SSTP: A 13.32 TFLOPS/W End-to-End Simultaneous Speech Translation Processor on Edge Devices

Bokyoung Seo[1], Ghangmin Yun[2], Chaeyoon Kim[2], Jueun Jung[2], and Kyuho Jason Lee[2]
[1]UNIST, Ulsan, Republic of Korea / [2]Yonsei University, Seoul, Republic of Korea
Email: kyoung8523@unist.ac.kr / kyuho.jsn.lee@yonsei.ac.kr

***Abstract*—Real-time cross-lingual communication requires on-device simultaneous speech-to-text translation (SimulS2T), yet no existing accelerator supports tightly-coupled execution of automatic speech recognition and machine translation that comprise SimulS2T. In addition, prior SimulS2T models rely on the compute-intensive Transformer architecture, whose $O(N^2)$ computational cost makes realization of SimulS2T on battery-constrained edge devices impractical. Furthermore, the on-the-fly policy-based operation of SimulS2T causes the irregular iteration of encoder and decoder stage, disrupting the chunk-level pipelining for real-time execution. To solve these challenges, this work presents an energy-efficient processor (SSTP) for real-time end-to-end SimulS2T system on edge devices co-optimized with the proposed Fourier Transform (FT)-accelerated SimulS2T model. The reconfigurable domain fetch unit and logarithmic number system-based PE with approximated log/linear converter efficiently support the mixed real-/complex-valued FT computation, while the context reuse management unit and policy-based task allocator enable chunk-level pipelined execution. Fabricated in 28nm CMOS, the proposed SSTP enables the end-to-end SimulS2T with 13.32 TFLOPS/W.**

I. Introduction

As the demand for seamless cross-lingual communication grows, on-device Simultaneous Speech-to-Text Translation (SimulS2T), which translates streaming input concurrently while speaker talks, has emerged as a pivotal technology [1]. Previously, it has been dominated by sequentially cascaded architectures; automatic speech recognition (ASR) followed by machine translation (MT). While this approach achieves high accuracy when provided with full-length input speech, it suffers from inherent error propagation and increased latency because of inter-model dependency [2]. Recent end-to-end SimulS2T models [3], [4] resolve the dependency by directly mapping chunk-wise data of frame-wise segmented streaming speech to translated text using Transformer only. However, $O(N^2)$ complexity of such models scales with speech length, leading to increased computation and energy consumption that limit operating time on battery-constrained edge devices, such as smart glasses and wireless earbuds. As an example case from [4], ~12.6 GFLOPS per chunk results in at most ~1 hour of operation of SimulS2T solely without AP under < 0.2 Wh energy budget. Nonetheless, prior accelerators only support either lightweight RNN-based ASR [5], [6] limited to short speech (< 3 sec) or compute-intensive Transformer-based MT [7], but have not addressed end-to-end SimulS2T.

To resolve these limitations, this paper presents a dedicated processor with hardware-efficient SimulS2T system optimization on edge devices. As shown in Fig. 1(a), the proposed Fourier Transform (FT)-accelerated SimulS2T (FT-SS2T) architecture replaces the costly attention module [4] of $O(N^2)$ with efficient FT-based modules of $O(N\log N)$. It processes streaming input in chunk units through a 12-layer encoder with FNet [8] and a 4-layer decoder with H3 [9],

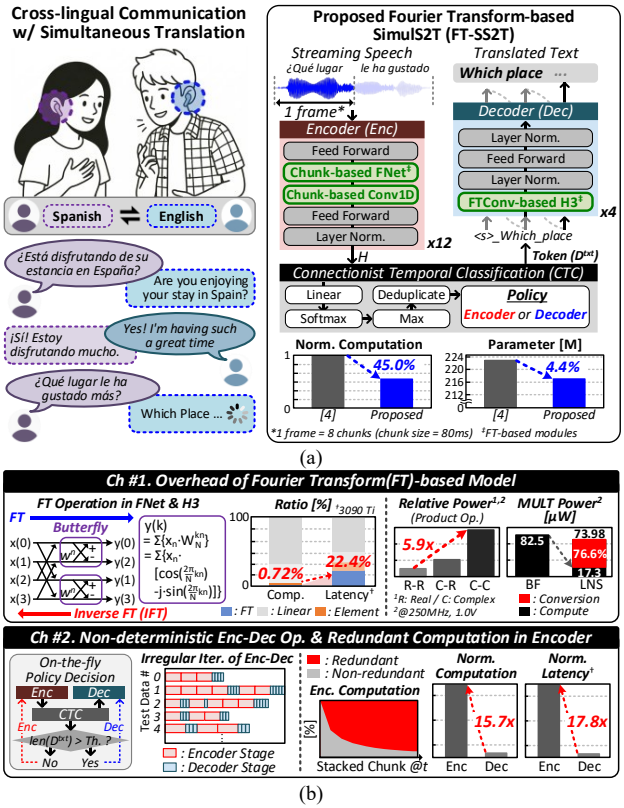


Fig. 1. (a) Overview of the FT-SS2T and (b) hardware challenges.

coordinated by a policy-based connectionist temporal classification (CTC). By reducing the computational overhead of encoder dominated by attention operation, FT-SS2T achieves a 45% of reduction in computation over [4] with a comparable parameter size.

Nevertheless, the hardware acceleration of FT-SS2T faces two challenges, as shown in Fig. 1(b). First, unlike attention, hardware-inefficient butterfly operations in FT incur distinct irregular memory access and computational cost. The memory bottleneck is caused by frequent data reordering, which alone consumes 22.4% of total latency despite 0.72% of negligible computation on GPUs. Meanwhile, since frequency-domain computation is particularly sensitive to accumulated errors, bfloat (BF) precision is essential for FT-based modules. In addition, complex-valued product in FT with costly BF further increases power, incurring 5.9× higher power than real-valued counterparts. Although logarithmic number system (LNS) mitigates *compute* power [10], precise log/ linear conversion emerges as a severe bottleneck, dominating 76.6% of the overall power. Second, SimulS2T inherently suffers from its non-deterministic Enc-Dec operation as well as the redundant computations in the encoder. The non-deterministic operation results from the irregular iteration of each encoder and decoder stage driven by on-the-fly policy decisions made by

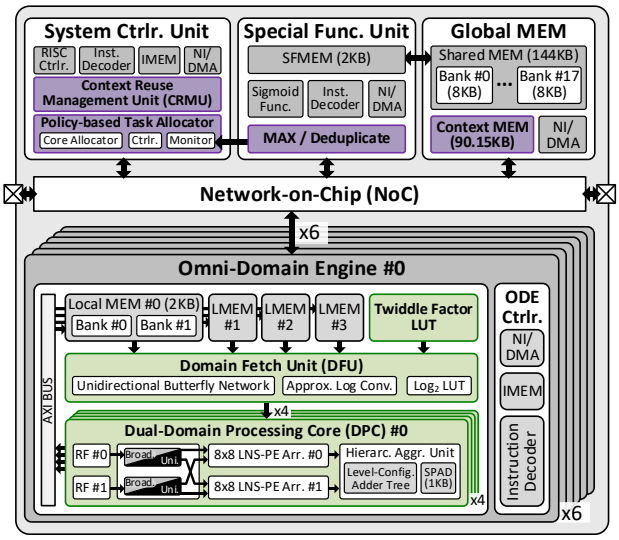


Fig. 2. Overall architecture of the SSTP.

CTC, hindering efficient chunk-level pipelining. In addition, encoder stacks the input chunks over time while processing them to capture accurate contextual information, leading to exponentially increasing redundant computations. As a result, the encoder consumes 15.7× higher computations and 17.8× higher latency than the decoder, leading to frequent pipeline stalls that directly degrade overall system throughput and preclude seamless real-time SimulS2T execution.

For practical implementation of a SimulS2T system, the proposed Simultaneous Speech Translation Processor (SSTP) introduces three key features: 1) Domain Fetch Unit (DFU) with fused-FT method to support reconfigurable dataflow for FT and tensor arithmetic in FT-SS2T; 2) LNS-PE with Approximated Log Converter (ALoC) and Approximated Linear Converter (ALiC) to reduce BF16 computation power by reducing LNS conversion power; 3) Context Reuse Management Unit (CRMU) to reduce the encoder overhead and Policy-based Task Allocator (PTA) to enable chunk-level pipelining of the entire SimulS2T system.

II. Proposed Simultaneous Speech Translation Processor (SSTP)

Fig. 2 describes the overall architecture of SSTP, which consists of a system controller unit (SCU), a special function unit, a global memory (GMEM) and 6 Omni-Domain Engines (ODEs). The ODE includes a twiddle-factor (TF) LUT, 4 local memories (LMEMs), a DFU, and 4 dual-domain processing cores (DPCs). Each DPC contains two sets of register file (RFs) and 8×8 LNS-PE arrays connected through an interconnect that supports mixed Real/Complex operands (C-C, R-R, R-C) using reconfigurable broadcast and unicast executions. Following BF16 computation in two 8×8 LNS-PE arrays, the hierarchical aggregation unit (HAU) aggregates results using a level-configurable adder tree and stores them in the double-buffered SPAD.

A. Domain Fetch Unit with Fused Fourier Transform

To support the efficient execution of FT-SS2T, the ODE must handle two distinct computational workloads: *fourier transform* and *tensor arithmetic*. As shown in Fig. 3, a common method for FT acceleration is Bailey's FT [11], which replaces inefficient butterfly by decomposing 1D sequence of length *N* onto a square 2D matrix for FT to exploit

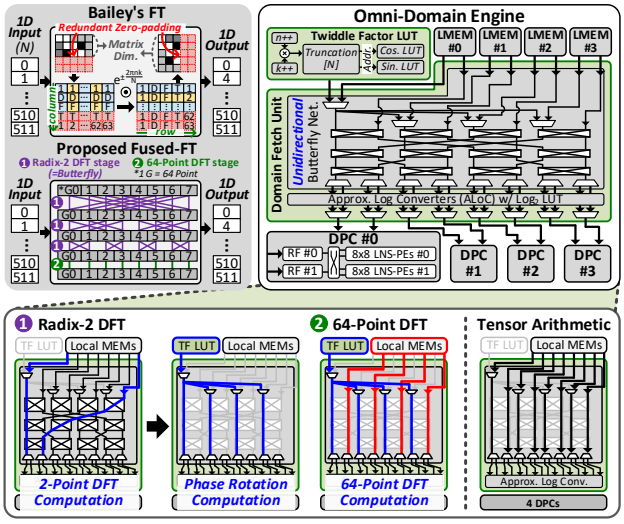


Fig. 3. The architecture and dataflow of DFU in ODE with the proposed fused-Fourier Transform (FT) method.

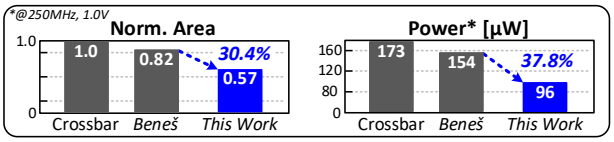


Fig. 4. Area and power reduction of DFU with *unidirectional* butterfly.

spatial parallelism across PE arrays. However, it demands an exact match between the matrix dimension and the PE array size to avoid redundant computations caused by meaningless zero-padding, which directly degrades hardware utilization in the PE array. To ensure full utilization regardless of input length, the proposed fused-FT processes 1D sequence grouped according to the PE array size. It then performs radix-2 DFT stages followed by a 64-point DFT mapped to the 8×8 LNS-PE array, reducing the number of required stages.

However, FT and inverse FT (IFT) in H3 layer (Fig. 1(b)) require the reversed reordering patterns, forcing the usage of costly reconfigurable networks such as crossbar or Beneš [12] in DFU to support the bidirectional dataflow. This overhead can be eliminated, as the intermediate computations between FT and IFT are *element-wise* and independent of data ordering, thereby enabling an identical *unidirectional* butterfly pattern. Therefore, DFU supports FT-SS2T workloads via a 2x2 switch-based unidirectional butterfly network, routing operands and TFs from LMEMs and TF LUT to RFs in each DPC. During radix-2 DFT, the butterfly network routes a 16b operand from each LMEM bank to compute 2-input DFT, then streams the corresponding 16b complex TFs from TF LUT for phase rotation. The 64-point DFT is performed by transmitting operands from LMEMs and TF LUT. For tensor arithmetic, DFU directly routes eight 32b data from LMEMs to each RF. As a result, the DFU handles all FT-SS2T workload efficiently, reducing area by 30.4% and power by 37.8% compared to the Beneš network, as shown in Fig. 4.

B. LNS-PE with Approximated Log/Linear Converter

Typically, LNS reduces the computational power of BF by mapping operands into the log domain, where a single linear-MULT is reduced to two log-ADDs over the exponent and mantissa components, as expressed in (1).

$$\log(AB) = \{(-1)^{sign_A} \cdot (-1)^{sign_B} \cdot (e_A + e_B - 2 \cdot bias) + log_2(1.f_A) + log_2(1.f_B)\} \quad \ldots (1)$$

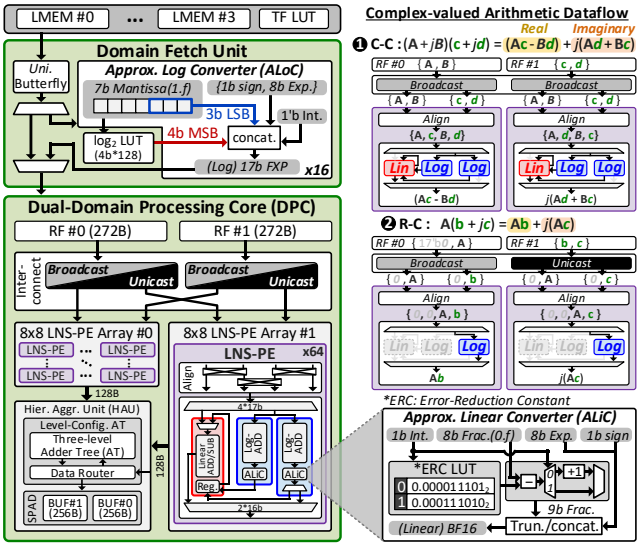


Fig. 5. The architecture of ALoC in DFU and LNS-PE in DPC, and the dataflow of DPC under complex-valued arithmetic (C-C, R-C).

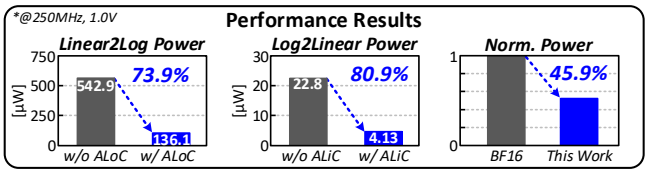


Fig. 6. Power reduction of ODE with ALoC and ALiC.

However, this benefit diminishes in complex-valued LNS MULT that consists of four log-ADDs followed by linear-ADD/SUB to combine real and imaginary partial products as in (2). Therefore, it inevitably requires power-intensive intermediate linear domain conversion.

$$\Sigma(C \cdot D) = \Sigma\{C_{re}D_{re} - C_{im}D_{im} + \boldsymbol{j}(C_{re}D_{im} + C_{im}D_{re})\} \quad \ldots (2)$$

Since complex-valued computation involves both real and imaginary parts, the interconnect and 8×8 LNS-PE arrays in DPC must operate differently for each operand combination. Detailed dataflows for C-C and R-C are illustrated in Fig. 5. For instance, in C-C MAC, the interconnect broadcasts each complex value across the 8×8 LNS-PE array for separate computation of real and imaginary partial products of (2). Each LNS-PE aligns the operands and proceeds log-ADDs followed by linear-ADD/SUB to execute MULT. Finally, the results from 8x8 LNS-PE arrays are accumulated in the 3-level adder tree in HAU and stored in SPAD through data router.

As linear-domain conversion is required per PE, ALiC must be placed inside LNS-PE. Meanwhile, ALoC is placed in the DFU to perform log conversion during data fetch to each DPC. With sufficient area margin, ALoC adopts a LUT-based design for low approximation error. However, storing precise $\log_2$ values for all 7b BF16 mantissas incurs excessive LUT size (5.25KB for ODEs). To reduce this overhead, 16 ALoCs share a $\log_2$ LUT storing 4b MSBs of precise $\log_2$ values. Each ALoC converts the ***1.f*** mantissa into an 8b fixed point in [0,1) using a LUT-based 4b MSB and a Mitchell [13] approximated 3b LSB with extra 1b, yielding a final 17b fixed point. In contrast, ALiC converts the LNS result, which is a 1b INT and ***0.f*** fraction in [0, 2), back to BF16 using a two-region piecewise linear approximation over [0, 1) and (1, 2], each applying a distinct error-reduction constant. As shown in Fig. 6, the ODE with ALoC and ALiC reduces conversion power

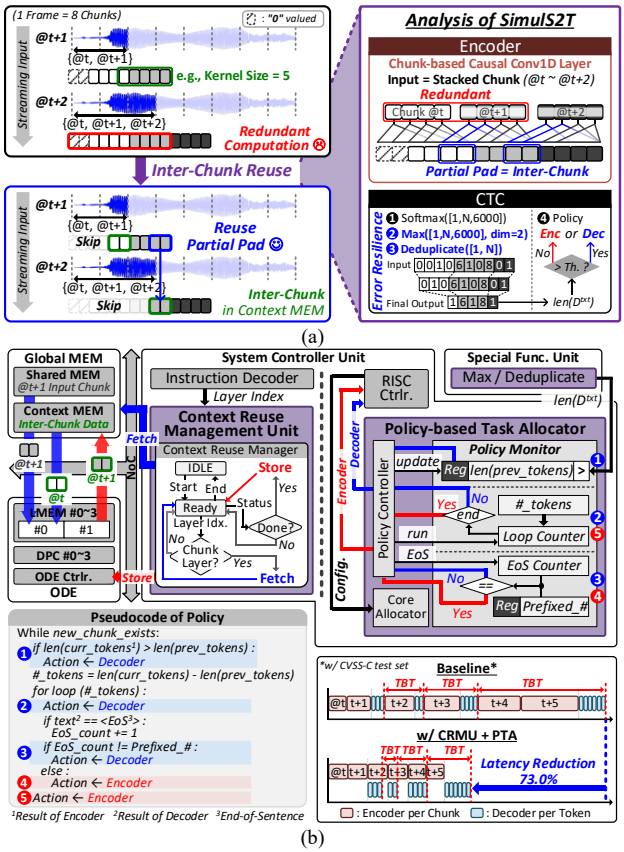


Fig. 7. (a) Redundancy removal via inter-chunk reuse. (b) The architecture of Context Reuse Management Unit and Policy-based Task Allocator.

by 73.9% and 80.9% compared to precise conversion, and 45.9% of overall power compared to BF16.

C. *Context Reuse Management Unit & Policy-based Task Allocator for Chunk-level Pipeline*

As shown in Fig. 7(a), the redundancy of encoder incurs with the successively stacked chunks. However, the chunk-based causal Conv1D layer in the encoder depends only on partial padded data from the previous chunk. Therefore, FT-SS2T reuses the inter-chunk data and eliminates redundant computation by skipping preceding data. It does not incur any accuracy loss, as CTC determines the final output token sequence through max and deduplication operations, which suppress the perturbations. Despite this improvement, the unpredictable policy decisions introduce a system-level bottleneck, requiring runtime-adaptive execution scheduling.

To handle both challenges, SCU jointly incorporates CRMU and PTA, as illustrated in Fig. 7(b). By monitoring the current layer index, CRMU identifies chunk-based layers and controls the usage of inter-chunk data. First, it fetches the kernel-size inter-chunk data from the GMEM's context memory into the ODE's double-buffered LMEM, followed by fetching the current chunk data from the shared memory into the identical LMEM. After the computation in DPCs, the current chunk data is stored back into the context memory to be reused as next inter-chunk data. Meanwhile, PTA enables chunk-level pipelining between encoder and decoder stages through five policy rules described in the pseudocode. The policy monitor compares the current token length derived from SFU's max and deduplicate unit against the previous token length to determine decoder execution, while loop and end-of-sentence counter decide whether to continue decoder.

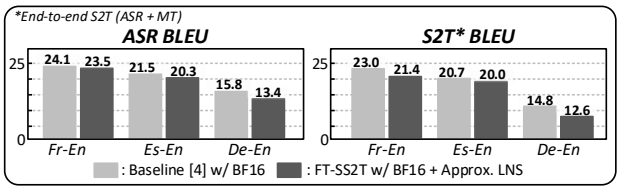
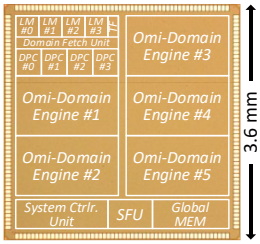


Fig. 8. Performance of the proposed FT-SS2T.



Specifications	
Technology	28nm CMOS
Die Area	12.96 mm²
Supply Voltage	0.75 – 1.0 V
Frequency	50 – 250 MHz
SRAM	308.9 KB
Peak Perform.	2.304 TFLOPS
Proposed SimulS2T Performance	
(*Enc*) Energy / Frame†	22.24 mJ
(*Dec*) Energy / Token	0.925 mJ

†Frame = 640ms

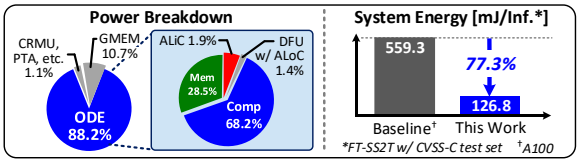


Fig. 9. Chip photograph and measurement results.

Using this policy, the RISC controller coordinates encoder and decoder execution on ODEs by following the predefined configuration. When the core allocator in PTA receives the configuration, it performs resource allocation by assigning full ODEs for encoder- or decoder-only, and partitioning ODEs for parallel execution according to the predefined allocation. As a result, the combined deployment of CRMU and PTA achieves a 73.0% reduction in end-to-end system latency.

III. Measurement Results

Fig. 8 shows the performance results of the proposed FT-SS2T on various cross-lingual S2T tasks of CVSS-C dataset [14], evaluated using ASR BLEU for encoder and S2T BLEU for end-to-end model. Compared to the baseline [4] with BF16 precision, FT-SS2T with BF16 and approximated LNS exhibits marginal BLEU degradation of no more than 2.4 points for ASR and 2.2 points for S2T. Although the minor degradation arises from approximate modeling of token-wise dependency in FNet and H3, it is far outweighed by the 45% reduction in computation with O(Nlog$N$) complexity.

Fig. 9 shows the chip photograph and measurement results. Fabricated in 28nm CMOS, SSTP occupies 12.96 mm² with 308.9KB SRAM and achieves 2.304 TFLOPS at 250 MHz and 1.0V. The measured energy consumption is 22.24 mJ/frame for the encoder and 0.925 mJ/token for the decoder. As shown in the power breakdown, ODE dominates power consumption at 88.2%, while the proposed features including DFU with ALoC and ALiC account for only 3.3% overhead. Nevertheless, these features enable a 77.3% system energy reduction over the GPU. Table I shows the comparisons with prior accelerators [5], [6] and the commercial computing platform, A100. SSTP achieves the highest energy and area efficiency of 13.32 TFLOPS/W and 0.18 TFLOPS/mm², respectively. Unlike [5], [6], which target short sequence, single-language ASR with RNN-based models, SSTP fully supports long sequence (> 10 sec), multilingual SimulS2T , including both ASR and MT. Even though A100 supports end-to-end model execution, SSTP reduces energy consumption by 3.16× for encoder and 4.27× for decoder.

TABLE I. Performance Comparison Table

		[5]	[6]	A100	This Work
Process [nm]		16 nm	28 nm	7 nm	28nm
Application		ASR[1] (Enc. Only)	ASR[1] (Enc. Only)	ASR[1]+MT[2]	***ASR[1] + MT[2] (end-to-end)***
Model		MRF+RNN	BLiGRU	FT-SS2T	**FT-SS2T**
Data type		FP8	INT8	BF16	**LNS16[3]-BF16**
Area [mm²]		21.8	9.58	826	12.96
SRAM [KB]		9800	-	-	308.9
Operating [V] / [MHz]		0.55–1.0 / 130–775	0.6–1.1 / 1.25–100	N/A / 1,410	0.72–1.0 / 50–250
Power [mW]		111	1.3	400,000	34.6–395.7
Energy Efficiency		2.6–7.8 TFLOPS/W	N/A	N/A	**5.82–13.32 TFLOPS/W**
Area Efficiency		0.13 TFLOPS/mm²	N/A	N/A	**0.18 TFLOPS/mm²**
Simultaneous Speech Translation System Performance					
ASR[1] (Enc.)	Energy /Frame [mJ]	2,247	12.7	70.34	**22.24**
MT[2] (Dec.)	Energy /Token [mJ]	***N/A***		3.948	**0.925**

[1]ASR : Automatic Speech Recognition [2]MT : Machine Translation [3]LNS w/ approximated conversion

IV. Conclusion

This paper presents SSTP, a dedicated processor co-optimized with the FT-SS2T architecture for real-time end-to-end SimulS2T system on edge devices. Through hardware-aware software optimization with FT-based modules, SSTP achieves a 45% reduction in computation. To enable efficient hardware implementation of FT-SS2T, fused-FT method with DFU supports distinct FT and tensor arithmetic dataflows, achieving 30.4% area and 37.8% power reduction. In addition, ALoC and ALiC in ODE reduce 45.9% of computation power than BF16, by reducing the dominated conversion power in LNS computation. By reusing the previous inter-chunk data and enabling chunk-level pipelining, CRMU and PTA reduce 73% of system latency. As a result, SSTP achieves 13.32 TFLOPS/W energy efficiency and operates end-to-end SimulS2T with 22.24 mJ/frame for ASR and 0.925 mJ/token for MT, which has not been enabled for prior processors.

References

[1] R. Yao, Google, Dec., 12, 2025. [Online]. Available: https://blog.google/products-and-platforms/products/search/gemini-capabilities-translation-upgrades/

[2] M. Sperber and M. Paulik, "Speech Translation and the End-to-End Promise: Taking Stock of Where We Are", In ACL, 2020.

[3] X. Lieu et al., "Recent Advances in End-to-End Simultaneous Speech Translation", In IJCAI, 2024.

[4] S. Zhang et al., "Streamspeech: Simultaneous speech-to-speech translation with multi-task learning", In ACL, 2024.

[5] T. Tambe et al., "A 25 mm² SoC for IoT devices with 18 ms noise-robust speech-to-text latency via Bayesian speech denoising and attention based sequence-to-sequence DNN speech recognition in 16nm FinFET", In IEEE ISSCC, 2021.

[6] Y.-H. Tsai et al., "A 28-nm 1.3-mW speech-to-text accelerator for edge AI devices", In IEEE JSSC, vol. 59, no. 11, Nov. 2024.

[7] S. Kim et al., "C-Transformer: An Energy-Efficient Homogeneous DNN-Transformer/SNN-Transformer Processor for Large Language Models", In IEEE JSSC, vol. 60, no. 10, Oct. 2025.

[8] J. Lee-Thorp, J. Ainslie, I. Eckstein, and S. Ontañón, "FNet: Mixing Tokens with Fourier Transforms", In NAACL, 2022.

[9] D. Fu et al., "Hungry hungry hippos: Towards language modeling with state space models", In ICLR, 2023.

[10] X. Geng et al., "Beyond Uniform Approximation: Towards Efficient Computing in Logarithmic Number System for Large Language Models", In IEEE TCAD, Feb. 2026.

[11] D. Fu, H. Kumbong, E. Nguyen, and C. Ré, "FlashFFTConv: Efficient Convolutions for Long Sequences with Tensor Cores", In ICLR, 2024.

[12] J. Lu, J. Tian, H. Li, I. Young, and Z. Zhang, "FETTA: Flexible and Efficient Hardware Accelerator for Tensorized Neural Network Training", In IEEE TCAD, Jan. 2026.

[13] J. N. Mitchell, "Computer multiplication and division using binary logarithms", In IRE TEC, vol. EC-11, no. 4, 1962.

[14] Y. Jia, M. T. Ramanovich, Q. Wang, and H. Zen, "CVSS corpus and massively multilingual speech-to-speech translation", In LREC, 2022.