DS hardware directly intercepts this failure mechanism. Because the flag bit dynamically shifts the arithmetic mapping, the outlier values bypass the saturated main scale and are instead mapped against a dedicated, wide-range grid. Consequently, the maximum error boundary is artificially clamped, preventing the generation of the extreme error magnitudes that populate the tail of the baseline histogram. This localized structural repair is essential for maintaining the stability of recurrent matrices.

Figure 5.6 provides the most definitive visual evidence of the outlier preservation mechanism, displaying the raw residual error for every single output sample across the test set. The x-axis represents the sample index sorted ascendingly by the magnitude of the ideal FP32 value, while the y-axis shows the deviation (quantized output minus ideal output). In the INT8-Clip scatter plot (top left), a ”Severe Clipping Region” clearly emerges on the far right side of the graph. As the ideal magnitude increases, the quantization error diverges sharply and linearly, proving that these large values exceed the maximum representable integer code and are simply truncated. Conversely, the Dual-Scale scatter plot (top middle) demonstrates stable and bounded behavior in this exact same high-magnitude region. The green data points show no structural divergence, confirming that ”Outlier Fidelity” is fully preserved even at the extreme edges of the distribution. Note that the per-sample residuals shown here are element-level deviations on the input tensor u (with

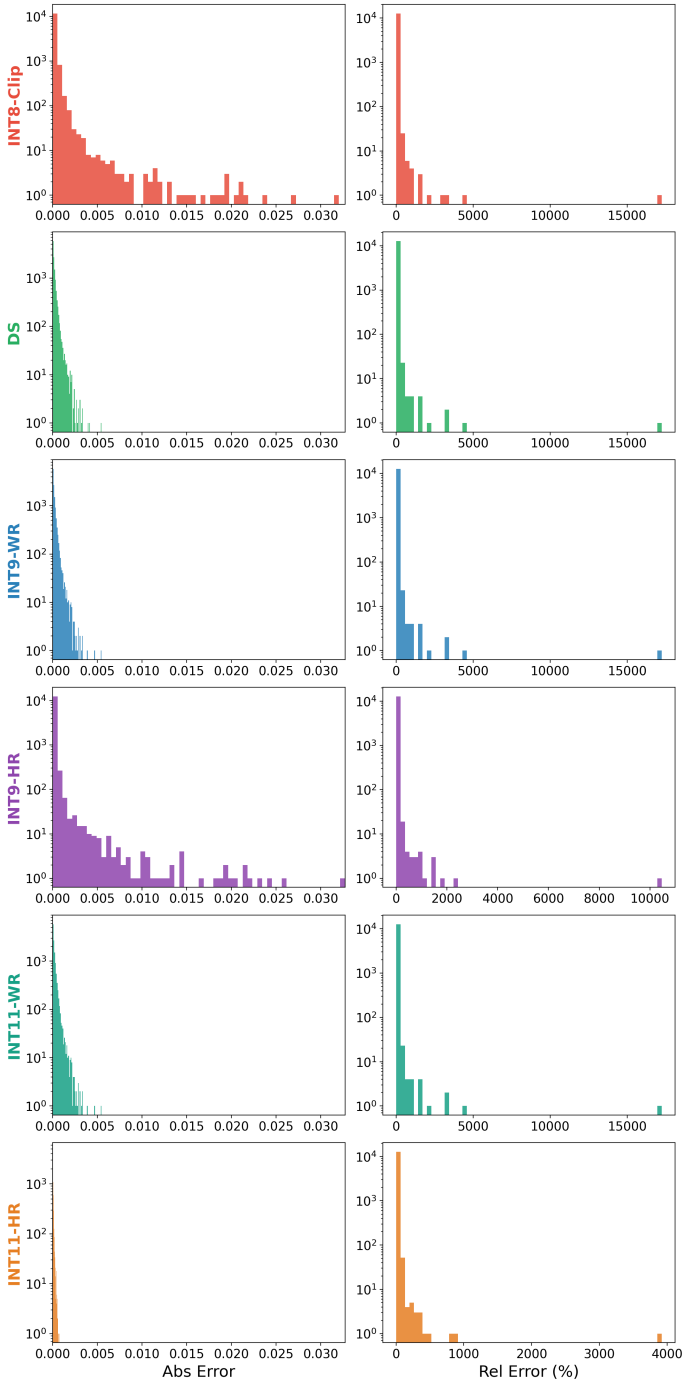

Figure 5.5: Absolute and Relative Error Distributions across Evaluated Quantization Schemes

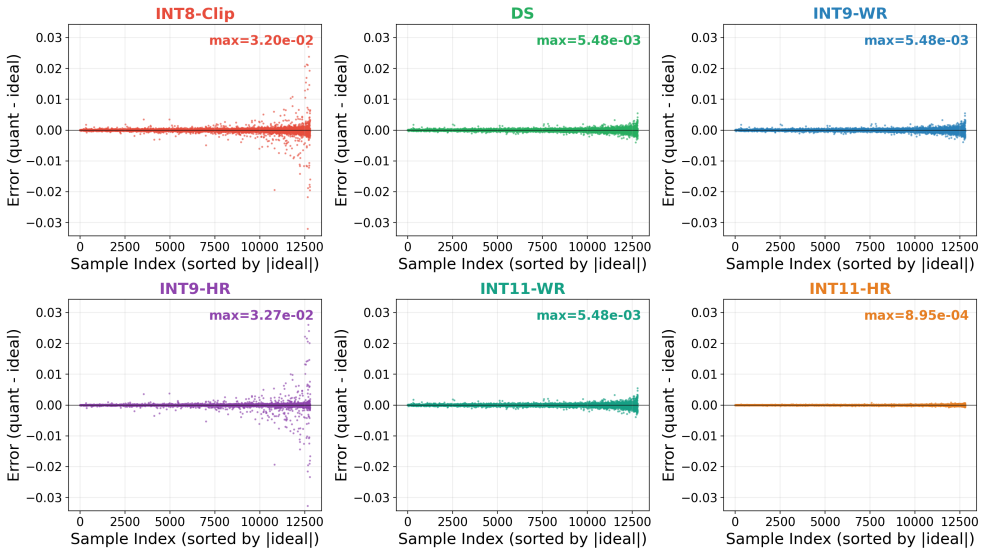

Figure 5.6: Residual Error per Sample sorted by Ideal Magnitude

a peak near 2.30×10^{-3} for DS), which are distinct from the sequence-level MaxAE on the output token y reported in Section 5.2; the former characterizes the dequantization stage in isolation, the latter the full per-token computation.

This scatter plot is critical for understanding why the Dual-Scale hardware architecture succeeds where standard uniform approaches fail. The linear divergence seen in the INT8-Clip graph is highly destructive to neural network inference; it implies that the network's strongest activation signals are being disproportionately corrupted and muted. If the hardware is unable to transmit these peak signals accurately, the downstream recurrence and output projection lose their primary contextual triggers. The DS scatter plot proves that the 1-bit multiplexer logic implemented in the PE frontend successfully captures these peak signals and routes them through the correct floating-point reconstruction path. By ensuring that the residual errors remain tightly clustered horizontally regardless of the input magnitude, the hardware guarantees that the foundational mathematics of the SSM are executed with high fidelity, resolving the fundamental bottleneck of static integer quantization.

5.3 Error Propagation in SSMs

While single-step numerical accuracy provides a foundational baseline, evaluating State Space Models (SSMs) requires tracking error dynamics over

continuous sequences. Unlike feed-forward networks where quantization noise remains localized within a specific layer, the recurrent architecture of SSMs causes input errors to accumulate over time. The hidden state h_t is updated via the recurrence relation $h_{t,n} = \bar{A}h_{t-1,n} + \Delta_{sp}u$. Consequently, quantization noise injected at the input boundary (u) is recursively multiplied by the transition matrix $\bar{A}$, creating a compounding effect. To evaluate this phenomenon, we simulate 64 continuous timesteps across 1000 random seeds, analyzing the error propagation under different clipping percentiles ($\alpha \in \{0.95, 0.99, 0.9995\}$).

Figure 5.7 tracks the mean L2 error of the hidden state vector h_t relative to the FP32 ideal execution. Across all evaluated configurations, a consistent pattern emerges: the error experiences a rapid initial ascent during the first 5 to 10 timesteps, followed by a transition into a steady-state plateau or a slow, continuous drift. This behavior reflects the "leaky integrator" mechanics of the SSM. The decay properties of the $\bar{A}$ matrix bound the infinite accumulation of past errors, establishing an equilibrium with the new quantization noise continually injected by the input projection $\Delta_{sp}u$.

Analyzing the upper and middle panels ($\alpha = 0.95$ and $\alpha = 0.99$), we observe regimes where the outlier clipping threshold is relatively tight. In these scenarios, the INT8-Clip baseline (red line) and the INT9-HR format (purple line) exhibit the highest levels of error accumulation, with the $\alpha = 0.95$ case reaching an L2 error near 0.08. This pronounced degradation occurs because structurally important outliers are repeatedly truncated at every timestep, injecting a consistent directional bias into the hidden state computation. Under these conditions, the proposed Dual-Scale (DS) architecture demonstrates a clear advantage. By utilizing the secondary scale to protect outliers, the DS configuration (green line) effectively bypasses the clipping penalty, tracking the lower error bounds of the Wide Range (WR) configurations (INT9-WR, INT11-WR) and the unclipped INT8-Naive baseline.

However, the lower panel ($\alpha = 0.9995$) reveals a nuanced trade-off in the DS design. At this threshold, the clipping boundary is exceptionally broad. Because few outliers fall outside this range, the penalty for clipping is minimized, but the resolution available for the dense inlier region is stretched thin. Consequently, High Resolution (HR) formats—specifically INT11-HR (orange line)—provide the lowest accumulated error (approximately 0.002). In contrast, the DS scheme, alongside the INT8 baselines and WR formats, clusters at a higher error bound (approximately 0.010). This indicates a specific limitation: while the DS approach successfully preserves extreme

values, its foundational 8-bit mantissa may lack the fine-grained resolution necessary to suppress the slow accumulation of low-magnitude inlier noise over extended sequences when the main scale is parameterized too broadly.

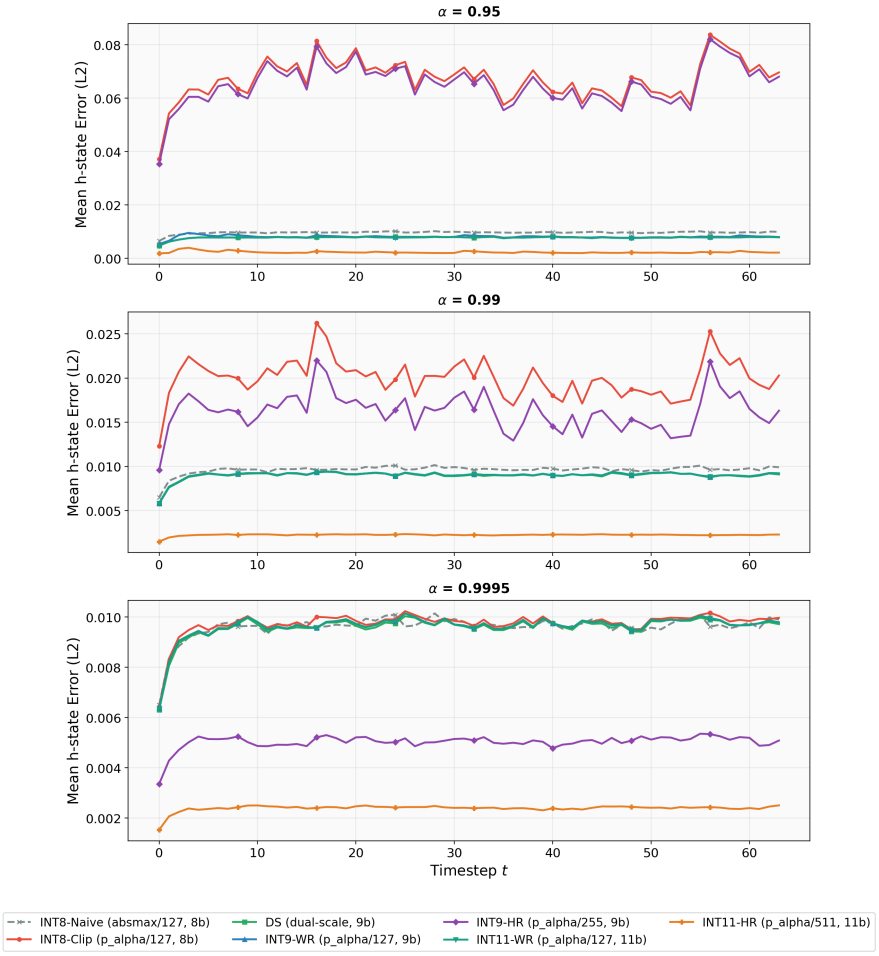

Figure 5.7: Mean L2 Error of the Hidden State h_t over 64 timesteps across varying alpha thresholds

Despite this limitation at extreme percentiles, the DS mechanism provides a robust and resilient middle ground for hardware deployment. It consistently avoids the catastrophic failure modes associated with uniform clipping at lower α values. By dynamically switching scales, it maintains acceptable hidden state fidelity without necessitating the expanded memory footprint and NoC bandwidth required by true 11-bit formats.

Figure 5.8 illustrates the mean absolute error of the final output projection y_t. The most immediate observation is that the overall magnitude of the

error is roughly an order of magnitude smaller than that of the hidden state. Furthermore, the plotted trajectories exhibit considerable step-to-step volatility, presenting a noisy and fluctuating pattern compared to the smooth accumulation curves seen in Figure 5.7.

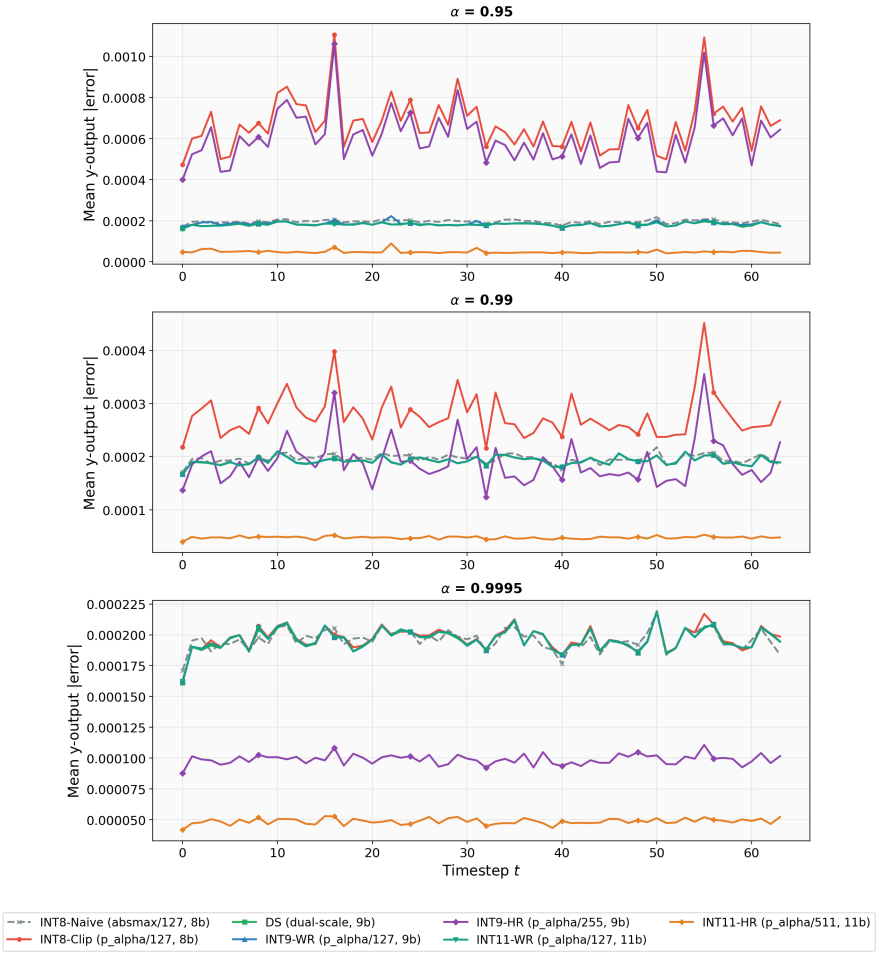

Figure 5.8: Mean Absolute Error of the Output Projection y_t over 64 timesteps across varying alpha thresholds

This volatility is a direct mathematical consequence of the output projection mechanism, defined as $y_t = Ch_t$. The projection matrix C acts as a linear transformation that maps the multi-dimensional hidden state back to a one-dimensional output. Because the elements within C are typically distributed around zero, the dot product with the accumulated error vector in h_t often results in a masking or cancellation effect. Positive and negative error components within the hidden state array can offset one another during the

summation, causing the observable error in y_t to fluctuate randomly. Due to this scrambling effect, the y_t metric serves primarily as a secondary reference rather than a strict diagnostic tool for continuous degradation.

Nevertheless, despite the inherent noise induced by the projection, the macro-level trends established during the hidden state analysis remain visible. For example, in the $\alpha = 0.95$ and $\alpha = 0.99$ panels, the INT8-Clip format generally sustains the highest error boundary, while the DS approach maintains a lower, more stable trajectory alongside the WR formats. Conversely, at $\alpha = 0.9995$, the HR formats again demonstrate a visible advantage over the clustered 8-bit and WR schemes.

The empirical evidence extracted from both the hidden state accumulation and the output projection tracking validates the physical implementation of the Dual-Scale hardware. The data confirms that managing outlier errors at the initial input boundary is critical for stabilizing the recurrent loop. By dynamically allocating numerical range via the 1-bit flag logic, the architecture effectively mitigates the most dangerous recurrent failure modes without inflating the system-level data payload.

5.4 Hyperparameter Robustness

In the deployment of PTQ for LLMs, selecting the optimal clipping threshold (α) is typically a fragile and resource-intensive process. The α parameter dictates the fundamental trade-off between clipping error and rounding error: setting α too low truncates structurally important outliers, while setting it too high compresses the dense inlier region into a limited number of quantization bins, eroding general resolution. Standard uniform quantization schemes are inherently brittle to this parameter, requiring exhaustive, layer-by-layer calibration searches to find an acceptable equilibrium. This section demonstrates that the proposed Dual-Scale (DS) hardware architecture decouples this dependency. By evaluating the quantization schemes across a broad sweep of percentiles ($\alpha \in \{0.90, 0.95, 0.99, 0.9995\}$), this section establishes that the DS approach maintains consistent fidelity regardless of the specific tuning parameter, thereby simplifying software deployment and reducing profiling overhead.

To confirm that these relative error trends translate into absolute numerical stability, the equivalent cross-comparison is performed using the Root Mean Square Error (RMSE), illustrated in Figure 5.9. The absolute error metrics mirror the trajectories established by the relative analysis. At $\alpha = 0.90$ and $\alpha = 0.95$, the standard INT8-Clip and INT9-HR schemes produce elevated RMSE

values, driven by the substantial magnitude of the truncated outliers. As α shifts toward 0.9995, the clipping threshold widens, alleviating the truncation penalty. However, at this broad setting, the INT8-Clip format begins to suffer from an inflated quantization step size (S), which elevates the ambient noise floor across the entire tensor, keeping its overall RMSE relatively high.

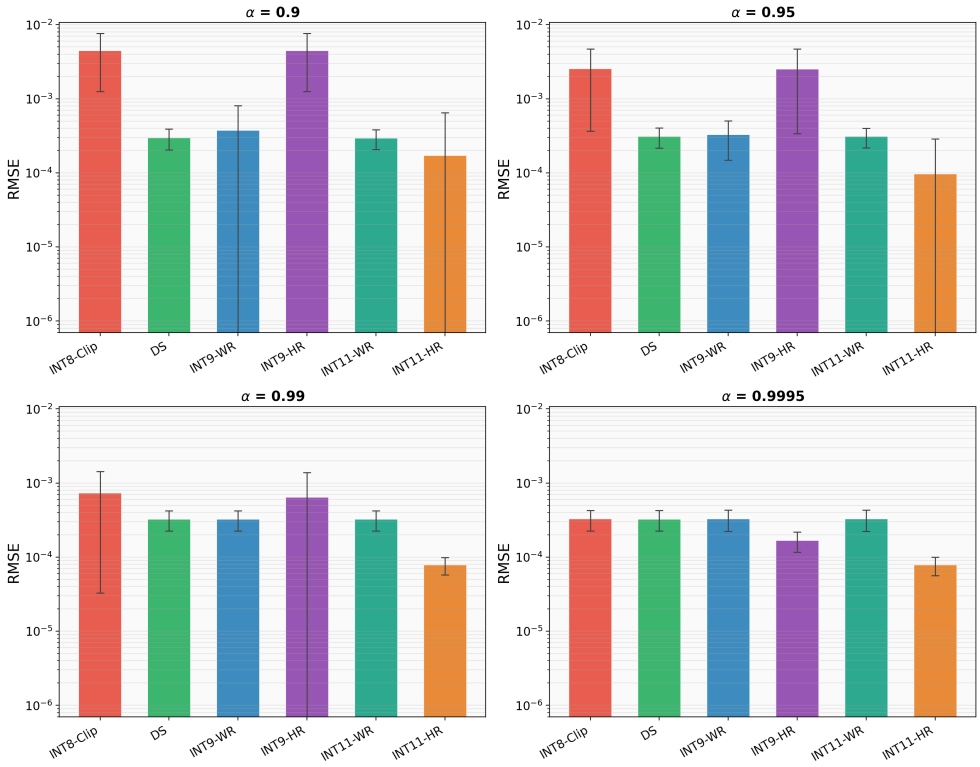

Figure 5.9: Box Plot Distribution of RMSE Variance across Different Calibration Percentiles

The DS scheme avoids both failure modes. By analyzing the RMSE data, it becomes clear that the DS format achieves an error baseline that closely mimics the behavior of the Wide Range (WR) configurations. The INT9-WR format (blue) performs steadily across the sweep because its inherently broad dynamic range absorbs the changing thresholds. However, comparing the DS scheme to INT9-WR highlights an important architectural distinction. Both formats occupy the same 9-bit network bandwidth payload, yet the DS scheme demonstrates a more consistent median and narrower variance in the box plots (Figure 5.9), particularly at intermediate thresholds like $\alpha = 0.95$. The WR format achieves its range by permanently sacrificing fine-grained resolution, whereas the DS multiplexer allocates wide range only to the specific elements

that require it, preserving higher resolution for the remaining data.

Furthermore, the data positions the 9-bit DS architecture favorably against the uncompressed 11-bit configurations. The error bounds maintained by the DS format remain competitive with the 11-bit WR format across multiple percentile choices. This observation carries weight for the overarching system design. It suggests that the proposed localized hardware multiplexing strategy is fundamentally more efficient than statically widening the data bus to absorb unknown variable distributions.

From a deployment perspective, this hyperparameter robustness translates directly into a reduction in operational friction. In a standard uniform quantization pipeline, machine learning engineers must meticulously profile the SSM over diverse datasets to pinpoint an α value that balances clipping and resolution for a specific workload. If the runtime data distribution shifts, the static threshold may cause the model to diverge. The Dual-Scale hardware architecture relaxes this constraint. Because the dynamic datapath is structurally immune to aggressive clipping errors and resistant to resolution dilution, the software stack can adopt a generalized, static percentile configuration without risking catastrophic failure during inference. This algorithmic leniency reduces the calibration overhead required for deployment associated with SSMs on resource-constrained edge devices, affirming the value of integrating adaptive quantization logic directly into the Processing Element.

5.5 Hardware Area and Timing Analysis

This section presents synthesis-level area and timing closure results for all evaluated configurations, confirming that the DS modification introduces negligible hardware overhead. The macroscopic hardware evaluation, illustrated in Figure 5.10, demonstrates that the DS architecture incurs a marginal total silicon area penalty compared to the baseline INT8 design. The DS configuration introduces an additional area of only $144\mu m^2$, elevating the total area footprint from $48671\mu m^2$ to $48815\mu m^2$. This corresponds to a relative area overhead of merely 0.30%, which is well within acceptable margins for physical chip layout and routing.

The uniform INT9 scheme occupies an identical 9-bit NoC payload to the DS scheme. However, the uniform INT9 format consumes 0.74% area overhead, which is more than double the area overhead of DS scheme. This indicates that adding a uniform bit of precision scales hardware complexity faster than adding a dedicated control flag.

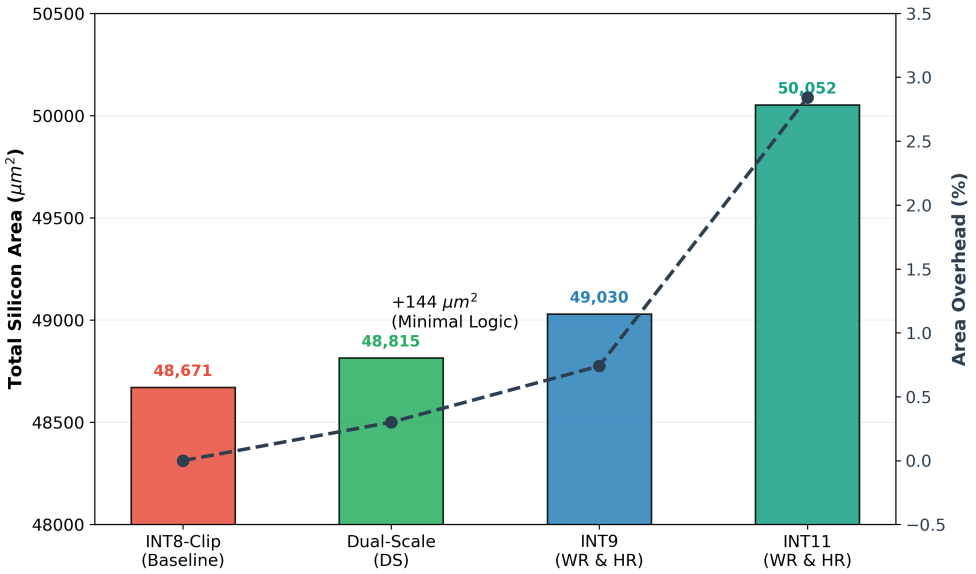

Figure 5.10: Total Area Comparison & Logic Overhead at 1GHz, 22nm technology node

For the uniform INT11 configuration, in order to approach the same level of numerical accuracy as the DS scheme, an area overhead of 2.84% (or $1381 \mu m^2$) is required. Compared to DS, this is nearly an order of magnitude larger than the physical cost of DS.

Figure 5.11 is a hierarchical breakdown of the silicon footprint. This figure contextualizes the area overheads, revealing that the shared FP32 execution backend and state registers dictate the fundamental area limits of the Processing Element (PE).

As shown by the truncated y-axis and the grey base layer, the shared FP32 components and memory arrays already consume over $48000 \mu m^2$. This base logic accounts for approximately 98% of the total PE area across all evaluated configurations. The dark blue layer represents the base area cost of baseline INT8 dequantizers, which is around $438 \mu m^2$. This cost is mandatory for bridging the compressed NoC traffic with the floating-point compute core.

Because the massive FP32 backend already consumes over 98% of the silicon budget, attempting to solve the outlier problem by uniformly expanding the frontend integer bit-width (e.g. to INT11) or transmitting uncompressed FP16 NoC payloads would be physically and spatially prohibitive. Therefore, intercepting the outliers via a lightweight, multiplexed Dual-Scale frontend is proven to be the viable, low-cost optimization path.

Analyzing the topmost layers of the area breakdown exposes the physical

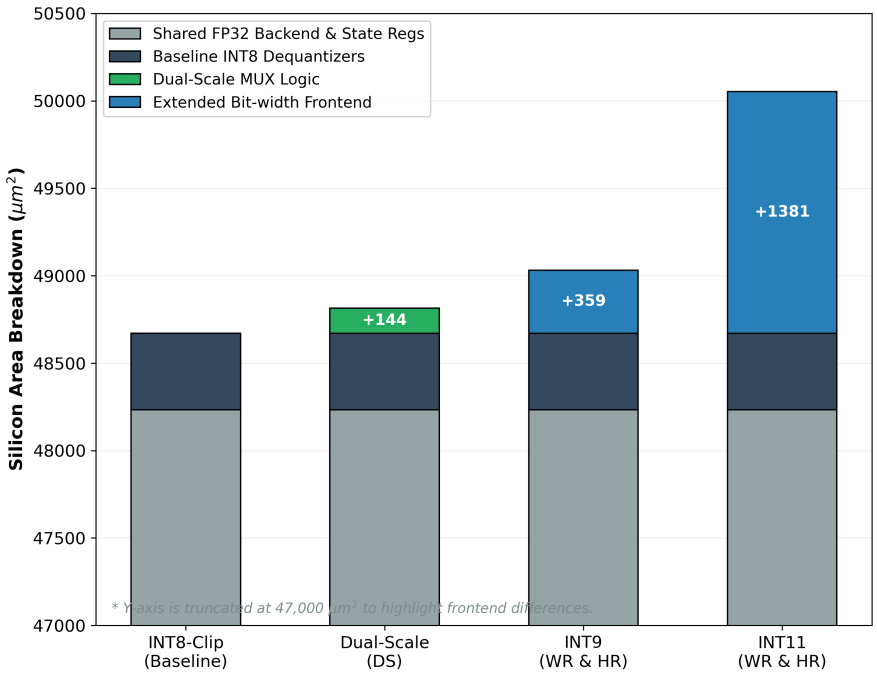

Figure 5.11: Hierarchical Area Breakdown across Evaluated Quantization Schemes

inefficiency of expanding uniform bit-widths compared to the targeted, localized multiplexing logic of the DS design. The Extended Bit-width Frontend layers in blue show that increasing the precision to INT9 and INT11 forces the synthesis tool to instantiate wider data buses and more complex combinational logic within the INTx_to_FP32 modules. INT9 and INT11 schemes add $359\mu m^2$ and $1381\mu m^2$ respectively.

In contrast, the DS logic, as the green block shows, introduces only $144\mu m^2$ of extra logic. Since it reuses the standard 8-bit combinational converter, the added area is strictly limited to the 1-bit flag registers and the lightweight 2-to-1 FP32 scale multiplexers. This discrepancy highlights the value of algorithm-hardware co-design. Since algorithmic profiling established that the long-tail outlier phenomenon is isolated to tensor u, paying the global hardware tax to expand the dequantizers for all tensors (A, B, C, Δ, z) – as uniform INT11 does – is architecturally wasteful.

Table 5.2 confirms that all four synthesized configurations meet the 1GHz timing constraint, with worst-case slack values of 0.00-0.01 ps. The critical path in both the baseline and DS designs originates within the `div_silu` FP32 division chain used for SiLU gating, confirming that the

Table 5.2: Timing closure results across evaluated configurations.

Design	WNS (ps)	Critical Delay (ps)	Critical Module
INT8-FSM (Baseline)	0.00	926.19	`div_silu`
DS	0.00	923.76	`div_silu`
INT9-WR/HR	0.00	928.37	`scale_mult_z`
INT11-WR/HR	0.01	926.44	`mult_Du`

22nm, 0.80 V, 25 °C, target clock period = 1000 ps, clock uncertainty = 50 ps.

Scale MUX inserted by the DS extension does not displace the timing bottleneck. DS achieves a critical path arrival of 923.76 ps, shorter than the baseline's 926.19 ps possibly due to post-insertion logic restructuring by Design Compiler. For the INT9 and INT11 variants, the critical path migrates to the wider input scale multiplier stages (`scale_mult_z` for example), reflecting the additional gate depth introduced by sign-extension in the `INTx_to_FP32` conversion. Yet both INT9 and INT11 variants remain within timing budget, validating the parameterized frontend approach of Section 4.4.

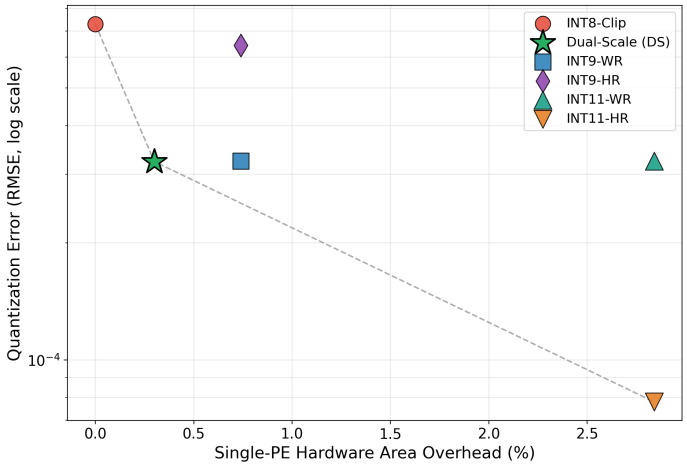

Figure 5.12: Pareto Frontier of Single-PE Area Overhead versus Quantization Error (RMSE)

In Figure 5.12, the multi objective optimization landscape is visualized through a comprehensive Pareto frontier. This graph directly contrasts the localized processing element area overhead against the resulting RMSE. The baseline INT8 configuration anchors the top left boundary of the evaluated

optimization space. This baseline scheme presents zero physical area overhead, but exhibits an elevated root mean square error. Specifically, the numerical error for this baseline configuration reaches approximately 7.29×10^{-4} under standard validation workloads. On the other hand, the uniform higher precision formats anchor the opposite extreme at the bottom right boundary. Among the uniform formats, only INT11-HR achieves a low error, but at the cost of significant area expansion. Ultimately, the area penalty for the INT11 configurations approaches approximately 2.84% at the processing element level.

Moreover, the uniform 9-bit formats illustrate an exceptionally suboptimal deployment of localized hardware resources. The INT9-HR configuration registers a relatively high mean root mean square error of 6.44×10^{-4} in this simulation; while it demands an area overhead of 0.74% within the processing element frontend. Similarly, the INT9-WR variant lowers the RMSE to approximately 3.24×10^{-4} in validation tests, but it still has the identical 0.74% area overhead to INT9-HR. Both uniform 9-bit designs incur more than double the area overhead of the proposed DS architecture: INT9-HR for a worse RMSE than DS, and INT9-WR for an RMSE essentially equal to DS. Neither improves on DS in accuracy, yet both cost over twice its area.

In comparison, the proposed Dual-Scale architecture dictates the optimal bend along the illustrated Pareto frontier curve. The DS configuration achieves a low RMSE of 3.23×10^{-4}, matching the numerical precision of the wider INT9-WR and INT11-WR variants (3.24×10^{-4}). Crucially, the DS scheme achieves this accuracy while restricting area overhead to merely 0.30%. DS therefore attains the same RMSE as INT9-WR at less than half the area overhead, and matches INT11-WR accuracy at roughly one-tenth of its area cost. Thus, the localized adaptive multiplexer logic successfully uncouples numerical fidelity from physical layout size within the hardware. This structural dominance clearly proves the superiority of asymmetric quantization when applied to small-fraction large-magnitude outlier distributions.

Ultimately, the area, timing, and Pareto results establish that the Dual-Scale design exploits algorithmic asymmetry to escape the conventional precision-area trade-off. The DS architecture successfully uncouples numerical fidelity from physical layout size. It attains the MaxAE suppression capabilities of an 11-bit system while maintaining an area profile that is leaner than a 9-bit system. By confining the hardware intervention to a localized frontend MUX rather than expanding the foundational bit-width, the PE preserves its tight area constraints. This directly enables the overall NoC to maintain dense 9-bit packet routing, ensuring that high spatial computation

density does not compromise network bandwidth.

5.6 System-Level Throughput and NoC Bandwidth Analysis

Evaluating the Processing Element in isolation captures only its local cost and accuracy. The system-level efficacy of each quantization scheme must also be assessed under the bandwidth constraints of the shared interconnect. Large-scale SSM inference is often assumed to be communication-bound, since continuous parameter streaming stresses the interconnect bandwidth. Whether this holds for a single PE depends on how much data actually crosses the network per token, which is quantified in the following analysis.

To situate the per-PE overhead within the complete NoC fabric, the 4×4 Constellation mesh wrapper was synthesized in Design Compiler with all 16 routers and 16 PEs instantiated, the PE locked as pre-compiled hierarchy so that the synthesis effort is directed at the routers and interconnect glue. The synthesized per-router area separates cleanly into three groups by mesh position and port count: the four corner routers (with two ports) at approximately $21277\mu m^2$, the eight edge routers (with three ports) at approximately $30933\mu m^2$, and the four interior routers (with four ports) at approximately $40760\mu m^2$. The total synthesized NoC area is $495612\mu m^2$, and the intra-group spread is below 0.05%. This monotonic growth with port count, together with the negligible within-group variance, confirms that the reported figures are genuine synthesis results rather than analytical estimates, and provides a single physically-grounded anchor for the otherwise analytical system-level model.

Two consequences follow for the Dual-Scale design. First, a router and a PE are of the same order of magnitude: the average router ($\approx 30976\mu m^2$) is roughly 64% of a single INT8 PE ($48671\mu m^2$), so a complete compute node of one PE plus one router measures$\approx 79647\mu m^2$. Second and consequently, the Dual-Scale modification is negligible at every level of this hierarchy. The $144\mu m^2$ that DS adds to the PE is 0.30% of the PE in isolation, and only 0.18% of a full PE-plus-router node. Because the DS extension is confined to the PE dequantization frontend and leaves the flit format carried by the routers unchanged, the router area is unaffected by construction. The system-level cost of Dual-Scale is therefore strictly bounded by its already-small PE-level overhead, and is diluted further once the substantial router area is accounted for.

Having established that the per-node area is dominated by the shared FP32 backend and the router, with the Dual-Scale logic contributing a negligible fraction of either, the remainder of this section turns to the throughput implications of each quantization scheme under the bandwidth constraints of the shared interconnect.

Figure 5.13 presents the roofline of a single SSM-PE. The horizontal axis is the operational intensity, measured as FP32 operations per byte transferred over the NoC. The vertical axis is the attained throughput in FP32 operations per second. Three horizontal compute roofs correspond to state dimensions $N_{state} = 8$, 16, and 32, at 1.29, 1.89, and 2.59 GOPS respectively. The $N_{state} = 16$ roof is anchored by the RTL-simulated 121-cycle per-token interval, and the other two follow the linear cycle model of Section 4.5. The per-token interval is 121 cycles because this includes the handshake of signals in the RTL simulation. The sloped lines are the communication roofs for the narrowest and widest flit widths. Each marker is one quantization scheme placed at its own operational intensity.

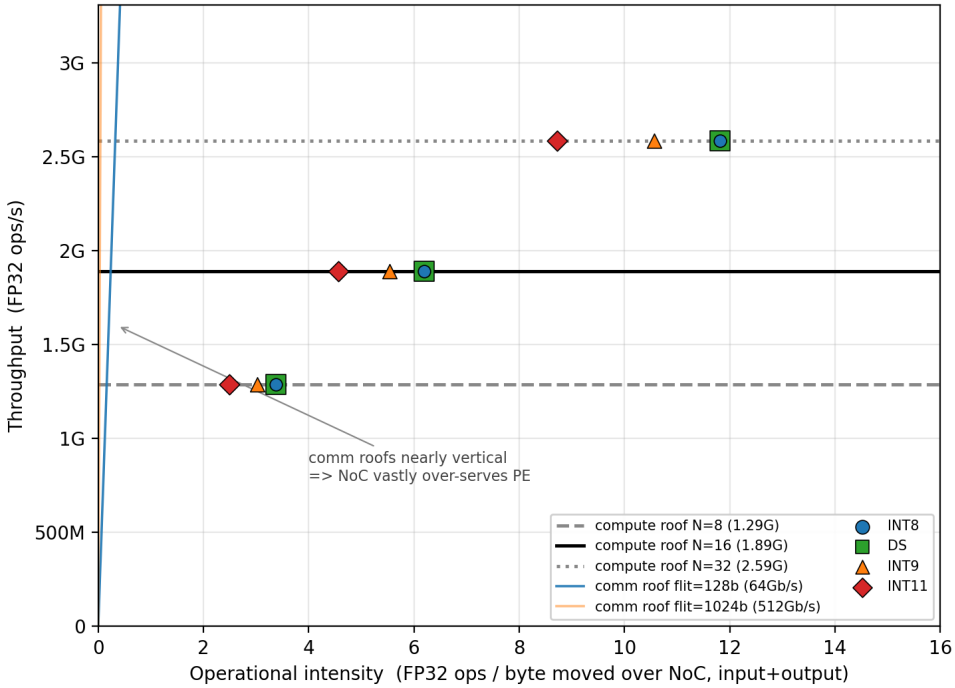

Figure 5.13: Roofline of a single SSM-PE across quantization schemes and state dimensions with $F_{PE} = 1$ GHz

All operating points lie on the compute roofs, far to the right of the

communication roofs. Even the narrowest 128-bit flit delivers 64 Gb/s, whereas the PE consumes only about 280-385 bits per token, so the communication roof is nearly vertical and all operating points rest on the compute roof. The quantization scheme shifts only the horizontal position of a point, since a wider payload lowers the operational intensity, but does not change the attained throughput. This is the opposite of the memory-bound regime typically expected of SSM inference: by keeping the recurrent state local to each PE, the design moves the per-PE operating point firmly into the compute-bound region.

Figure 5.14 reports the achieved throughput as the NoC per-port bandwidth is swept across flit widths from 8 to 1024 bits. The horizontal axis is the per-port bandwidth on a logarithmic scale, computed as flit width × routing frequency. The vertical axis is the achieved throughput. Each curve corresponds to one quantization scheme, and the dashed line marks the 1.89 GOPS compute ceiling at $N_{state} = 16$. The shaded band on the left indicates the communication-bound region.

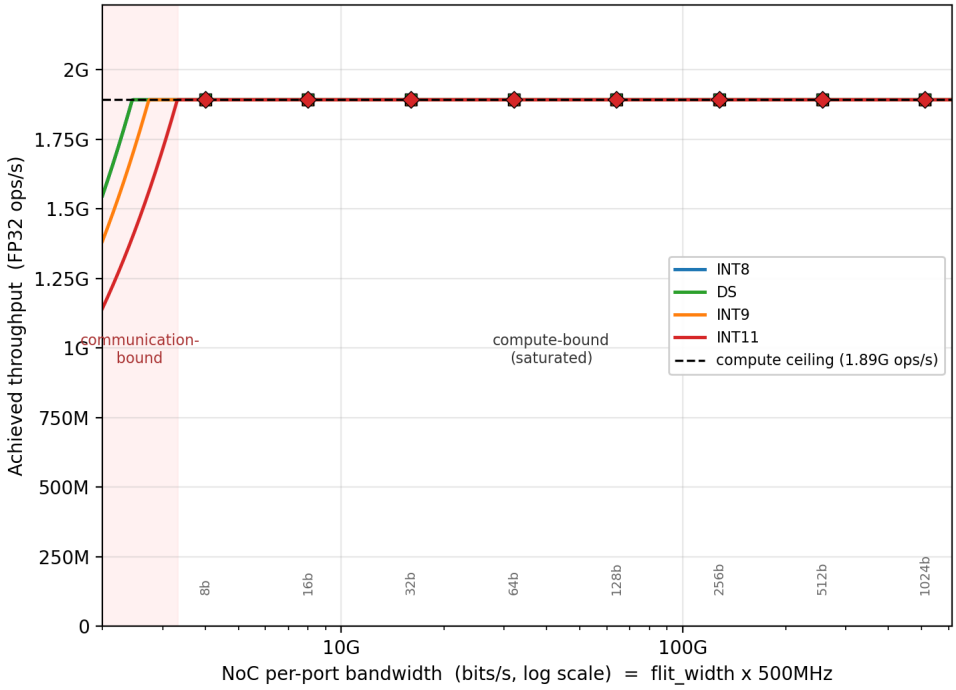

Figure 5.14: Achieved Throughput against NoC per-port Bandwidth across Flit Widths

This figure characterizes where the system transitions between the two

regimes. The achieved throughput rises with bandwidth only within a narrow band at the smallest flit widths, and reaches the compute ceiling for every realistic configuration with the knee lying below an 8-bit flit. The four schemes differ only inside the communication-bound band where the wider INT11 payload reaches the ceiling last, and they coincide for all large flit widths. No practical NoC configuration therefore starves the PE. The fabric over-serves the compute unit by more than an order of magnitude, which confirms that the bottleneck of a single PE is computation rather than communication.

5.7 Discussion

This section interprets the experimental findings in the context of the research question, explains the mechanisms underlying the key results, and situates the contributions of this thesis relative to existing work in the field.

The MaxAE advantage of DS is over the clipping-based schemes, not the wide-range schemes, and it is a direct structural consequence of converting clipping error into bounded rounding error. INT8-Clip and INT9-HR retain a representable range of $[-X_\alpha, X_\alpha]$ – INT9-HR spends its extra bit on a finer step ($X_\alpha/255$) rather than on wider range – so any element exceeding X_α is saturated, producing their large MaxAE values of 4.75×10^{-3} and 4.53×10^{-3}. DS instead assigns the outlier elements $S_{\text{outlier}} = X_{\text{max}}/127$, extending its range to $[-X_{max}, X_{max}$: no element is clipped, and the rounding error is bounded by $S_{outlier}/2 = X_{max}/254$. This is the mechanistic explanation for the 69% MaxAE reduction relative to INT8-Clip reported in Section 5.2. The wide-range INT9-WR and INT11-WR formats reach the same unclipped regime by widening their range to $\pm 2X_\alpha$ and $\pm 8X_\alpha$ respectively, and accordingly DS shows no MaxAE advantage over them (1.46×10^{-3} vs 1.47×10^{-3}); the near-equality in RMSE (3.23×10^{-4} vs 3.24×10^{-4}) further confirms that DS and the wide-range formats handle both the inlier majority and the outlier tail with comparable resolution. The distinguishing factor is therefore payload width alone: DS attains this shared accuracy floor at a 9-bit payload, whereas INT11-WR requires 11 bits.

The empirical collapse of INT9-HR observed in Section 5.4 has a precise structural explanation. INT9-HR sets $s_{\text{HR}} = X_\alpha/255$, achieving a finer inlier step size than INT9-WR, but the clipping threshold remains at X_α. At $\alpha = 0.99$, this boundary is tighter than at $\alpha = 0.9995$, meaning a larger fraction of the activation distribution exceeds it; these elements are saturated and generate large clipping errors that dominate the RMSE. DS is structurally immune to this trade-off because the flag mechanism decouples dynamic range

from resolution entirely: inliers always receive the fine s_{main} step, and outliers always receive the wide s_{outlier} range, regardless of how α is set.

The 0.30% area overhead of the DS modification is not merely a favorable measurement; it is a structural consequence of a general principle. When an algorithmic asymmetry exists in the input distribution (outliers concentrated in u, not in A, B, C), a hardware design that exploits this asymmetry will outperform one that treats all channels uniformly. Uniform bit-width expansion cannot exploit input asymmetry by construction: widening `INPUT_WIDTH` from 8 to 11 applies the extra gate count to every tensor channel in every clock cycle, regardless of whether that channel ever contains an outlier. The DS multiplexer, by contrast, adds logic only at the point where the asymmetry exists, and activates the wide-range path only when the flag signals an outlier.

The roofline analysis in Figure 5.13 places every quantization scheme on the compute roof rather than the communication roof, establishing that a single SSM-PE is compute-bound once its recurrent state is kept local. Within this regime, the coincidence of the DS and INT9 throughput curves in Figure 5.14 is not a limitation of the DS design but a confirmation of its efficiency. Both schemes transmit a 9-bit payload per element, making them informationally equivalent at the NoC router level; the DS accuracy advantage exists entirely within the dequantization logic inside the PE, invisible to the network fabric. As shown in Figure 5.14, the 9-bit payload reaches the compute ceiling at a smaller NoC bandwidth than the 11-bit INT11 format. DS thus attains INT11-WR accuracy at a 9-bit payload, a combination that uniform quantization schemes do not achieve at equal payload width.

This thesis builds on Quamba's central empirical finding that outlier activations in selective SSMs are concentrated in tensor u [6], but departs from its outlier-handling strategy. Quamba resolves the inlier/outlier tension by clipping the input outliers to a percentile threshold, accepting the resulting truncation error in the tail as the price of finer inlier resolution. This thesis takes a different position: rather than discarding the outliers, it introduces an explicit second scale for the outlier region that preserves them, and resolves the resulting per-element scale selection entirely in combinational ASIC hardware via a 1-bit flag and a 2-to-1 MUX. The novelty is therefore twofold. First, at the algorithmic level, dual-scale preserves the structurally critical outliers that Quamba's percentile clipping truncates, converting unbounded clipping error into bounded rounding error (Section 3.1.3). Second, at the hardware level, this thesis quantifies the silicon cost of per-element flag-driven scale selection synthesized at a 22nm node, showing that it adds only 0.30% area and zero

pipeline stages. The transition from a lossy algorithmic clipping recipe to an outlier-preserving, synthesizable per-element hardware mechanism is the primary contribution of this thesis.

As established in Section 2.4, existing SSM hardware accelerators either optimize the compute datapath without addressing quantization [13, 14], or address quantization on FPGA platforms where timing constraints differ fundamentally from a 22nm ASIC target [7, 39]. This thesis occupies a distinct position: it targets the input dequantization frontend rather than the compute datapath, addressing the NoC-to-PE interface that other works assume away.

All area and timing results are synthesis-level estimates from Design Compiler and do not account for post-layout parasitics, routing congestion, or IR drop. These limitations do not invalidate the relative comparisons between quantization schemes, since all variants are synthesized under identical constraints with a structurally frozen FP32 backend; area deltas are therefore attributable exclusively to the input representation width, regardless of absolute post-layout shifts. The near-zero worst-case slack reflects the convergence behaviour of `compile_ultra -retime` rather than a marginal timing result.

The throughput analysis is based on an analytical GALS model rather than cycle-accurate multi-node RTL simulation. The primary claim, that the 9-bit payloads reach the compute ceiling at a lower NoC bandwidth than the 11-bit payloads, followed structurally from the payload width ratio and is expected to hold under the analytical model's assumptions; quantifying its robustness to router contention and flow-control effects requires the cycle-accurate simulation identified below. Absolute throughput numbers, however, should be treated as upper bounds. Cycle-accurate multi-node simulation using tools such as BookSim2 or a gem5-based NoC model is identified as the most important missing validation step for future work.

5.8 Summary

This chapter presented a comprehensive evaluation of the proposed Dual-Scale architecture across four dimensions. At the element level ($\alpha = 0.99$, 200 seeds), DS achieves RMSE $= 3.23 \times 10^{-4}$, matching INT9-WR and INT11-WR (3.24×10^{-4}), while suppressing MaxAE to 1.46×10^{-3} which is a 69% reduction relative to INT8-Clip. Critically, INT9-HR degrades to INT8-level RMSE (6.44×10^{-4}) at $\alpha = 0.99$, confirming that HR formats are structurally brittle at deployment-realistic clipping thresholds. The hyperparameter robustness analysis confirmed that DS maintains consistent

per-seed RMSE distributions across $\alpha \in 0.90, 0.95, 0.99, 0.9995$, while INT9-HR exhibits high-variance behavior at tighter α values. Timing closure was met across all four synthesized configurations with worst-case slack of 0.00-0.01 ps at 1 GHz. The DS modification contributes $144\mu m^2$ or 0.30% overhead, compared to 0.74% for INT9 and 2.84% for INT11, delivering INT11-WR accuracy at less than one-tenth the area cost. At the system level, the single PE is compute-bound, so DS and INT9 attain identical throughput while DS is better than INT9 in Pareto on accuracy-area. These results confirm that ASIC combinational logic enables per-element scale selection at negligible cost, achieving an accuracy-bandwidth trade-off that uniform quantization schemes cannot match at equal payload width.

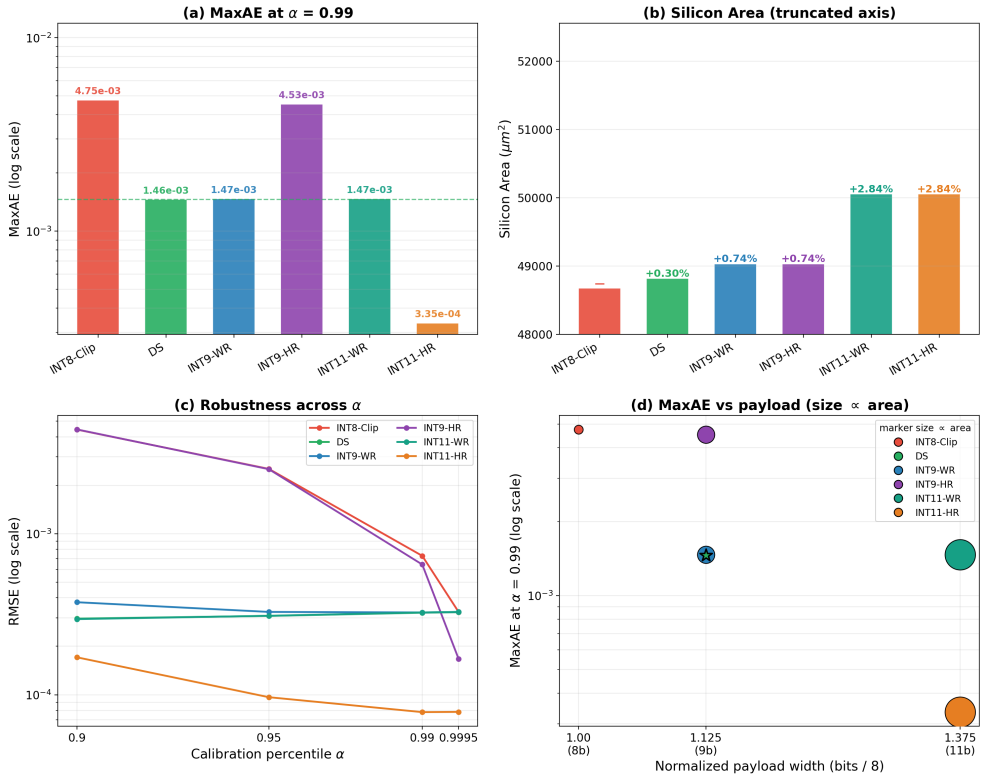

Figure 5.15: Summary of accuracy, area, robustness, and bandwidth efficiency across the evaluated schemes

Figure 5.15 consolidates the four evaluation dimensions into a single view. Subfigure (a) reports MaxAE at $\alpha = 0.99$; subfigure (b) reports the area, where the WR and HR variants of each bit-width share an area value because area depends only on the integer width of the dequantization frontend; subfigure (c) reports the RMSE trend across the calibration-percentile sweep;

and subfigure (d) reports MaxAE against normalized payload width, with marker size proportional to area overhead. Read together, the panels show a single consistent results: DS matches the wide-range INT9-WR and INT11-WR formats in accuracy (a) at only 0.30% area overhead (b), remains stable across α sweep where the clipping-based INT8-Clip and INT9-HR degrade (c), and reaches this accuracy floor at the minimum 9-bit payload (d). In subfigure (d), DS and INT9-WR coincide in both accuracy and payload, but DS's lower area cost distincts from INT9-WR.

Chapter 6

Conclusions and Future work

6.1 Conclusions

This thesis set out to determine whether large-magnitude activation outliers in selective State Space Models can be preserved directly in hardware, without the area and bandwidth penalties of wider uniform formats. To answer this, a flag-bit-driven Dual-Scale (DS) dequantization frontend was designed for an SSM processing element, implemented in synthesizable SystemVerilog RTL, verified against a bit-accurate Python golden model across 1000 random seeds, and synthesized at a 22nm technology node using Synopsys Design Compiler. The DS scheme assigns each element of the input tensor u a 1-bit flag during offline calibration, which at inference drives a combinational multiplexer selecting between a fine-grained inlier scale and a wide-range outlier scale.

The research question is answered affirmatively across all three of its dimensions. On accuracy, DS matches the wider INT9-WR and INT11-WR formats while operating at a lower payload width: at $\alpha = 0.99$ over 200 random seeds, it attains an RMSE of 3.23×10^{-4} compared to 3.24×10^{-4} of INT9-WR and INT11-WR and suppresses the maximum absolute error to 1.46×10^{-3}, a 69% reduction relative to INT8-Clip baseline. On area, DS reaches INT11-WR-level accuracy at less than one-tenth of its area cost: the modification adds only $144\mu m^2$, a 0.30% overhead relative to the complete PE, compared to 0.74% for uniform INT9 and 2.84% for uniform INT11. On latency, timing closure is met across all variants at 1 GHz with no additional pipeline stage, as the multiplexer delay is absorbed within the existing INT8-to-FP32 conversion. In combination, DS delivers the accuracy of an 11-bit uniform format at a 9-bit payload width and negligible silicon cost.

The broader significance of these results lies in what they confirm about the ASIC implementation paradigm. GPU-based quantization is constrained to uniform dequantization kernels because CUDA thread-block execution cannot perform per-element scale selection without warp divergence. An ASIC combinational multiplexer achieves the same per-element selectivity with no cycle cost, absorbing the selection latency within an existing converter delay that the compiler would otherwise leave idle. This architectural freedom, unavailable on GPU but straightforward on custom silicon, is a defining advantage of the ASIC realization.

6.2 Future work

The work in this thesis can be extended along three broad directions: hardening the physical implementation, broadening the evaluation toward end-to-end deployment, and generalizing the dual-scale mechanism beyond its current single-channel, single-accelerator scope. Because the DS frontend is modular and confined to the dequantization datapath, this last direction includes integrating it into existing SSM accelerator datapaths, pointing towards a direction for combined datapath-and-dequantization optimization. The remainder of this section develops these directions from the most immediate to the longer-term.

The most immediate next step is place-and-route in 22nm technology node, enabling parasitic-extracted timing verification and SAIF-annotated power analysis. Post-layout results would either confirm the synthesis-level area deltas or identify routing-driven discrepancies that motivate architectural adjustment. Gate-level power simulation with a realistic switching activity file would also resolve the current omission of power numbers from the area and timing evaluation. Cycle-accurate multi-node simulation of the 16-PE Constellation fabric using simulation tools would validate the analytical GALS throughput model under realistic router contention and credit-based flow control, converting the current upper-bound estimates into empirically grounded throughput figures. A third near-term direction is multi-tensor DS profiling: characterizing the activation distributions of A, B, C, and Δ in Mamba-2 and larger model variants would determine whether the outlier-in-u assumption generalizes. If not, it would motivate a multi-channel DS extension whose additional area cost could be quantified through the existing parameterized synthesis methodology.

At a longer horizon, end-to-end task accuracy evaluation represents the most impactful validation gap. The RMSE and MaxAE results of Chapter 5

establish element-level and sequence-level numerical fidelity, but do not directly measure the effect on downstream language modeling performance. Connecting the RTL PE output to a Quamba-style inference pipeline and evaluating perplexity on benchmarks such as LAMBADA [51] or WikiText-103 would bridge this gap and provide the application-level evidence needed for a complete accuracy claim. A second longer-term direction is non-uniform quantization: logarithmic or power-of-two quantization grids could potentially reduce S_{outlier} while maintaining comparable MaxAE, which would further compress the outlier payload and improve NoC efficiency beyond the current 9-bit floor. Finally, the FP32 MAC datapath identified in the timing analysis as the critical module is also the highest fault-sensitivity point in the recurrent computation. Integrating concurrent error detection using residue arithmetic into this path would be a natural orthogonal enhancement that adds resilience without perturbing the quantization frontend.

6.3 Reflections

This section reflects on the methodological choices made during the project and their wider implications.

A central methodological decision was to implement and functionally verify the RTL before performing logic synthesis, rather than estimating cost from high-level models. The 1000-seed bit-accurate comparison between the Python golden model and the SystemVerilog testbench ensured that the reported area and timing numbers reflect a design that faithfully implements the specified datapath rather than an idealized abstraction. In retrospect, two methodological improvements would strengthen a repeat of this work. First, adopting a formal UVM-based verification environment from the outset, rather than a directed SystemVerilog testbench, would enable systematic functional-coverage analysis and reduce the risk of unexercised corner cases. Second, the system-level interconnect was characterized with an analytical bandwidth model owing to time constraints; a cycle-accurate NoC simulation would provide a more faithful account of contention behaviour, particularly once multiple processing elements share the fabric.

The interdisciplinary scope of the project, spanning quantization algorithms, RTL microarchitecture, and physical synthesis, required substantial context-switching between domains and toolchains. Parameterizing the synthesis scripts by `INPUT_WIDTH` and state dimension proved valuable for producing a fair, like-for-like comparison across quantization schemes, and a more systematic build automation framework from the beginning would have

reduced the re-run overhead incurred whenever a design variant was added.

Beyond the immediate technical contribution, the work carries broader relevance. By reducing the precision required to transmit and store SSM activations without sacrificing accuracy, dual-scale quantization lowers the memory and the bandwidth demands of inference, which in turn reduces the energy consumed per token on custom hardware. Energy-efficient inference accelerators contribute to the sustainability of deploying large sequence models, and lowering the hardware cost of accurate low-bit inference may help make such models viable on resource-constrained edge devices rather than only in data centers. At the same time, more efficient language-model inference lowers the barrier to large-scale deployment, which reinforces the existing need for responsible use of generative models. This consideration lies outside the technical scope of the thesis but is acknowledged as part of its wider context.

References

[1] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the 44th annual international symposium on computer architecture*, 2017, pp. 1–12.

[2] W. A. Wulf and S. A. McKee, "Hitting the memory wall: Implications of the obvious," *ACM SIGARCH computer architecture news*, vol. 23, no. 1, pp. 20–24, 1995.

[3] B. N. Patro and V. S. Agneeswaran, "Mamba-360: Survey of state space models as transformer alternative for long sequence modelling: Methods, applications, and challenges," *Engineering Applications of Artificial Intelligence*, vol. 159, p. 111279, 2025.

[4] A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," *arXiv preprint arXiv:2312.00752*, 2023.

[5] T. Dao and A. Gu, "Transformers are ssms: Generalized models and efficient algorithms through structured state space duality," 2024. [Online]. Available: https://arxiv.org/abs/2405.21060

[6] H.-Y. Chiang, C.-C. Chang, N. Frumkin, K.-C. Wu, and D. Marculescu, "Quamba: A post-training quantization recipe for selective state space models," in *International Conference on Learning Representations*, vol. 2025, 2025, pp. 101 328–101 354.

[7] R. Wei, S. Xu, L. Zhong, Z. Yang, Q. Guo, Y. Wang, R. Wang, and M. Li, "Lightmamba: Efficient mamba acceleration on fpga with quantization and hardware co-design," in *2025 Design, Automation & Test in Europe Conference (DATE)*. IEEE, 2025, pp. 1–7.

[8] J. Kim, J. Lee, J. Lin, A. Kanani, S. Miao, U. Ogras, and J. Park, "emamba: efficient acceleration framework for mamba models in edge computing," *ACM Transactions on Embedded Computing Systems*, vol. 24, no. 5s, pp. 1–22, 2025.

[9] H. Kim, B. Ko, M. Kang, M. Kim, D. Lee, J. Lee, S. Yoon, and D. Jung, "SSDi8: Accurate and efficient 8-bit quantization for state space duality," in *The Fourteenth International Conference on Learning Representations*, 2026. [Online]. Available: https://openreview.net/forum?id=pjMDZJd4rT

[10] Z. Xu, Y. Yue, X. Hu, Z. Yuan, Z. Jiang, Z. Chen, J. Yu, C. Xu, S. Zhou, and D. Yang, "Mambaquant: Quantizing the mamba family with variance aligned rotation methods," *arXiv preprint arXiv:2501.13484*, 2025.

[11] H.-Y. Chiang, C.-C. Chang, N. Frumkin, K.-C. Wu, M. S. Abdelfattah, and D. Marculescu, "Quamba2: A robust and scalable post-training quantization framework for selective state space models," 2025. [Online]. Available: https://arxiv.org/abs/2503.22879

[12] W. J. Dally and B. P. Towles, *Principles and practices of interconnection networks*. Elsevier, 2004.

[13] J. Li, S. Huang, J. Xu, J. Liu, L. Ding, N. Xu, and G. Dai, "Marca: Mamba accelerator with reconfigurable architecture," in *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*, ser. ICCAD '24. New York, NY, USA: Association for Computing Machinery, 2025. doi: 10.1145/3676536.3676798. ISBN 9798400710773. [Online]. Available: https://doi.org/10.1145/3676536.3676798

[14] S. Raja, C. Demirkiran, A. Sarkar, M. Popovic, and A. Joshi, "Systolic array-based accelerator for structured state-space models," 2025. [Online]. Available: https://arxiv.org/abs/2507.21394

[15] J. Li, S. Huang, J. Xu, J. Liu, N. Xu, and G. Dai, "Marca-v2: Mamba accelerator with complementary state space model sparsity and reconfigurable architecture," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2025.

[16] S. Williams, A. Waterman, and D. Patterson, "Roofline: an insightful visual performance model for multicore architectures," *Communications of the ACM*, vol. 52, no. 4, pp. 65–76, 2009.

[17] S. Lu, X. Yu, T. Zhao, S. Miao, X. Sheng, C. Wu, L. Zhao, T.-J. Lin, and L. He, "Mambaopu: An fpga overlay processor for state-space-duality-based mamba models," in *2025 62nd ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2025, pp. 1–7.

[18] E. Lindholm, J. Nickolls, S. Oberman, and J. Montrym, "Nvidia tesla: A unified graphics and computing architecture," *IEEE micro*, vol. 28, no. 2, pp. 39–55, 2008.

[19] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE journal of solid-state circuits*, vol. 52, no. 1, pp. 127–138, 2016.

[20] L. Zhong, S. Xu, H. Wen, T. Xie, Q. Guo, Y. Wang, and M. Li, "Specmamba: Accelerating mamba inference on fpga with speculative decoding," in *2025 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 2025, pp. 1–9.

[21] A. Gu, K. Goel, and C. Ré, "Efficiently modeling long sequences with structured state spaces," *arXiv preprint arXiv:2111.00396*, 2021.

[22] L. Zhu, B. Liao, Q. Zhang, X. Wang, W. Liu, and X. Wang, "Vision mamba: Efficient visual representation learning with bidirectional state space model," *arXiv preprint arXiv:2401.09417*, 2024.

[23] O. Lieber, B. Lenz, H. Bata, G. Cohen, J. Osin, I. Dalmedigos, E. Safahi, S. Meirom, Y. Belinkov, S. Shalev-Shwartz *et al.*, "Jamba: A hybrid transformer-mamba language model," *arXiv preprint arXiv:2403.19887*, 2024.

[24] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," in *Low-power computer vision*. Chapman and Hall/CRC, 2022, pp. 291–326.

[25] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for

efficient integer-arithmetic-only inference," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2704–2713.

[26] S. Siegel, M.-J. Yang, Y. Bouhadjar, M. Fabre, E. Neftci, and J. P. Strachan, "Qs4d: Quantization-aware training for efficient hardware deployment of structured state-space sequential models," *Advanced Intelligent Systems*, vol. 8, no. 4, p. e202501019, 2026.

[27] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer, "Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale," *Advances in neural information processing systems*, vol. 35, pp. 30 318–30 332, 2022.

[28] G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth, and S. Han, "Smoothquant: Accurate and efficient post-training quantization for large language models," in *International conference on machine learning*. PMLR, 2023, pp. 38 087–38 099.

[29] J. Lin, J. Tang, H. Tang, S. Yang, W.-M. Chen, W.-C. Wang, G. Xiao, X. Dang, C. Gan, and S. Han, "Awq: Activation-aware weight quantization for on-device llm compression and acceleration," *Proceedings of machine learning and systems*, vol. 6, pp. 87–100, 2024.

[30] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh, "Gptq: Accurate post-training quantization for generative pre-trained transformers," *arXiv preprint arXiv:2210.17323*, 2022.

[31] C. Guo, J. Tang, W. Hu, J. Leng, C. Zhang, F. Yang, Y. Liu, M. Guo, and Y. Zhu, "Olive: Accelerating large language models via hardware-friendly outlier-victim pair quantization," in *Proceedings of the 50th Annual International Symposium on Computer Architecture*, 2023, pp. 1–15.

[32] F. Liu, N. Yang, H. Li, Z. Wang, Z. Song, S. Pei, and L. Jiang, "Spark: Scalable and precision-aware acceleration of neural networks via efficient encoding," in *2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2024, pp. 1029–1042.

[33] B.-Y. Shi, Y.-C. Lo, A.-Y. Wu, and Y.-M. Tsai, "Post-training quantization for vision mamba with k-scaled quantization and reparameterization," in *2025 IEEE 35th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2025, pp. 1–6.

[34] A. Pierro and S. Abreu, "Mamba-ptq: Outlier channels in recurrent large language models," *arXiv preprint arXiv:2407.12397*, 2024.

[35] Y. Chen and A. Hemani, "Late breaking results: Quamba-se: Soft-edge quantizer for activations in state space models," 2026. [Online]. Available: https://arxiv.org/abs/2601.09451

[36] Y. Chen, H. Qin, Z. Zhang, M. Magno, L. Benini, and Y. Li, "Q-mambair: Accurate quantized mamba for efficient image restoration," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026, pp. 2524–2533.

[37] S. Lyu, Y. She, P. S. Y. Hung, R. C. C. Cheung, and W. Xu, "Vim-q: Scalable algorithm-hardware co-design for vision mamba model inference on fpga," 2026. [Online]. Available: https://arxiv.org/abs/2605.01935

[38] S. Ashkboos, A. Mohtashami, M. L. Croci, B. Li, P. Cameron, M. Jaggi, D. Alistarh, T. Hoefler, and J. Hensman, "Quarot: Outlier-free 4-bit inference in rotated llms," *Advances in Neural Information Processing Systems*, vol. 37, pp. 100 213–100 240, 2024.

[39] A. Wang, H. Shao, S. Ma, and Z. Wang, "Fastmamba: A high-speed and efficient mamba accelerator on fpga with accurate quantization," in *2025 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, vol. 1. IEEE, 2025, pp. 1–6.

[40] K. Zhou, H. Jiao, W. Huang, and Y. Huang, "An efficient fpga-based hardware accelerator of fully quantized mamba-2," in *2025 IEEE 33rd Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, 2025, pp. 217–226.

[41] K. Shi, Y. Cao, Y. Du, J. Chen, J. Yu, and K. Wang, "Tomamba: Towards token-efficient mamba architecture on fpga," in *2025 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 2025, pp. 1–9.

[42] D. Yoon, G. Lee, J. Chang, Y. Lee, D. Lee, and M. Rhu, "Mamba-x: An end-to-end vision mamba accelerator for edge computing devices," in *2025 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 2025, pp. 1–9.

[43] E. Park, D. Kim, and S. Yoo, "Energy-efficient neural network accelerator based on outlier-aware low-precision computation," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2018, pp. 688–698.

[44] R. Zhao, J. Dotzel, Z. Hu, P. Ivanov, C. De Sa, and Z. Zhang, "Overq: Opportunistic outlier quantization for neural network accelerators," *arXiv preprint arXiv:1910.06909*, 2019.

[45] A. H. Zadeh, I. Edo, O. M. Awad, and A. Moshovos, "Gobo: Quantizing attention-based nlp models for low latency and energy efficient inference," in *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2020, pp. 811–824.

[46] F. Liu, N. Yang, Z. Song, Z. Wang, H. Li, S. Huang, Z. Song, S. Pei, and L. Jiang, "Inspire: Accelerating deep neural networks via hardware-friendly index-pair encoding," in *Proceedings of the 61st ACM/IEEE Design Automation Conference*, 2024, pp. 1–6.

[47] A. Ramachandran, S. Kundu, and T. Krishna, "Microscopiq: Accelerating foundational models through outlier-aware microscaling quantization," in *Proceedings of the 52nd Annual International Symposium on Computer Architecture*, 2025, pp. 1193–1209.

[48] C. Hooper, C. Sakr, B. Keller, R. Venkatesan, K. Keutzer, S. Shao, and B. Khailany, "Fgmp: fine-grained mixed-precision weight and activation quantization for hardware-accelerated llm inference," *arXiv preprint arXiv:2504.14152*, 2025.

[49] X. Xie, L. Wang, L. Xiao, M. Han, L. Sun, S. Zheng, and X. Xu, "Fineq: Software-hardware co-design for low-bit fine-grained mixed-precision quantization of llms," in *2025 Design, Automation & Test in Europe Conference (DATE)*. IEEE, 2025, pp. 1–7.

[50] N. P. Pandey, J. Park, O. Gungor, F. Ponzina, and T. Rosing, "Qmc: Efficient slm edge inference via outlier-aware quantization and emergent memories co-design," *arXiv preprint arXiv:2601.14549*, 2026.

[51] D. Paperno, G. Kruszewski, A. Lazaridou, N.-Q. Pham, R. Bernardi, S. Pezzelle, M. Baroni, G. Boleda, and R. Fernández, "The lambada dataset: Word prediction requiring a broad discourse context," in *Proceedings of the 54th annual meeting of the association for*

computational linguistics (volume 1: Long papers), 2016, pp. 1525–1534.

Appendix A

Algorithms

A.1 Dual-Scale Offline Calibration

Algorithm 1 Dual-Scale Offline Calibration for Tensor u

Input: FP32 activation tensor $U = \{u_i\}$, quantization bit-width b, percentile threshold α

Output: main scale S_{main}, outlier scale S_{outlier}, packed dataset $\{(u_{q,i}, f_i)\}$

// Compute calibration statistics

$$X_\alpha \leftarrow \text{percentile}(|U|, \alpha)$$
$$X_{\max} \leftarrow \max(|U|)$$

// Derive quantization scales

$$S_{\text{main}} \leftarrow \frac{X_\alpha}{2^{b-1}-1}$$
$$S_{\text{outlier}} \leftarrow \frac{X_{\max}}{2^{b-1}-1}$$

// Per-element quantization

foreach $u_i \in U$ **do**

$$f_i \leftarrow (|u_i| > X_\alpha)\ ?\ 1 : 0$$
$$S_i \leftarrow (f_i = 1)\ ?\ S_{\text{outlier}} : S_{\text{main}}$$

$$u_{q,i} \leftarrow \text{clamp}\left(\text{round}\left(\frac{u_i}{S_i}\right), -(2^{b-1}-1), 2^{b-1}-1\right)$$

store $(u_{q,i}, f_i)$ as a 9-bit packed word

store S_{main} and S_{outlier} in PE configuration registers

A.2 SSM Processing Element: Per-Token Inference

Algorithm 2 SSM Processing Element: Per-Token Inference (Part a — Scalar Stage)

Input: INT8 inputs $\{u_q, f, dt_q, z_q, D_q\}$; FP32 scales $\{S_u^{\text{main}}, S_u^{\text{out}}, S_{dt}, S_D, S_z\}$; `bool` *enable_gating*

Output: Latched intermediates $\{dt_{\text{sp}}, dt_u, Du, silu_z\}$; initialized accumulator `accum_reg`

`// ST_SCALAR stage (31 cycles)`

8 $$S_u \leftarrow (f = 1)\ ?\ S_u^{\text{out}} : S_u^{\text{main}}$$; `// Dual-Scale MUX selection`

9 $$u_{\text{fp32}} \leftarrow \text{INT8toFP32}(u_q) \cdot S_u$$
$$dt_{\text{fp32}} \leftarrow \text{INT8toFP32}(dt_q) \cdot S_{dt}$$
$$z_{\text{fp32}} \leftarrow \text{INT8toFP32}(z_q) \cdot S_z$$

10 $$dt_{\text{sp}} \leftarrow \text{softplus}\left(dt_{\text{fp32}} + \delta_{\text{bias}}\right)$$

11 $$dt_u \leftarrow dt_{\text{sp}} \cdot u_{\text{fp32}}$$
$$Du \leftarrow \text{INT8toFP32}(D_q) \cdot S_D \cdot u_{\text{fp32}}$$

12 $$silu_z \leftarrow \frac{z_{\text{fp32}}}{1 + \exp(-z_{\text{fp32}})}$$

13 latch $\{dt_{\text{sp}}, dt_u, Du, silu_z\}$

14 initialize `accum_reg` $\leftarrow 0$

Algorithm 3 SSM Processing Element: Per-Token Inference (Part b — State Loop and Combine; continued from Algorithm 2)

Input: Latched intermediates $\{dt_{\text{sp}}, dt_u, Du, silu_z\}$ and `accum_reg` from Algorithm 2; INT8 inputs $\{B_q[N], C_q[N]\}$ via register files $A_{\text{rf}}, B_{\text{rf}}, C_{\text{rf}}$; FP32 scales $\{S_A, S_B, S_C\}$; previous hidden state $h_{\text{prev}}[N]$; `bool` *enable_gating*

Output: FP32 output y_{fp32}, updated hidden state $h_{\text{next}}[N]$

`// ST_STATE_LOOP stage (79 cycles, II = 3)`

for $n \leftarrow 0$ **to** $N-1$ **do**

$$A_{\text{fp32}} \leftarrow \text{INT8toFP32}(A_{\text{rf}}[n]) \cdot S_A$$

$$\bar{A}_n \leftarrow \exp\left(-\exp(A_{\text{fp32}}) \cdot dt_{\text{sp}}\right)$$

$$h_{\text{next}}[n] \leftarrow \bar{A}_n \cdot h_{\text{prev}}[n] + dt_u$$

$$B_{\text{fp32}} \leftarrow \text{INT8toFP32}(B_{\text{rf}}[n]) \cdot S_B$$
$$C_{\text{fp32}} \leftarrow \text{INT8toFP32}(C_{\text{rf}}[n]) \cdot S_C$$

$$\texttt{accum_reg} \leftarrow \texttt{accum_reg} + B_{\text{fp32}} \cdot C_{\text{fp32}} \cdot h_{\text{next}}[n]$$

write $h_{\text{next}}[N]$ back to state register file

`// ST_COMBINE stage (9 cycles)`

$$y_{\text{pre}} \leftarrow \texttt{accum_reg} + Du$$
$$y_{\text{gate}} \leftarrow y_{\text{pre}} \cdot silu_z$$

$$y_{\text{fp32}} \leftarrow enable_gating \; ? \; y_{\text{gate}} : y_{\text{pre}}$$

A.3 Validity Propagation via `vld_pipe` Shift Register

Algorithm 4 Validity Propagation via `vld_pipe` Shift Register

Input: `loop_fire` pulse; pipeline constants `H_WRITE_STAGE`= 25, `BCH_VALID_STAGE`= 29

Output: registered control signals `h_write_en`, `bch_valid`

```
// Cycle-accurate validity propagation
25
                vld_pipe[0] ← loop_fire
26 for k ← 1 to 29 do
27 |
   |            vld_pipe[k] ← vld_pipe[k − 1]
   |
   // Generate gated control signals
28
        h_write_en ← vld_pipe[H_WRITE_STAGE]
         bch_valid ← vld_pipe[BCH_VALID_STAGE]
```

A.4 Parameterized Synthesis Sweep for Fair Area and Timing Comparison

Algorithm 5 Parameterized Synthesis Sweep for Fair Area and Timing Comparison

Input: bit-width set $W = \{8, 9, 11\}$; 22nm technology node; target clock frequency $F_{\text{clk}} = 1$ GHz

Output: area report $area[w]$ and timing report $slack[w]$ for each $w \in W$

compile FP32 backend as a `dont_touch` hierarchy
; `// Freezes MAC, Exp, Add, Div, and Softplus blocks`

foreach $w \in W$ **do**

  set top-level parameter `INPUT_WIDTH` $\leftarrow w$
  ; `// Only INTx-to-FP32 frontend logic changes`
  run Design Compiler synthesis

$$area[w] \leftarrow \texttt{report_area -hierarchy}$$
$$slack[w] \leftarrow \texttt{report_timing -max_paths 20}$$

`// Synthesize DS variant separately`
run synthesis for `ssm_compute_core_ds`

$$area[\text{DS}] \leftarrow \texttt{report_area -hierarchy}$$
$$slack[\text{DS}] \leftarrow \texttt{report_timing -max_paths 20}$$

`// Area delta relative to INT8 baseline`

$$\Delta area[w] \leftarrow area[w] - area[8]$$

for each $w \in \{9, 11, \text{DS}\}$

TRITA-EECS-EX-2026:388
Stockholm, Sweden 2026

www.kth.se