Thanos: Energy-Efficient Keyword Spotting Processor with Hybrid Time-Feature-Frequency-Domain Zero-Skipping

Sangyeon Kim[†], Hyunmin Kim[†], Sungju Ryu[*]
[†]Department of Electronic Engineering, [*]Department of System Semiconductor Engineering,
Sogang University, Seoul, Republic of Korea
{sangyeonkim, hyunminkim, sungju}@sogang.ac.kr

***Abstract*—In recent years, the keyword spotting algorithm has gained significant attention for applications such as personalized virtual assistants. However, the keyword spotting system must be always turned on to listen to the input voice for the recognition, which worsens the battery constraint problem in the edge devices. In this paper, we first analyze the sparsities in the keyword spotting computation. Based on the characteristic, we introduce the keyword spotting processor called Thanos to enable the zero-skipping scheme in the multiple keyword spotting domains to mitigate the burdensome energy consumption. Experimental results show that our hybrid-domain zero-skipping scheme reduces the latency by 80.3-87.3% and energy consumption by 48.1-79.8% on average, respectively, over the baseline architecture.**

***Index Terms*—Keyword spotting, sparsity, zero-skipping, hardware accelerator, neural processing unit.**

I. Introduction

Keyword spotting (KWS) tasks are typically performed on battery-constrained edge devices, including smartphones, smartwatches, and various wearables, prompting the exploration of several energy-saving techniques such as temporal convolution [1], AAD-KWS [2], and skip-score-based dynamic power gating [3]. Meanwhile, regarding the feature extraction, we analyzed the following potential improvement points using implicit/explicit zero values. 1) In the KWS, a large number of input samples in each input frame do not contribute to classifying keywords, and they are sometimes from noise sources. Hence, we can reduce the number of ineffective computations by thresholding and masking them with zero values. 2) If all the samples in a single input feature are zero in the time-domain computation, we can skip the rest of the feature extraction and backend DNN stages. 3) Mel filter matrix mostly consists of zero values, so we can directly skip the meaningless multiply-by-zero computations.

Our main contributions of this work are outlined as follows:

1) We analyze the implicit/explicit sparsity in the keyword spotting algorithm.
2) We present the hybrid time-feature-frequency domain zero-skipping scheme exploiting the sparsity on our analysis.
3) We introduce a Thanos hardware architecture to maximize the performance of keyword spotting application tailored to the hybrid domain zero-skipping.

*Corresponding Author

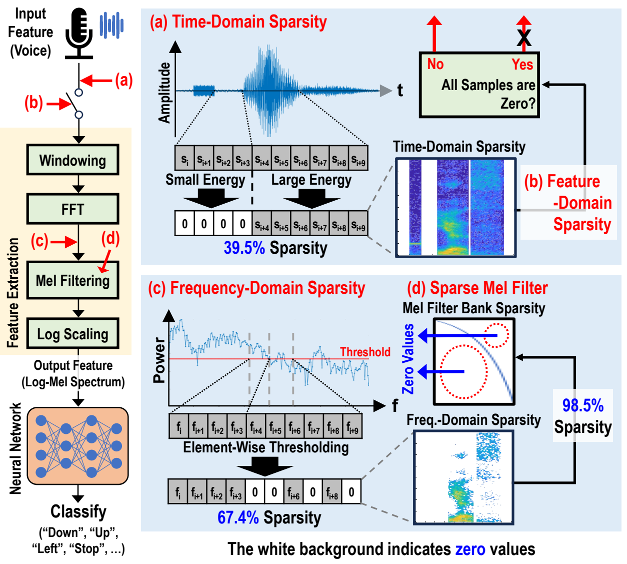

Fig. 1. Overview of proposed hybrid domain zero-skipping scheme. Sparsity values are from Google Speech Commands Dataset (GSCD). Our observation of sparsities in keyword spotting application: (a, b) Time-domain sparsity due to low amplitude values, (c) Frequency-domain sparsity from less critical values, and (d) Mel filter sparsity with 98.5% zero values. The white background indicates zero values.

II. Preliminary: Keyword Spotting on Hardware

The KWS algorithm (Fig. 1) consists of a front-end (preprcessing) feature extraction stage, which includes framing, windowing, Fourier transform, Mel filtering, and Log scaling, and a back-end (post-processing) DNN stage for keyword classification.

A. *Previous KWS Hardware*

AAD-KWS [2] implemented the clock-gating scheme for the AI computation block to maximize the energy-efficiency in the case of silent circumstances. However, it is widely known that the most of the energy is typically consumed in the non-AI computation phase [4], and hence it obviously has limitations in the optimization of energy consumption.

Seol et al. [3] proposed a skip controller that reduces a large number of computations on a frame-by-frame basis. However, the original neural network must be modified to the recurrent neural network-based algorithm.

Kosuge et al. [5] proposed a wired-logic-based nonlinear neural network with a pruning scheme. However, such a structure omits the feature extraction stage and it can compute

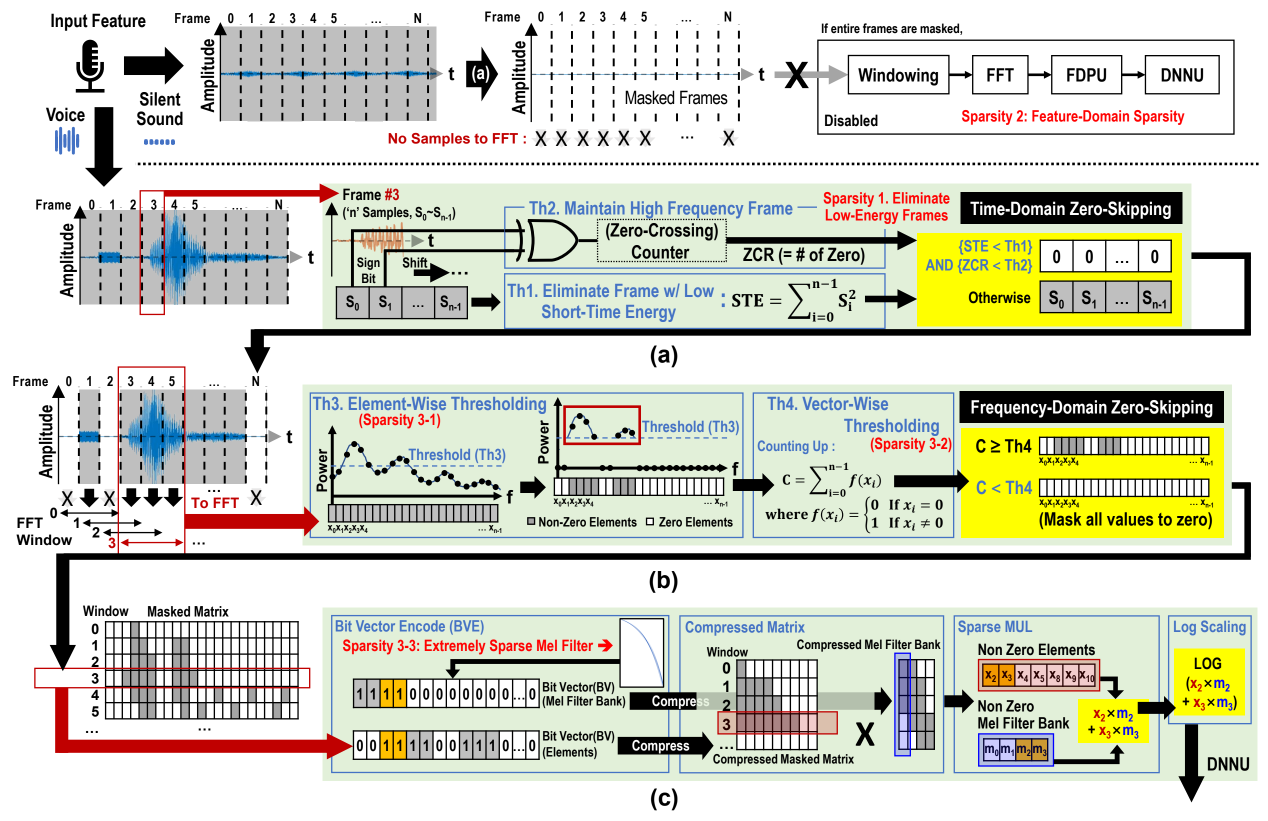
Fig. 2. (a) Time-domain processing unit (TDPU) and (b-c) frequency-domain processing unit (FDPU).

the customized model only. Hence, such a design is not compatible with general hardware design.

III. Proposed Architecture: Thanos

In this Section, we introduce our energy-efficient keyword spotting processor called Thanos. The key idea behind our approach is to mask less important numbers with zeros and to eliminate the redundant zero computations in each domain, thereby reducing the number of MAC computations for 3 domains (time, feature, and frequency) and maximizing the energy-efficiency of the KWS system. First, we overview the key idea of the proposed hybrid domain zero-skipping method. Then, the microarchitectures for the domain-wise processing units are described, and top-level Thanos architecture is finally explained.

A. *Overview of Hybrid Domain Zero-Skipping*

Fig. 1 shows the overview of the proposed hybrid domain zero-skipping technique. The system begins by analyzing the time-domain waveform to identify and mask low-energy samples, which are deemed less likely to impact feature extraction. This process eliminates 39.5% of the data (Sparsity 1: Eliminate low-energy frames). When all samples are zero, indicating silence or low-importance frames, the rest of the computation including the frequency-domain and neural network stages is skipped (Sparsity 2: feature-domain sparsity). A Fourier Transform is applied to the remaining data. Afterwards, the frequency domain analysis is performed. Here, elements with amplitudes below a set threshold are zeroed out, further

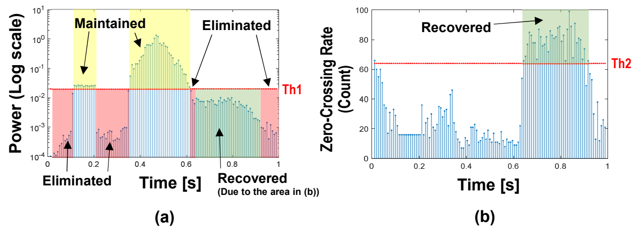
Fig. 3. (a) Threshold value 1 (Th1) eliminates low amplitude values. (b) Threshold value 2 (Th2) detects important high frequency signals.

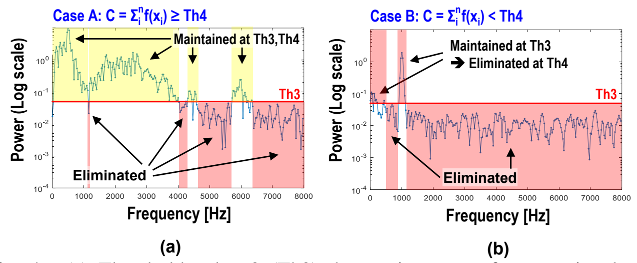
Fig. 4. (a) Threshold value 3 (Th3) detects important formant signals. (b) Threshold value (Th4) eliminates non-critical signals.

reducing the data by 67.4% (Sparsity 3-1/2: Element-/vector-wise thresholding). Additionally, the Mel filter bank which has a 98.5% sparsity is applied to the non-zero elements (Sparsity 3-3: Extremely sparse Mel filter). Finally, the refined features are forwarded to the neural network for classification.

B. *Thanos Architecture*

1) Time-Domain Processing Unit: Fig. 2a describes the Time-Domain Processing Unit (TDPU) within the Thanos architecture, which performs a critical role in identifying and skipping non-essential data early in the processing pipeline.

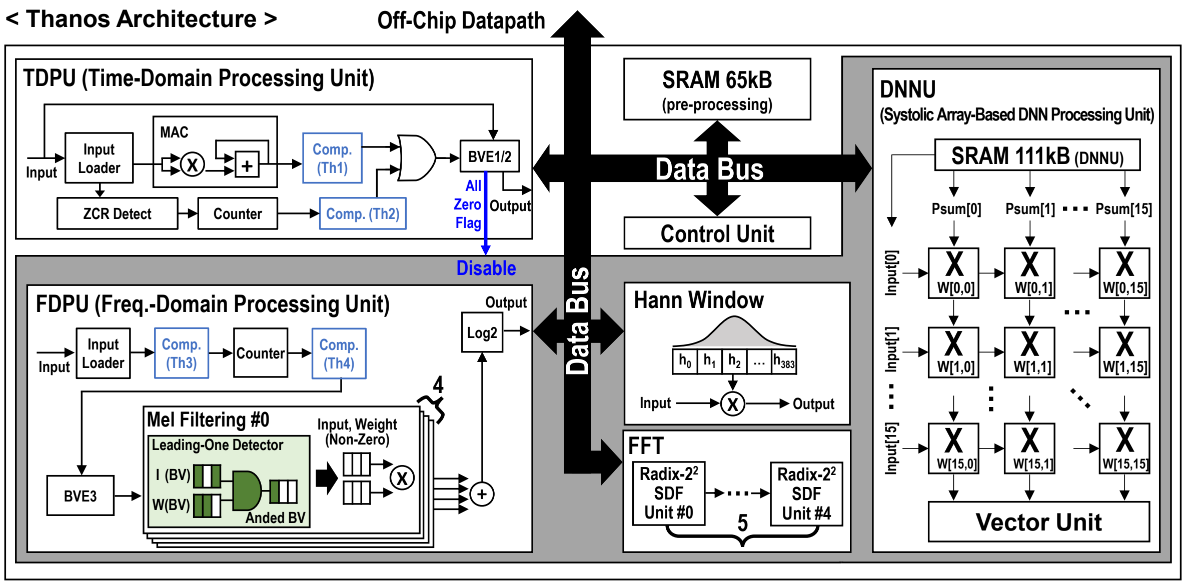

Fig. 5. Top-level implementation of the Thanos accelerator.

The TDPU begins by analyzing the amplitude of samples across frames in the input waveform. It employs a two-threshold mechanism to implement time-domain zero-skipping.

The first threshold value (Th1) is applied using short-time energy (STE, Eq. 1), which calculates the sum of squared amplitudes within a frame. If the computed energy is below Th1 indicating low amplitude, the corresponding frame is masked to zero as shown in Fig. 3a.

$$STE = \sum_{n=0}^{N-1} x^2[n] \tag{1}$$

However, to prevent the loss of important high-frequency components that might have low amplitude, a second threshold value (Th2) based on the zero-crossing rate (ZCR, Eq. 2) is employed. ZCR counts the number of sign changes (zero-crossings) across a frame, which helps identify and preserve high-frequency signals that STE might otherwise discard as depicted in Fig. 3b.

$$ZCR = \sum_{n=0}^{N-1} sgn(x[n-1]) \times sgn(x[n]) \tag{2}$$

When all the samples in an input feature are masked to zero, meaning both the STE and ZCR values have fallen below their respective thresholds, it indicates a silent or low-importance input with no effective voice information. In such cases, the TDPU generates a sleep signal which is propagated to all subsequent processing stages including the frequency domain and DNN units. This action effectively reduces computational latency and maximizes energy efficiency by bypassing unnecessary processing.

*2) **Frequency-Domain Processing Unit:*** After completing the Fourier transform, the frequency-domain elements are processed by the frequency-domain processing unit (FDPU) as shown in Fig. 2b-c. The FDPU is designed to exploit the characteristics of formants which are peaks in acoustic signals that carry the most critical information due to their resonance across multiple frequency bands, as shaped by the human vocal tract [6]. Elements with magnitudes lower than these formants are considered less significant.

The FDPU applies threshold value 3 (Th3) to each element, masking those below Th3 to zero to filter out non-essential frequency-domain data (Fig. 4a).

Next, the FDPU counts the non-zero elements in each window vector, denoted as C in Eq. 3. This count is then compared to a fourth threshold value (Th4). If C is below Th4, indicating that the vector lacks sufficient critical formant values, all elements in the vector are zeroed out (Fig. 4b). If C exceeds Th4, only elements below Th3 are masked preserving the important frequency-domain information for further processing.

$$C = \sum_{i=0}^{n-1} f(x_i), \quad f(x_i) = \begin{cases} 0 & (x_i = 0) \\ 1 & (x_i \neq 0) \end{cases} \tag{3}$$

This dual-threshold approach in the FDPU ensures that only the most significant frequency components are retained. Thus, optimizing the processing pipeline by eliminating redundant data and focusing computational resources on key features.

*3) **Top-Level Implementation:*** Fig. 5 illustrates the top-level architecture of the Thanos accelerator, comprising several key processing units: the TDPU, Hann computation core, FFT core, FDPU, and DNNU. The process begins with the TDPU, which processes the input voice features eliminating low-amplitude values. If all samples in a feature are zero, the remaining blocks are disabled via clock-gating to conserve energy. Next, the frequency domain stage is initiated using Hann windowing and FFT computation. The FDPU filters out less significant signals, isolating critical formant data. This data is then processed through a Mel filter bank. The extracted features are finally passed to the DNNU, which consists of a 16×16 array of MAC units designed with a systolic array architecture for efficient AI computations. The off-chip DRAM communication occurs in the following cases only: 1) pre-loading parameters for use by the pre-/post-processor, 2)

TABLE I
HYPERPARAMETERS FOR TRAINING.

		CNN Based model [1], [7], [8]	Transformer Based model [9]
Trained Epochs		200	
Batch Size		128	
Optimizer		Stochastic Gradient Descent	AdamW
Weight Decay		0.001	0.1
Learning Rate	**Scheduler**	Cosine annealing w/ warm up scheme	
	Warm up Policy	First 5 ephs (0 to 0.1)	First 10 ephs (0 to 0.001)

storing input data points, and 3) transferring the final output data of the neural network.

IV. RESULTS

A. *Experimental Setup*

In this section, we evaluate the Thanos hardware architecture by comparing it to a baseline design. For a fair comparison, both architectures were synthesized using 28nm CMOS technology at a target clock frequency of 303 MHz, which is the gate-level synthesizable maximum frequency of the Thanos architecture. Performance was analyzed with a cycle-accurate simulator, and timing and power metrics were extracted using Synopsys PrimeTime/PrimePower/PrimeSim. We conducted a quantitative analysis on the impact of hybrid-domain zero-skipping on latency, where five pre-processing stages and a backend DNN are selectively computed based on thresholding results and the average latency is calculated. As discussed in Section III-B3, the architecture uses FP16 for pre-processing and INT8 for the systolic array. These parameters are orthogonal to the key concept, and other configurations can be applied.

The evaluation utilized four well-known neural network benchmarks at KWS: TC-ResNet8 [1], DS-CNN (TinyML) [8], BC-ResNet-1 [7], BC-ResNet-1.5 [7], and KWT-1 [9], which were trained with post-training quantization (PTQ). The hyperparameters were set as detailed in Table I. For voice input data, we employed the Google Speech Commands Dataset (GSCD) [10] ver.1, with 12 distinct classes.

B. *Comparison with Previous Works*

We compare our Thanos architecture with the previous keyword spotting hardware (Table II). Among them, AAD-KWS [2] and Seol et al. [3] support the zero-skipping scheme. Different from the previous two architectures where feature-domain skipping is only supported, our approach can perform the zero-skipping in feature, time, frequency domains. Seol et al. needs to modify the neural network for the computation, so only dedicated networks can be executed. Kosuge [5] implemented a shift register-based convolutional engine which stores all the weights in the flip-flop array, and it does not use any SRAM cells. It can only compute a customized small-sized CNN model only, so it is impossible to compare with the Kosuge design. In addition, it does not support the computational skipping. Hence, to directly evaluate the effectiveness of our hybrid zero-skipping approach, we compare our design with 1) the baseline which executes the keyword spotting without zero-skipping and 2) the AAD-KWS which skips feature-domain computation.

TABLE II
COMPARISON WITH PREVIOUS KEYWORD SPOTTING HARDWARE

	AAD-KWS [2]	Seol et al. [3]	Kosuge *et al.* [5]	Ours
Technology	28 nm	28 nm	40 nm	28 nm
Supported Networks	DS-CNN[1]	Customized RNN Model Only	Customized CNN Model Only	Various DNNs
Dataset	GSCD	GSCD	GSCD	GSCD
Number of classes	2	7	12	12
Accuracy	97.8%[2]	92.8%	88.0%	95.4-98.3%
Zero-skipping (Skipped Operation)	Feature-domain (DNN)	Feature-domain (FFT+Mel+DNN)	Not Supported	Feature-domain (FFT+Mel+DNN), Time-domain (FFT+Mel), Frequency-domain (Mel)

[1]The paper demonstrated DS-CNN but may be compatible with other DNN models.
[2]Accuracy on 2-class GSCD

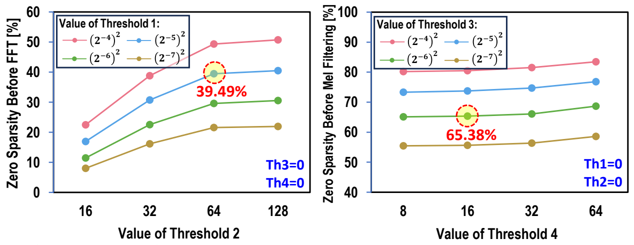

Fig. 6. Portion of zero values when sweeping Threshold 1 and 2 (left), Threshold 3 and 4 (right) values. The red circles indicate selected optimal conditions for our rest experiments.

C. *Results*

1) Select the Threshold values: To determine the optimal threshold values for zero-skipping, we performed extensive experiments across both time-domain and frequency-domain thresholds. We evaluated 16 combinations of Th1 and Th2 in the time-domain and Th3 and Th4 in the frequency-domain. As seen in Fig. 6, increasing threshold values generally raised sparsity. This reduced computational complexity. However, Figs. 7 and 8 highlight the trade-off between sparsity and accuracy. Our analysis revealed that Th1 should not exceed $(2^{-5})^2$, with Th2 optimally set at 64. For the frequency-domain, Th3 was set at $(2^{-6})^2$ and Th4 at 16. Exceeding these threshold values led to significant accuracy degradation, surpassing the acceptable accuracy limit of 1.0% compared to the baseline.

2) Accuracy and sparsity with thresholding: We analyzed the accuracy across 16 thresholding configurations, where each condition was either true (T) or false (F) for the thresholding parameters (Th1-4). As shown in Table III, applying thresholding to all stages (TTTT) results in overall accuracy that remains comparable, with differences ranging from -0.37% to -1.05% relative to the baseline. Table IV indicates that when applying thresholding to all stages, the time-domain achieves 39.52% sparsity, and the frequency-domain reaches 67.39% sparsity. This level of sparsity is consistent across all neural networks, as these operations occur in the non-AI computational stages. Feature-domain skipping was not considered here, but during silent inputs, all frames are removed, leading to 100% sparsity in both domains.

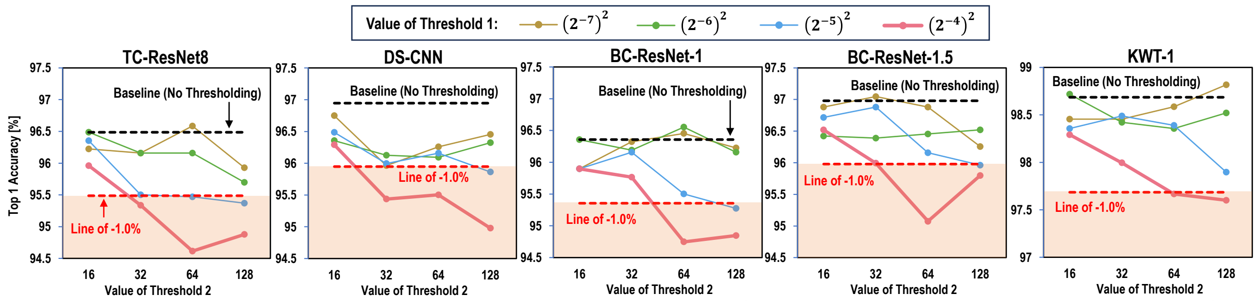

Fig. 7. Top-1 test dataset accuracy when sweeping Threshold 1 and 2 values. The legend refers to the Threshold 1 value, and a 'line of -1.0%' refers to a value that is 1.0 percentage points below the baseline accuracy. The region filled with peach color results in the unacceptable accuracy drop.

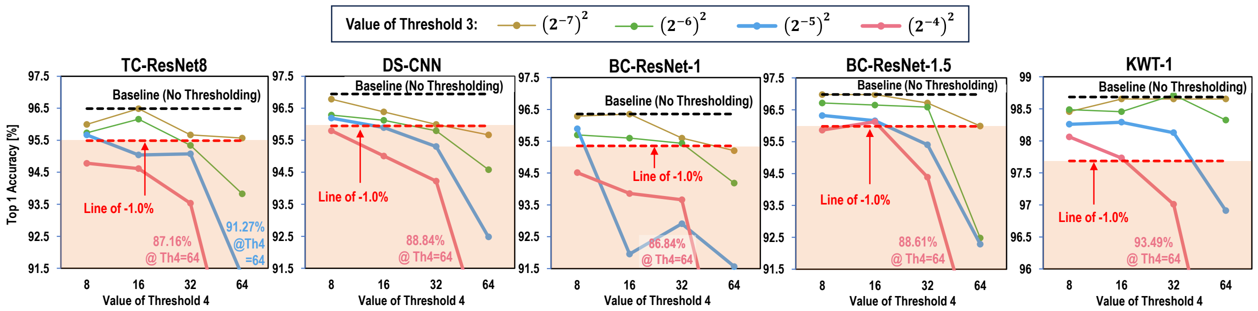

Fig. 8. Top 1 test dataset accuracy when sweeping across Threshold 3 and 4 values. The legend refers to the Threshold 3 value, and a 'line of -1.0%' refers to a value that is 1.0 percentage points below the baseline accuracy, respectively. The region filled with peach color results in the unacceptable accuracy drop.

TABLE III
TOP-1 TEST DATASET ACCURACY FOR 4 BENCHMARKS. TEST CONDITION: {TH1,TH2,TH3,TH4} = $\{(2^{-5})^2, (2^6), (2^{-6})^2, (2^4)\}$

Top-1 Accuracy [%]	{Th1,Th2,Th3,Th4} / "T": Using the Threshold, "F": Not Using the Threshold																a−b
	FFFF[a]	FFFT	FFTF	FFTT	FTFF	FTFT	FTTF	FTTT	TFFF	TFFT	TFTF	TFTT	TTFF	TTFT	TTTF	TTTT[b]	
TC-ResNet8 [1]	**96.49**	96.65	96.29	96.49	96.72	96.75	96.55	96.26	96.62	96.42	96.22	96.03	95.73	95.86	95.34	**95.44**	-1.05
DS-CNN [8]	**96.95**	96.55	96.42	96.55	96.65	96.75	96.62	96.42	96.85	96.75	96.55	96.22	96.09	96.16	96.13	**95.96**	-0.99
BC-ResNet-1 [7]	**96.36**	96.72	96.45	95.93	96.72	96.52	95.99	96.13	96.45	96.72	96.32	96.26	95.99	95.80	95.44	**95.44**	-0.92
BC-ResNet-1.5 [7]	**96.98**	96.55	96.42	96.55	96.82	96.75	96.62	96.42	97.11	96.75	96.55	96.22	96.55	96.16	96.13	**95.96**	-1.02
KWT-1 [9]	**98.69**	98.69	98.39	98.69	98.78	98.78	98.62	98.88	98.69	98.82	98.62	98.65	98.19	98.42	98.32	**98.32**	-0.37

TABLE IV
PORTION OF ZERO VALUES FOR TIME AND FREQUENCY COMPUTING DOMAINS. TEST CONDITION: GSCD (TEST DATASET), THRESHOLDING FOR ALL STAGES (TTTT) IN TABLE III.

Computation Domain	Time-Domain	Frequency-Domain
Portion of Zeros	39.52%	67.39%

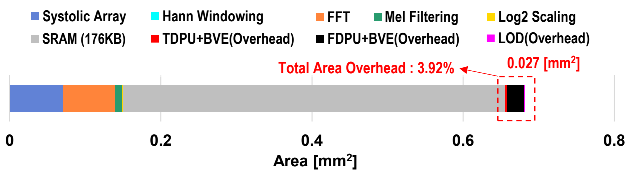

Fig. 9. Area breakdown of the Thanos accelerator.

*3) **Area**:* The area breakdown of the Thanos hardware, as shown in Fig. 9, indicates that the area overhead resulting from the implementation of the zero-skipping scheme (primarily due to the thresholding circuits and bit-vector encoding logic) is negligible. The total area overhead is only 3.92%, with a small footprint of 0.027 mm^2. The feature extraction parameters require 65KB of SRAM, and the maximum weight capacity for the DNN benchmarks is 111KB, leading to the selection of a 176KB SRAM capacity in total. While the current design efficiently uses the available area, additional SRAM could be integrated if more capacity is needed, either through tiling or by increasing on-chip data reuse.

*4) **Latency**:* The zero-skipping rates in the Thanos architecture fluctuate depending on input voice signal patterns, impacting latency. As shown in Fig. 10a, latency was comprehensively analyzed by considering the best (baseline: not applicable, AAD-KWS [2]: skipping in DNN computations, Thanos: feature-domain skipping for silent inputs), worst (baseline: not applicable, AAD-KWS: no skipping in DNN computations, Thanos: only Mel filter sparsity applied), and average scenarios across various benchmarks. As a result, DS-CNN [8] and BC-ResNets [7], despite being dominated by depthwise convolution with limited MAC utilization, benefit from significant latency reductions in the Thanos architecture. AAD-KWS, which does not skip non-AI operations, partially limits its effectiveness compared to the Thanos architecture.

*5) **Energy Consumption**:* The Thanos architecture achieves significant reductions in energy consumption, as seen in Fig. 10b, with similar trends to latency improvements. Although DS-CNN [8] and BC-ResNets [7] consume more energy than TC-ResNet [1] due to their larger filter counts and inefficient MAC utilization in depthwise convolution layers, our approach still achieves considerable energy savings. Notably, the energy consumed by the non-AI logic component is higher than other parts, which aligns with prior research showing that non-AI operations (e.g., feature extraction) dominate computations in KWS applications. Additionally, our SRAM capacity is sufficient to store all KWS parameters, eliminating the need for off-chip access and further reducing energy consumption.

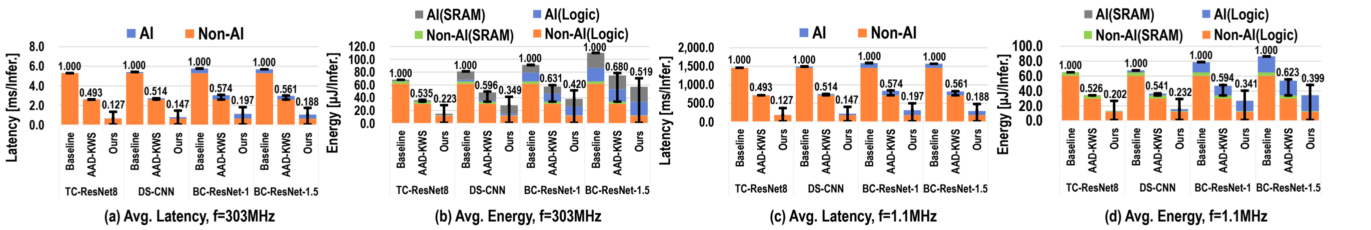
Fig. 10. Latency (a, c) and energy (b, d) consumption of the Thanos architecture compared to baseline and AAD-KWS [2] for various neural networks. The numbers above the bars indicate the normalized values for average-case scenario relative to the baseline cases. The black I-shaped markers represent the range of values across all scenarios (bottom: best-case scenario, top: worst-case scenario).

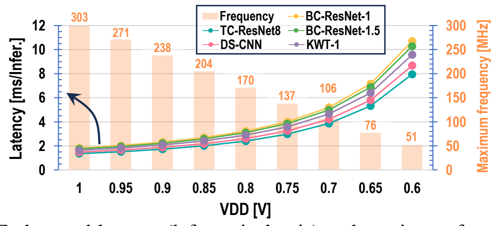
Fig. 11. End-to-end latency (left vertical axis) and maximum frequency (right vertical axis) are swept across various supply voltages (VDD) for different neural networks. Latency values are measured at the maximum frequency for each VDD.

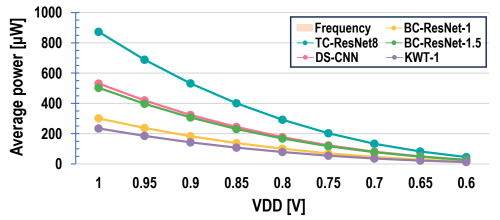
Fig. 12. Average power sweeping for each supply voltages (VDD) and various neural networks. The average power is measured at new target frequency (1.1MHz) under the constraint.

*6) **Analysis on Low supply voltage and Low-frequency mode**:* The 28nm design kit we used for the synthesis has the nominal voltage of 1.0V. At this voltage, the end-to-end latency for the KWS task is at most 1.8 ms (Fig. 11), which is significantly lower than the median inference time required for on-device recognition systems [11].

We further explored the optimal operating conditions at lower supply voltage to minimize power and energy consumption while maintaining the 500 ms latency constraint. To this end, we investigated the variations in latency and power consumption under the low supply voltage and low frequency conditions. The latency analysis revealed that the minimum operating frequency that satisfies the latency constraint is 1.1 MHz, which is achievable across all supply voltages shown in Fig. 11. In the power consumption analysis, reducing the operating frequency led to a decrease in dynamic power consumption, which consequently reduced the total power consumption (Fig. 12). Several supply voltages, particularly those below 0.8V, met the criteria for achieving a one-year battery life with an LR6 battery (3000 mWh) [12]. These analyses demonstrate that the Thanos architecture can operate efficiently, even in highly power-constrained environments.

For energy consumption, the same trend as power consumption was observed (Figs. 10c and d, Fig. 13b).

*7) **Evaluation on Transformer architecture**:* We extended our hybrid-domain zero-skipping approach to the KWT-1 Transformer model [9], which has parameter requirements ranging from 100k to 1M. (Fig. 13a and b) we employed 1024

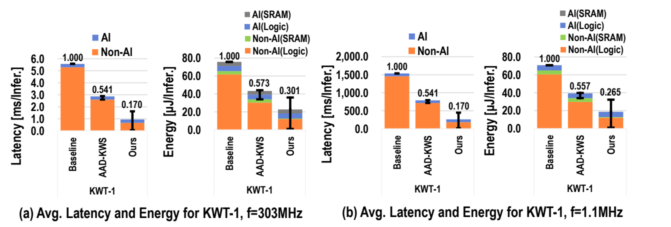
Fig. 13. (a) Latency and (b) energy consumption of the Thanos architecture for KWT-1. The numbers above bars indicate the normalized values to the baseline cases.

processing elements (PEs), 755.5 KB of SRAM, and a total area of 2.453 mm^2, with all 690.5KB of weights stored on-chip, avoiding off-chip access. Time, frequency, and feature-domain skipping were applied. KWT-1 also demonstrated significant latency and energy reductions under average cases, similar to the results shown in Sections IV-C4 and IV-C5.

V. Conclusion

We proposed a low-latency and energy-efficient specialized keyword spotting processor called Thanos in this paper. Considering that voice signals sometimes consist of less important samples for a given input feature, we observed the implicit/-explicit sparsites at hybrid time-feature-frequency domain in the keyword spotting algorithm. Based on the characteristics, we designed Thanos hardware architecture which consists of domain-wise processing units to maximize the performance by exploiting the sparsity in each domain. The Thanos design is synthesized in a 28nm CMOS technology, and the experimental results show that our hybrid-domain zero-skipping scheme reduces the latency by 80.3-87.3% and energy consumption by 48.1-79.8% on average, respectively.

Acknowledgment

This work was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2022-0-00266, Development of Ultra-Low Power Low-Bit Precision Mixed-mode SRAM PIM, 40%), National Research Foundation (NRF) funded by the Korean government (MSIT) (No. RS-2024-00439520, 40%), MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(IITP-2025-RS-2023-00260091, 10%) supervised by the IITP(Institute for Information & Communications Technology Planning & Evaluation), and Korea Institute for Advancement of Technology(KIAT) grant funded by the Korea Government(MOTIE) (P0017011, HRD Program for Industrial Innovation, 10%). The EDA tool was supported by the IC Design Education Center(IDEC), Korea.

References

[1] S. Choi, S. Seo, B. Shin, H. Byun, M. Kersner, B. Kim, D. Kim, and S. Ha, "Temporal Convolution for Real-Time Keyword Spotting on Mobile Devices," in Proc. Interspeech 2019, 2019, pp. 3372–3376.

[2] W. Shan, J. Qian, L. Zhu, J. Yang, C. Huang, and H. Cai, "Aad-kws: A sub-μ w keyword spotting chip with an acoustic activity detector embedded in mfcc and a tunable detection window in 28-nm cmos," IEEE Journal of Solid-State Circuits, vol. 58, no. 3, pp. 867–876, 2022.

[3] J.-H. SeoI, H. Yang, R. Rothe, Z. Fan, Q. Zhang, H.-S. Kim, D. Blaauw, and D. Sylvester, "A 1.5-μw end-to-end keyword spotting soc with content-adaptive frame sub-sampling and fast-settling analog frontend," in 2023 IEEE International Solid-State Circuits Conference (ISSCC). IEEE, 2023, pp. 1–3.

[4] Y. Dong, T. Jia, K. Du, Y. Jing, Q. Wang, P. Zhan, Y. Zhang, F. Yan, Y. Ma, Y. Liang et al., "A model-specific end-to-end design methodology for resource-constrained tinyml hardware," in 2023 60th ACM/IEEE Design Automation Conference (DAC). IEEE, 2023, pp. 1–6.

[5] A. Kosuge, R. Sumikawa, Y.-C. Hsu, K. Shiba, M. Hamada, and T. Kuroda, "A 183.4nj/inference 152.8w single-chip fully synthesizable wired-logic dnn processor for always-on 35 voice commands recognition application," in 2023 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits), 2023, pp. 1–2.

[6] J. C. Catford, A Practical Introduction to Phonetics. Oxford University Press, 1988.

[7] B. Kim, S. Chang, J. Lee, and D. Sung, "Broadcasted residual learning for efficient keyword spotting," Interspeech 2021, 2021.

[8] Y. Zhang, N. Suda, L. Lai, and V. Chandra, "Hello edge: Keyword spotting on microcontrollers," 2018.

[9] A. Berg, M. O'Connor, and M. T. Cruz, "Keyword Transformer: A Self-Attention Model for Keyword Spotting," in Proc. Interspeech 2021, 2021, pp. 4249–4253.

[10] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," 2018.

[11] S.-Y. Chang, B. Li, D. Rybach, Y. He, W. Li, T. N. Sainath, and T. Strohman, "Low latency speech recognition using end-to-end prefetching," in Interspeech 2020, 2020, pp. 1962–1966.

[12] P. Jokic, E. Azarkhish, R. Cattenoz, E. Türetken, L. Benini, and S. Emery, "A sub-mw dual-engine ml inference system-on-chip for complete end-to-end face-analysis at the edge," in 2021 Symposium on VLSI Circuits, 2021, pp. 1–2.