

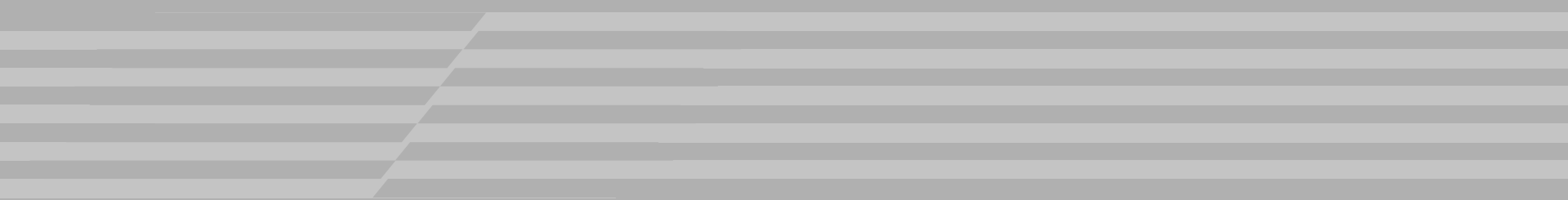
TransCoder: Efficient Hardware Implementation of Transformer Encoder

Sangki Park, Chan-Hoon Kim, Soo-Min Rho,
Jeong-Hyun Kim, Seo-Ho Chung, Ki-Seok Chung*

Hanyang University, Seoul, Korea

Jan. 22. 2024

Contents

- Introduction
 - TransCoder Architecture
 - Experiments and Results
 - Conclusion
- 
- A series of horizontal lines in various shades of gray, some solid and some dashed, creating a decorative pattern at the bottom of the slide.

Introduction

- Self-Attention is key operation in Transformer
- However self-attention is major bottleneck
- Softmax is expensive function for HW
 - Previous works use **floating-point** for softmax
 - Need **dequantization** if model is quantized
- Solutions?
 - Use only **fixed-point** in self-attention layer computation
 - Need **fixed-point based softmax** function

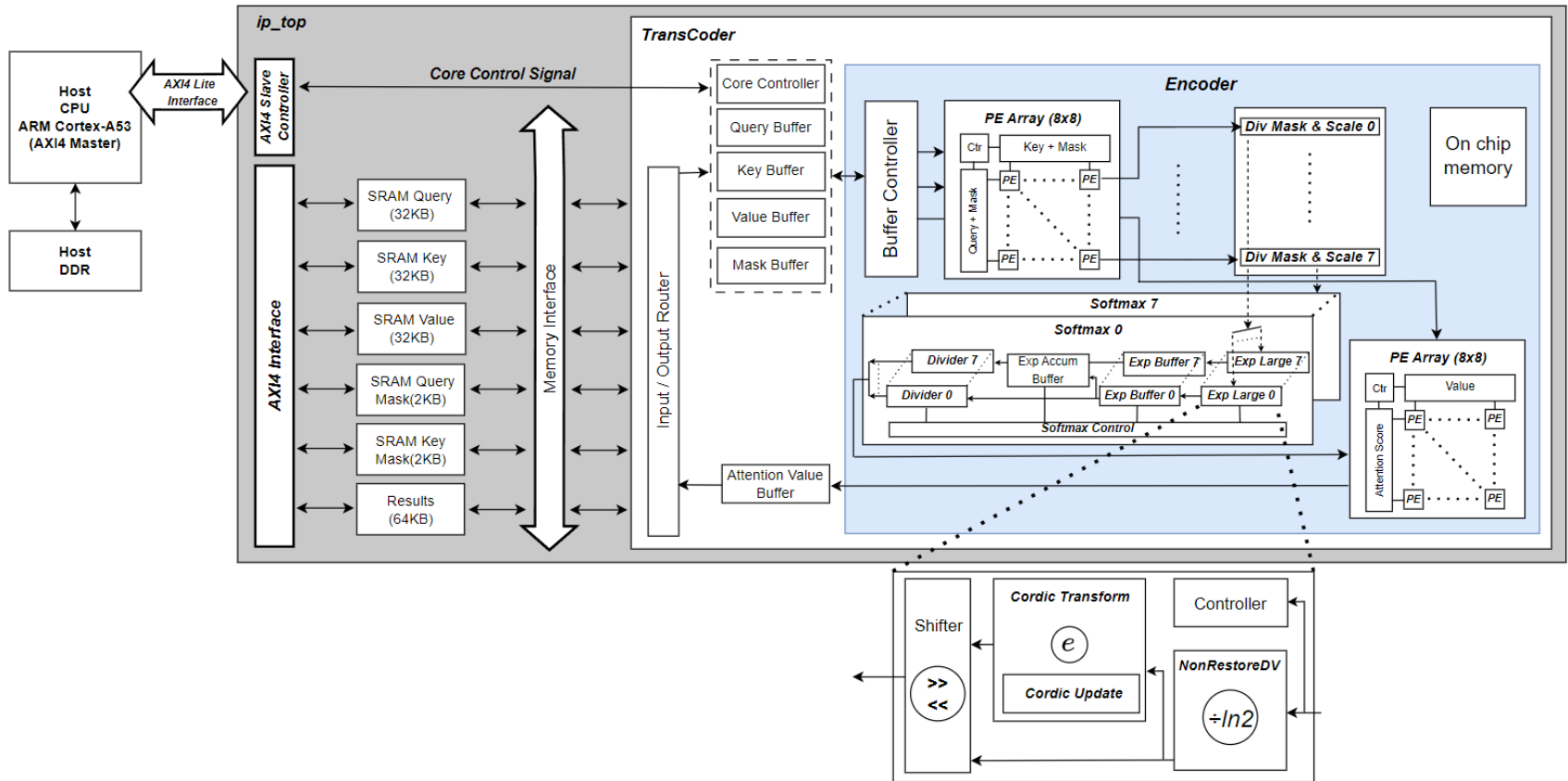
Introduction

- Self-Attention is key operation in Transformer
- However self-attention is major bottleneck
- Softmax is expensive function for HW
 - Previous works use **floating-point** for softmax
 - Need **dequantization** if model is quantized
- Solutions?
 - Use only **fixed-point** in self-attention layer computation
 - Need **fixed-point based softmax** function

 **TransCoder**

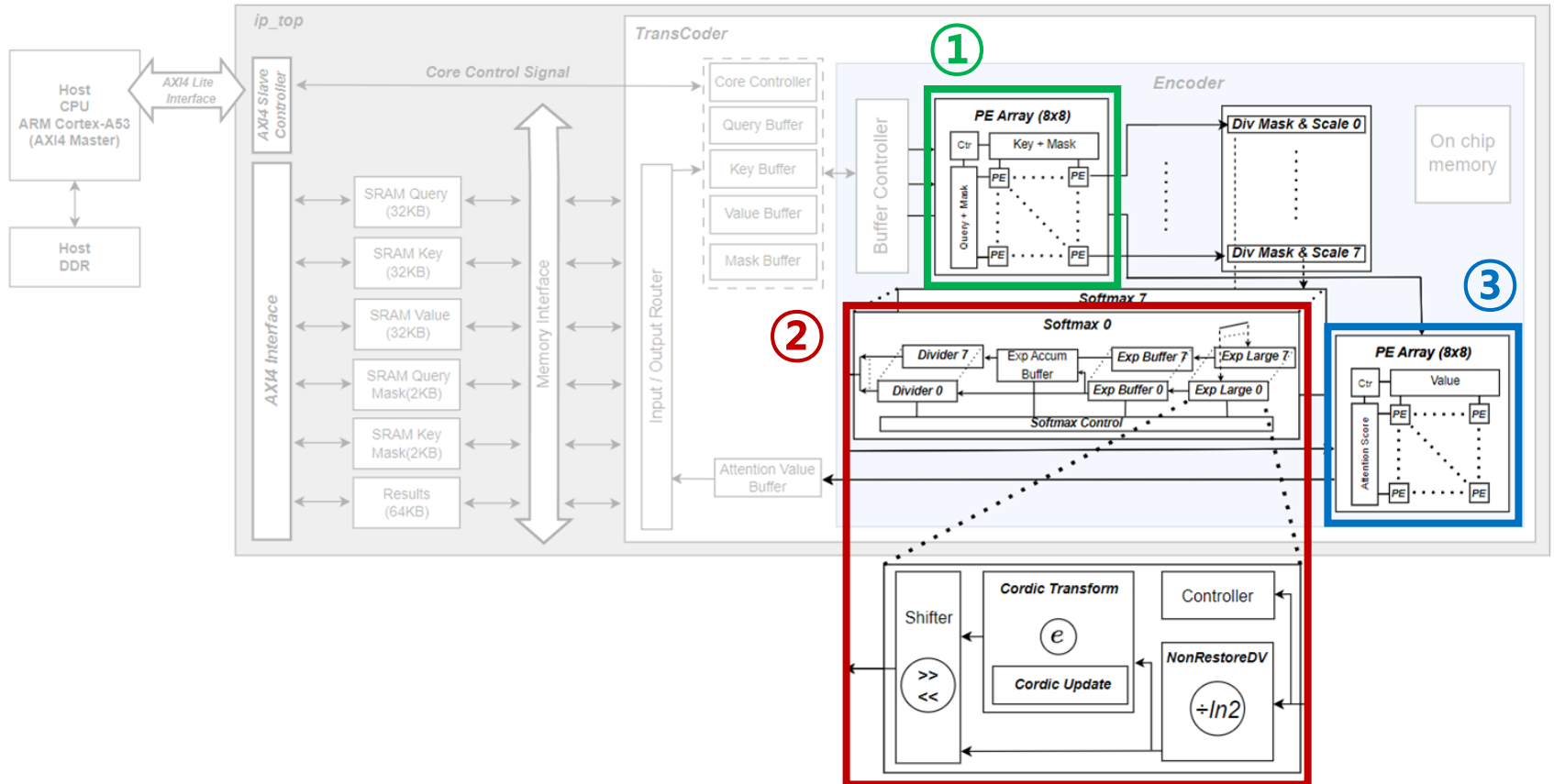
TransCoder Architecture

- Overview



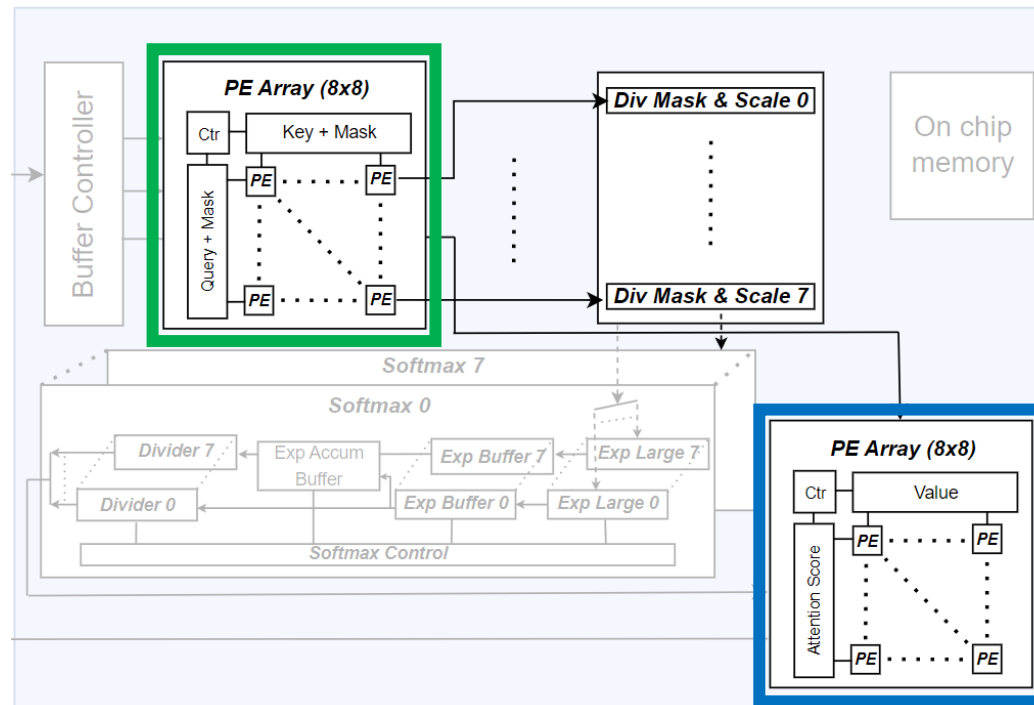
TransCoder Architecture

- Overview
 - Process Flow



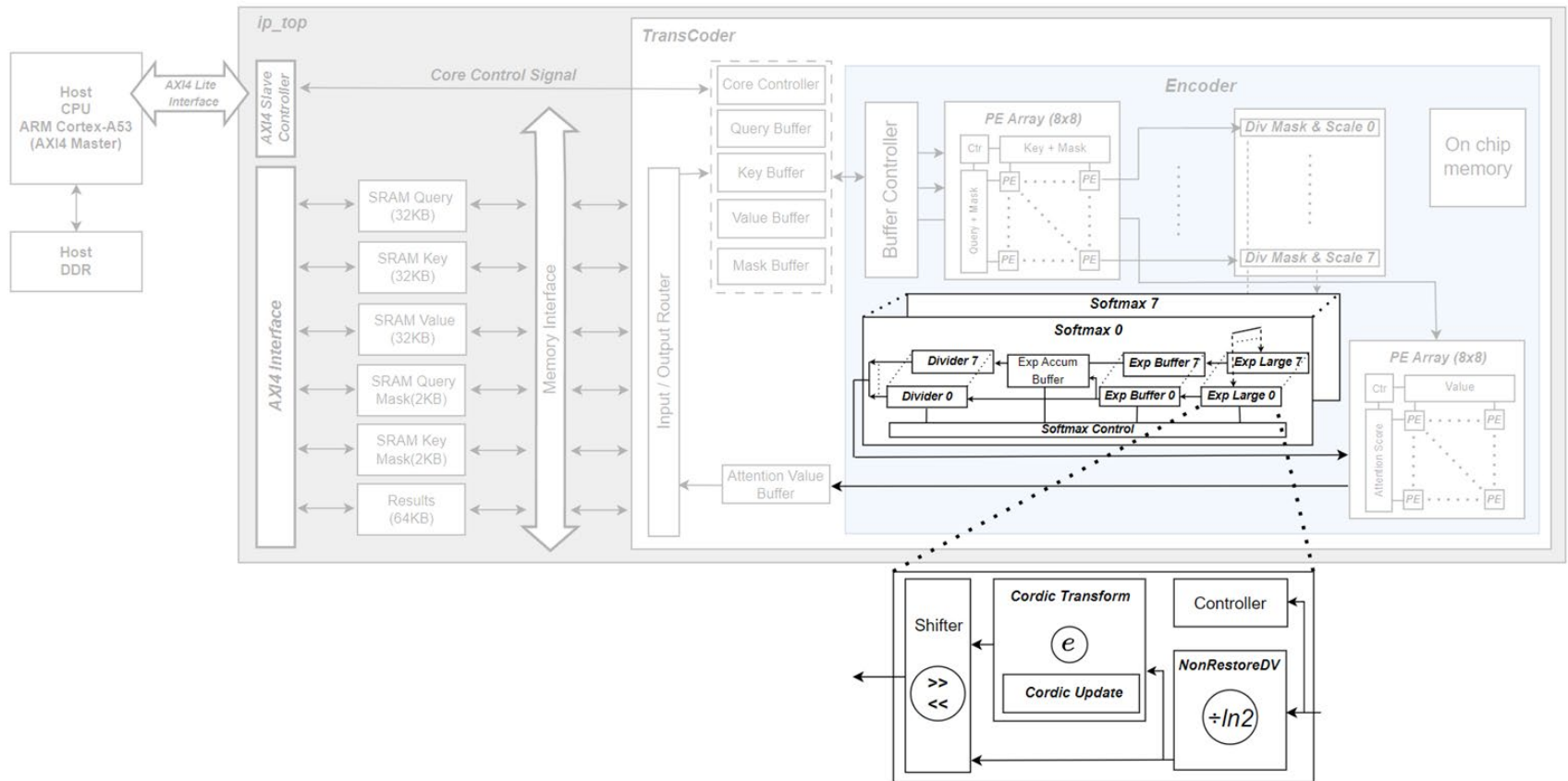
TransCoder Architecture

- Systolic Array
 - Row-wise parallel computation
 - Two 8x8 PE array for:
 - Attention score: $QK^T / \sqrt{d_k}$,
 - Context layer: $\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) * V$



TransCoder Architecture

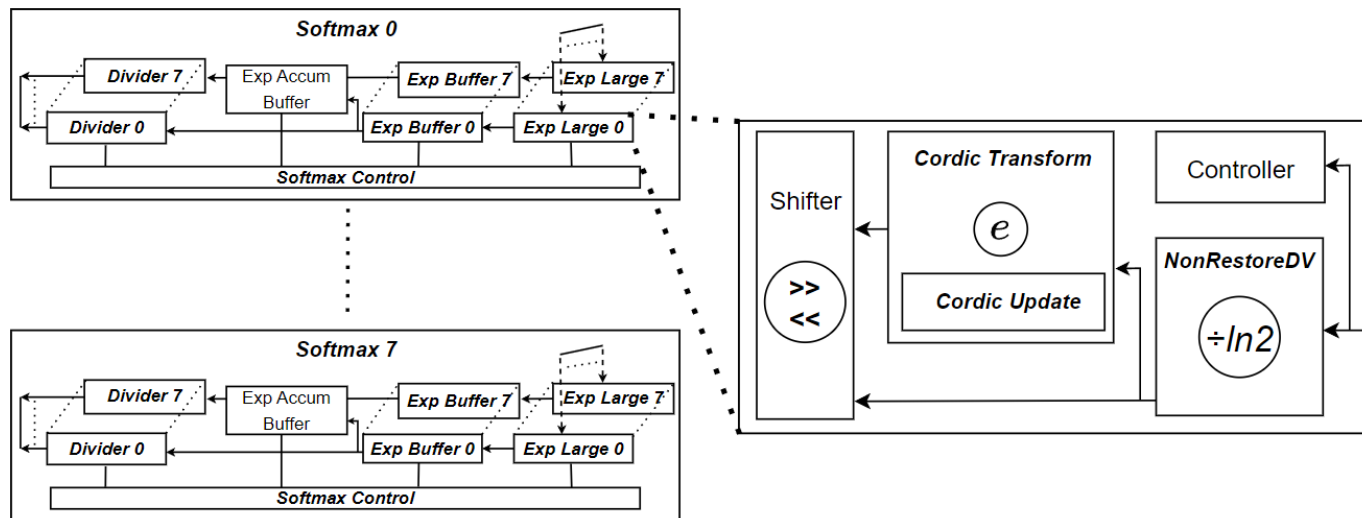
- ExCORDIC



TransCoder Architecture

- ExCORDIC

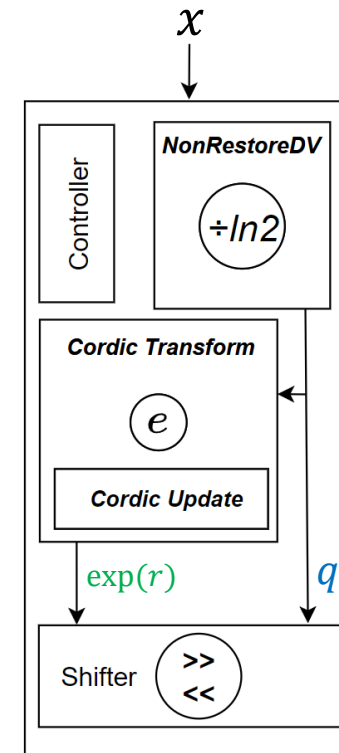
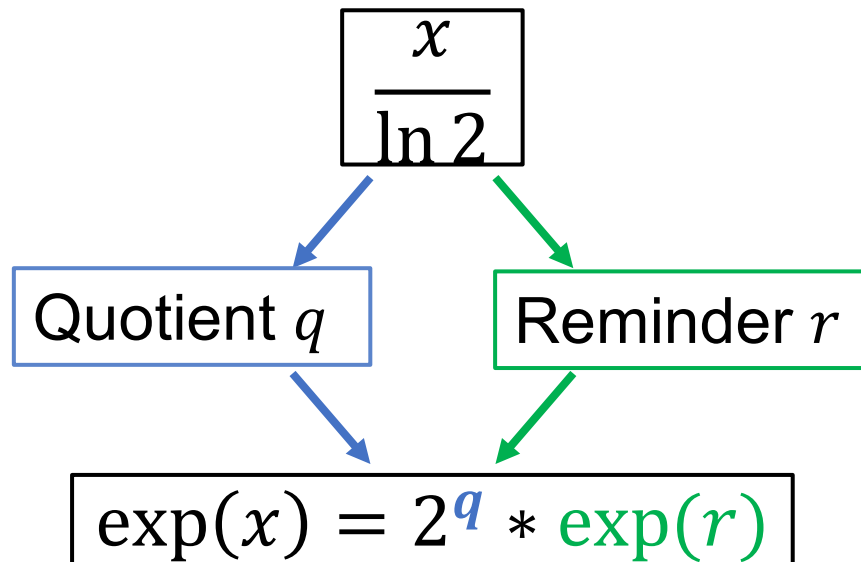
- Integer based softmax approximation
- Use 8 modules for row-wise parallel computation
- Extended CORDIC algorithm for wide input range



TransCoder Architecture

- ExCORDIC

- Extended CORDIC algorithm for wide input range
 - Input range of conventional CORDIC is limited to $[-1, +1]$
 - ExCORDIC uses quotient as scale factor
 - Extend the input range to $[-20, +20]$



Experiments and Results

- Evaluation Methodology
 - Written in Verilog HDL
 - Target board: Xilinx ZCU111 FPGA
 - Target models: BERT-tiny, BERT-Large
 - Datasets: GLUE tasks
- HW: Vivado & Vitis 2019.2
- SW: PyTorch & Transformers with custom API
- HW-SW integration with PYNQ

Experiments and Results

- Performance on GLUE


		CoLA	MNLI	MNLI (MM)	MRPC	QNLI	RTE	SST-2	STSB*	WNLI
BERT Large	FP32	63	86.2	86.3	85.8	92.7	69.3	93	0.888	45.1
	Ours	63	86	85.1	85.2	91.7	69	92.4	0.886	45
BERT tiny	FP32	62	65	67	68.4	77.3	59	80.9	0.53	44
	Ours	65.1	64.9	63	68.4	77.3	58	80.7	0.53	44

Experiments and Results

- Hardware Resource

	DSP	BRAM	LUT	Freq.	Power
Ye, et al.	1044	260	132K	300 MHz	19.3 W
Ours	128	16	126K	215 MHz	7.3 W

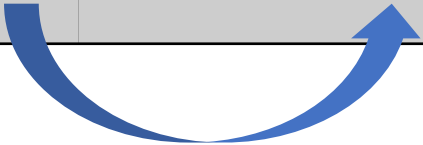
x2.64
less power



Experiments and Results

- Latency Comparison
 - Single attention module latency

	ARM A53	Ours
Latency (ms)	11.7766	2.8209



x4.17 Faster

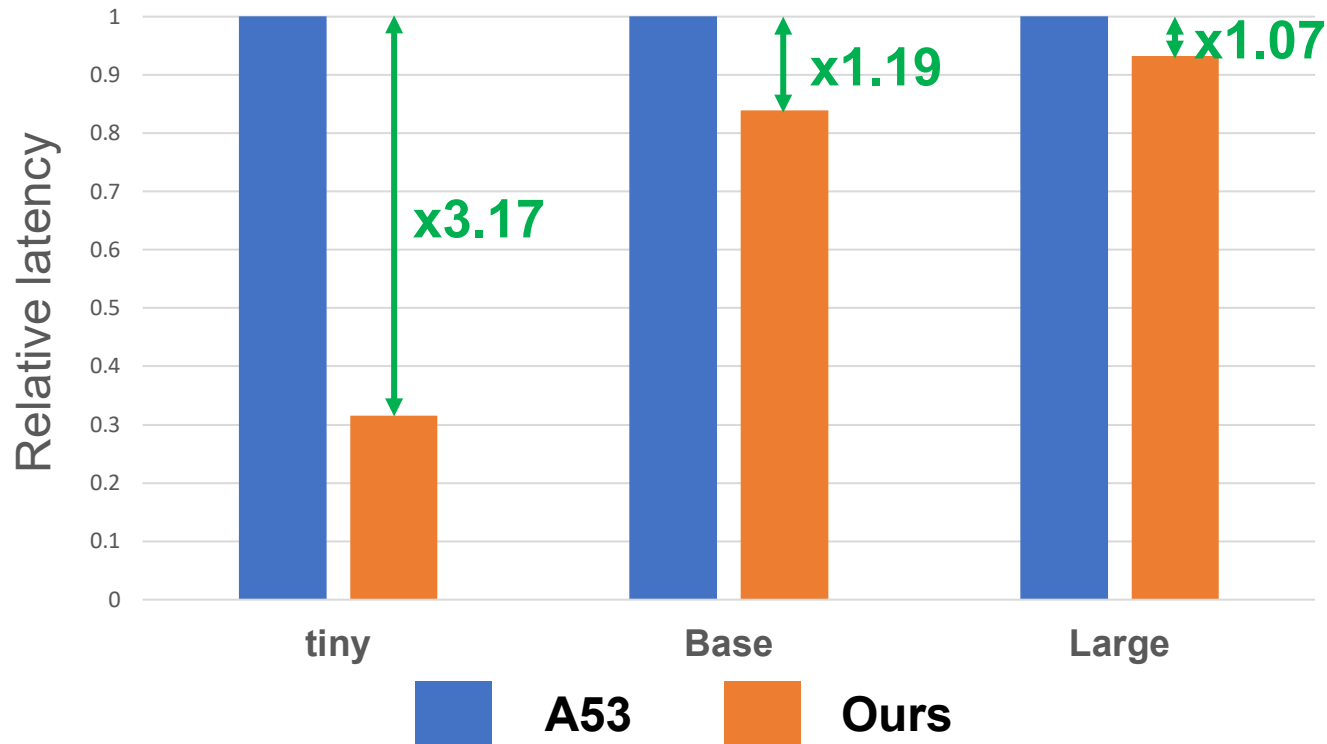
Experiments and Results

- Latency Comparison
 - End-to-end model latency on ZCU111
 - Average of 1000 samples from CoLA valid set

	ARM A53 (s)	Ours (s)
BERT-tiny	0.2324	0.0732
BERT-Base	2.8915	2.425
BERT-Large	8.82	8.22

Experiments and Results

- Latency Comparison
 - End-to-end model latency on ZCU111



Conclusion

- TransCoder: Hardware Accelerator for Self-Attention
 - **Fixed-point** based GEMM and Softmax
 - Only **1.2%** accuracy drop compared to FP32 BERT-Large
 - **x3.17 faster** on end-to-end model latency than A53
 - **x2.64 power effective** than previous work

Q&A
