

Implementation and Evaluation of Stable Diffusion on a General-Purpose **CGLA** Accelerator

Takuto Ando, Yu Eto, Yasuhiko Nakashima

Nara Institute of Science and Technology (NAIST)



AI's Energy Problem & Our Solution

Problem: AI relies on power-hungry GPGPUs.

GPGPU: High Flexibility 😄 , Low Efficiency 😞

Specialized ASIC: High Efficiency 😄 , Low Flexibility 😞



Solution: CGRA - Bridging the Gap

We focus on CGRAs (Coarse-Grained Reconfigurable Arrays) as an architecture to solve the trade-off.

- ▲ **Efficiency:** They achieve ASIC-like efficiency by directly mapping dataflows.
- ▲ **Flexibility:** They retain flexibility by dynamically reconfiguring the datapath.

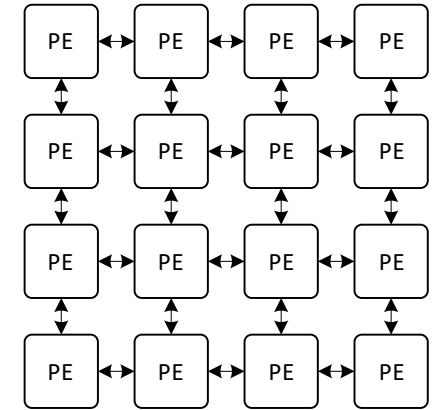
Our CGLA “IMAX” ...What’s this? 🤔

2

Conventional CGRA

Concept: Reconfigurable at the PEs level on a 2D array

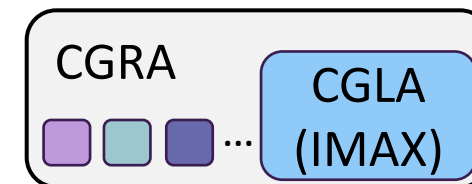
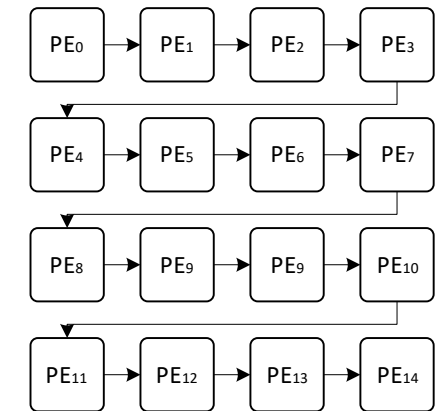
Challenge: Complex routing algorithms lead to long compilation times



IMAX Architecture (CGLA: Coarse-Grained Linear Array)

Concept: PEs are arranged in a linear structure, fast compilation

Design: Interleaves 64 PEs with 512KB of local cache memory to minimize slow main memory access



➡ The Next Step: Deployment to **Stable-Diffusion (Gen Image)**

Stable-Diffusion Challenges



What is **Stable-Diffusion**?

A latent diffusion model capable of generating images from text.

Gen Image models are highly accurate but computationally massive.

Prior Stable-Diffusion Accelerators 🤪



Focused on specialized hardware (ASICs/FPGAs) [1-2].

When new models emerged, this hardware became **obsolete**.

Our Objective

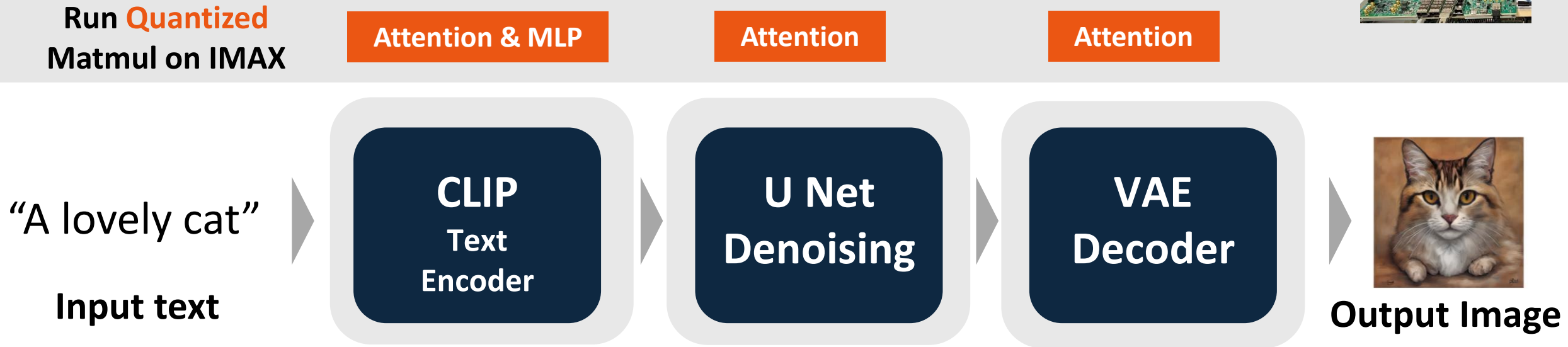
Implementing quantized operations of Stable Diffusion (Gen ImageAI)
on IMAX3 to reveal its actual performance and potential when ASIC-ized.

[1] H. Zhou, et.al, "SDAcc:A stable diffusion accelerator on FPGA via software-hardware co-design," 2024.

[2] G. Yang, et.al, "SDA: Low-bit Stable Diffusion acceleration on edge FPGAs," 2024.

Stable-Diffusion System Overview

Standard Stable-Diffusion (Gen Image) Process:



Execute **quantization Matrix-Multiplication** of each stage on IMAX.

The initial step is to exclusively use the existing quantized **Q3_K** and **Q8_0**.

TABLE I: Breakdown of execution time in dot-product kernel

Quantized Type	F32	F16	Q3_K	Q8_0
Q3_K Model	30.7 %	59.0 %	10.3 %	–
Q8_0 Model	21.8 %	62.0 %	–	16.3 %

Actually, there are more FP kernels.

Mat Mul (A x B) Kernel Implementation

5

Q3_K kernel

1. Load:

18(Int-8) input data and **Q3_K data**.

2. Restructure (via OP_CVT53):

Weight expansion by weights and scale-data within Q3K data.

3. Compute (Float):

Performs the **multiply-accumulate operation in 18 (int8)**.

4. Store:

Sum the accumulation results and store them.

Q8_0 kernel

1. Load:

18(Int-8) quantized data (A,B).

2. Compute (Int):

SIMD **8-bit Integer Dot Product** (18 x 18).

3. Accumulate (Int):

Accumulates the parallel dot product results as 32-bit integers.

4. Decode (Dequantize):

Converts the final integer sum into an FP32 using the loaded scale and a LUT.

IMAX enhancements

Key Optimization Techniques

▲ In-line Conversion:

PEs perform decode on-the-fly.

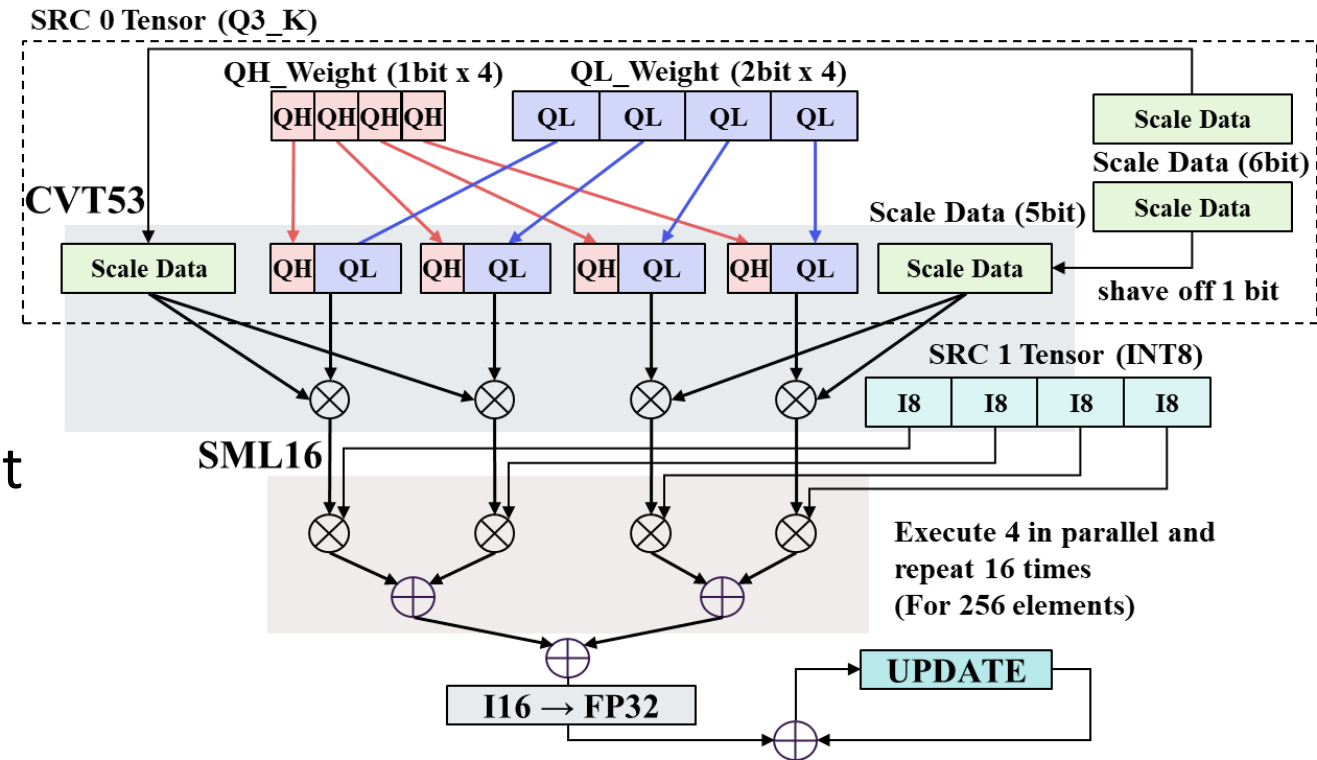
▲ SIMD Execution:

The 64-bit datapath processes two 32-bit operations in parallel.

▲ Column-wise Multithreading:

A single physical FPU executes

4 logical multiply-add operations through time-multiplexing.

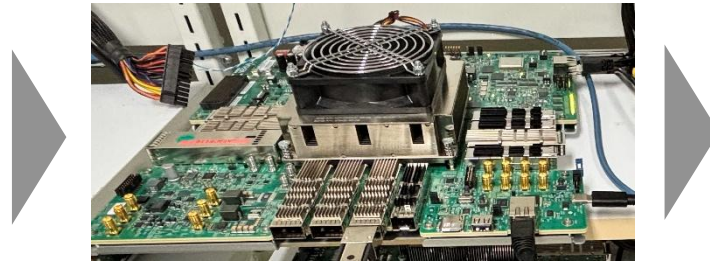


Extract one column out of 4

Experimental Setup

Workload: Generation of 512x512 images using the SD-Turbo model.

“A lovely cat”



VPK180 FPGA (145 MHz) 2-thread



512 x 512

Two key metrics for **the end-to-end transcription task**:

- ▲ **Performance:** The *E2E latency required to transcribe the audio file, measured in sec.
- ▲ **Energy Efficiency:** The Power-Delay Product (PDP), calculated as $Latency (s) \times Power (W)$.

Performance and power were projected for a 28nm ASIC implementation, allowing for a **realistic 6x frequency increase to 840 MHz** (FPGA:145MHz).

E2E Latency

▲ GPUs are Fastest:

- IMAX3: 790.3s.
- ASIC Projection: 754.5s.
- Xeon / GPU: Significantly faster

▲ IMAX is slow because the FP kernels:

- Account for a large part of the computation, are executed on the CPU.
- If we could run the FP kernel on IMAX, it would be faster.

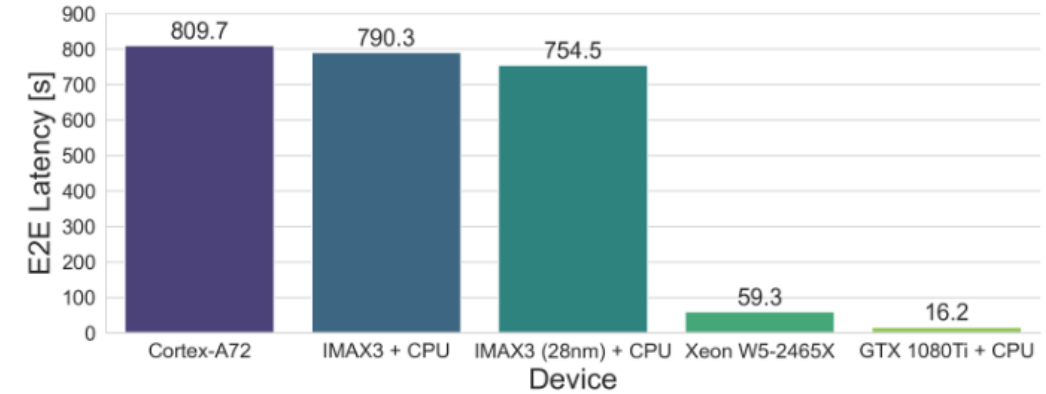


Fig. 6: End-to-end latency comparison by device for Q3_K model inference.

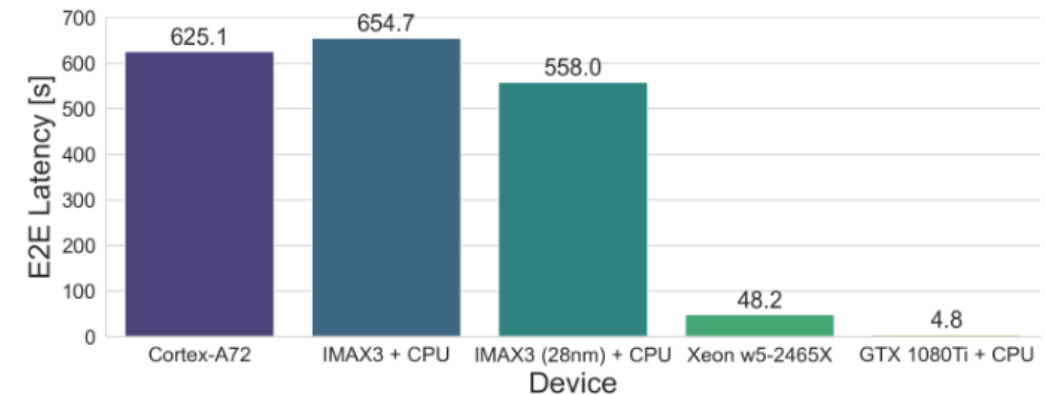


Fig. 7: End-to-end latency comparison by device for Q8_0 model inference.

E2E latency comparison by device

Energy Efficiency (PDP)

$$PDP = \text{Latency (s)} \times \text{Power Consumption (W)}$$

**lower is better*

ASIC IMAX3 (28nm) :

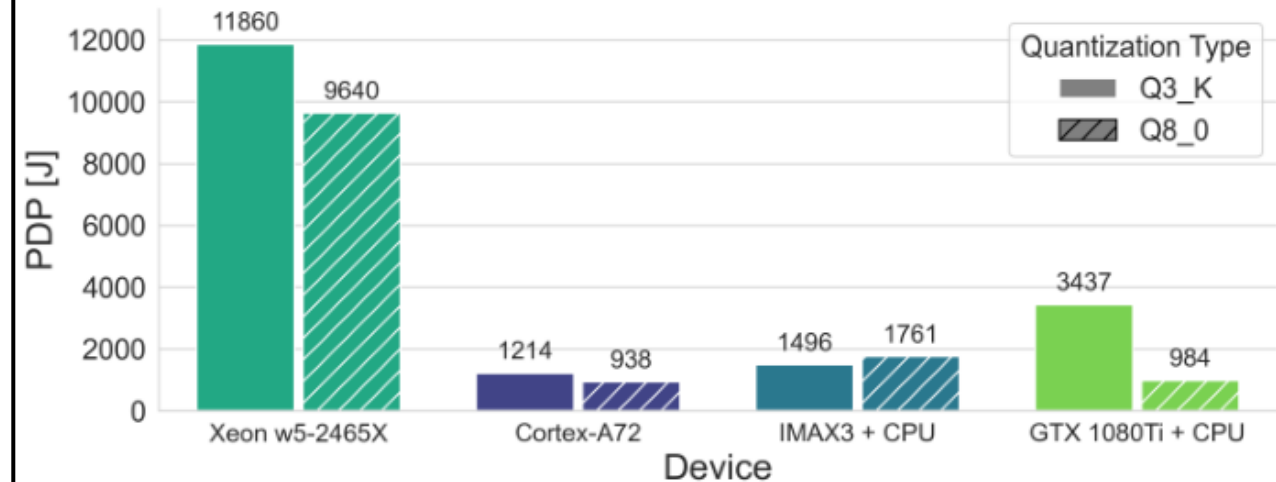
more efficient than the Xeon CPU

for both quantization models.

Surpassing GPU:

For the Q3_K model,

more efficient than the GTX 1080 Ti.



PDP performance comparison by device (lower is better)

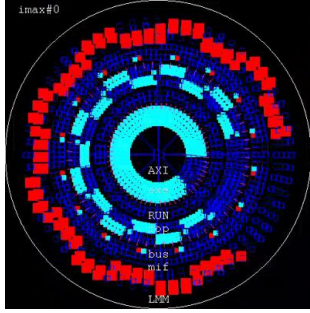
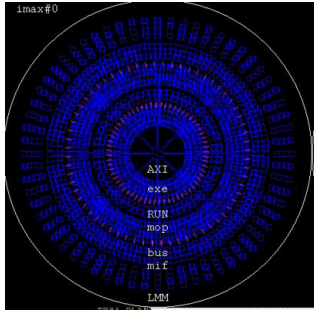
Cortex-A72:	0.6 W
GTX 1080 Ti:	250 W
Xeon w5-2465X:	200 W
IMAX3 (28nm):	47.7W (Q8_0) / 52.8 W (Q3_K)

It uses 25% the power of a GPU,

outperforming in total energy efficiency (PDP) despite longer processing times.

Summary

<u>Background</u>	AI's high energy demand requires efficient hardware alternatives to power-hungry GPUs.
<u>Our Approach</u>	Image generation with existing IMAX quantization kernels using Matrix-Multiplication implementation.
<u>Key Results</u>	ASIC projections achieved superior energy efficiency (PDP) compared to a GPU (Q3_K)
<u>Conclusion</u>	CGLA demonstrates potential as a highly energy-efficient platform for image generation.
<u>Future Work</u>	Increase the offload ratio Resolve host bottlenecks for better performance, alongside evaluating scalability with larger resolutions.



IMAX execution visualization DEMO

Thanks!

Do you have any **questions?**



Full Paper



Lab Page



Takuto ANDO



Yu ETO



Yasuhiko NAKASHIMA