A Fully-Integrated Energy-Scalable Transformer Accelerator Supporting Adaptive Model Configuration and Word Elimination for Language Understanding on Edge Devices

Zexi Ji* *MIT* Cambridge, MA, USA

Hanrui Wang* *MIT* Cambridge, MA, USA

Miaorong Wang *MIT* Cambridge, MA, USA

Win-San Khwa *TSMC Corporate Research* Hsinchu, Taiwan

Meng-Fan Chang *TSMC Corporate Research* Hsinchu, Taiwan

Song Han *MIT* Cambridge, MA, USA

Anantha P. Chandrakasan *MIT* Cambridge, MA, USA

*Equally contributing authors

2023 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED) | 979-8-3503-1175-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/ISLPED58423.2023.10244459

***Abstract*—Efficient natural language processing on the edge is needed to interpret voice commands, which have become a standard way to interact with devices around us. Due to the tight power and compute constraints of edge devices, it is important to adapt the computation to the hardware conditions. We present a Transformer accelerator with a variable-depth adder tree to support different model dimensions, a SuperTransformer model from which SubTransformers of various sizes can be sampled enabling adaptive model configuration, and a dedicated word elimination unit to prune redundant tokens. We achieve up to 6.9× scalability in network latency and energy between the largest and smallest SubTransformers, under the same operating conditions. Word elimination can reduce network energy by 16%, with a 14.5% drop in F1 score. At 0.68V and 80MHz, processing a 32-length input with our custom 2-layer Transformer model for intent detection and slot filling takes 0.61ms and 1.6μJ.**

***Index Terms*—hardware accelerators, machine learning, natural language processing, transformers**

I. Introduction

There are close to 15 billion connected devices in the internet of things (IoT) today and many of them require efficient natural language processing (NLP) algorithms to interpret voice commands. Attention-based Transformer models have replaced recurrent neural networks (RNNs) as the predominant model for NLP applications due to parallel input processing and the attention mechanism being able to capture both short and long-range relations, resulting in faster training and increased accuracy on a variety of NLP tasks. However, existing mainstream models (e.g. BERT, GPT) are way too large for edge devices, with the largest model (Switch Transformer) containing over a trillion parameters (Fig. 1). For simple NLP tasks on the edge (e.g. smart watches, home assistants), tiny custom Transformer models can achieve good accuracy, while being much more suitable for constrained hardware [1].

979-8-3503-1175-4/23/$31.00 ©2023 IEEE

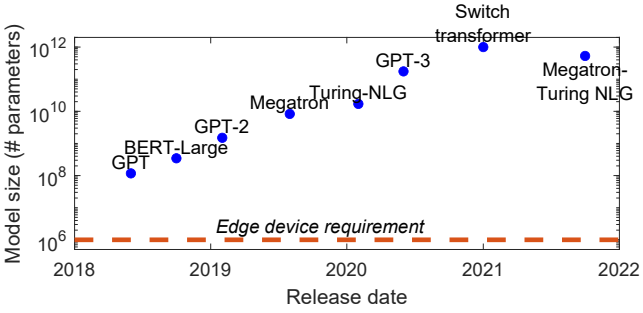


Fig. 1. Growth in model size of recent popular language models.

There are two main challenges when deploying lightweight NLP models on edge devices (Fig. 2). Firstly, hardware constraints can fluctuate based on battery level, latency requirements, availability of compute resources, and accuracy tolerance. Effectively adapting to these varying conditions typically requires multiple models of different sizes. For instance, when the device is less constrained (i.e. high battery level, relaxed latency demands etc.), we may use a large model. On the other hand, under more constrained conditions, we may opt for a small model to reduce latency and energy consumption. But storing multiple models incurs a significant memory overhead, potentially exceeding the on-chip memory capacity and requiring us to load different models from external memory. This additional data transfer would diminish or even negate the energy and latency benefits from using multiple models. Secondly, sentences usually contain redundant words that contribute little to the overall understanding and may potentially be skipped during the majority of the processing. Conventional models spend an equal amount of time processing each word, leading to unnecessary computation [2].

Our paper addresses these challenges with an energy-

scalable Transformer accelerator targeting small IoT devices with three key features: 1) a variable-depth MAC adder tree to support different model dimensions; 2) adaptive model configuration using a custom SuperTransformer model to generate models of various sizes, while only taking up the memory footprint of a single full model; and 3) a comparator-based word elimination unit to progressively remove unimportant words from the sentence, reducing computation.

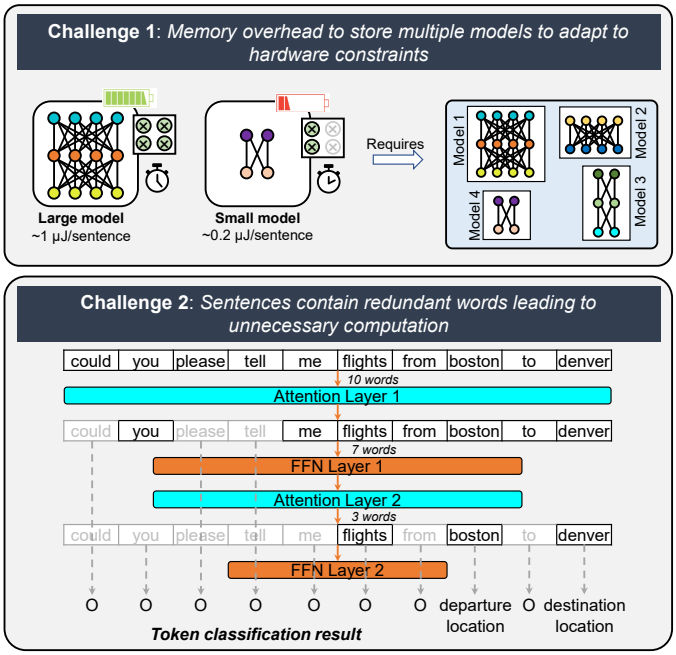


Fig. 2. Challenges of deploying lightweight models on edge devices.

II. Background

A. Transformer models

Transformer models are used to process sequence data (e.g. sentences) with a basic structure shown in Fig. 3. Inputs are first passed through an embedding layer, where each input is mapped to an embedding vector of length d_{emb}. A positional encoding vector is then added to each embedding vector to represent the location of each input. This is distinct from RNNs where the positional information is inherent in the sequential processing, whereas Transformers process all inputs in parallel.

The position-encoded inputs are passed through a series of encoder layers. Each layer consists of a multi-head attention mechanism followed by a feed-forward network (FFN, i.e. fully-connected layer). In the attention mechanism, each input is split along the embedding dimension into several heads and projected to a query, key, and value vector of dimension d_{QKV}/N_{head}. A similarity score (typically dot product) is computed between every pair of query and key vectors and normalized by softmax to generate the attention probability matrix, which gives a measure of the relevance of each pair of inputs. This matrix is multiplied by the value matrix to obtain a new representation of the inputs. Results from the different heads are combined through a fully-connected layer. This is followed by two FFNs, first mapping to dimension d_{FFN}, then back to d_{emb} to act as input to the next layer. The final encoder layer can be connected to either a classification head or a language model head, depending on the nature of the task.

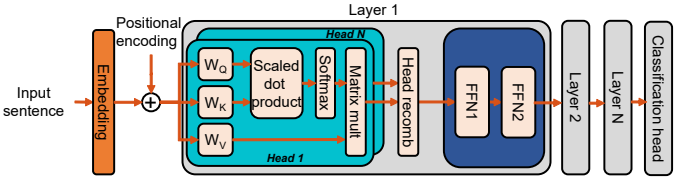


Fig. 3. Basic structure of transformer model.

B. Intent detection and slot filling task

We focus on the intent detection and slot filling task, which is representative of the complexity required for processing voice commands. This is a classification task in which the overall intent of a sentence is determined (i.e. the command) and relevant fields for the intent are identified by labeling each word with an appropriate slot. Fig. 4 shows an example with the desired intent being requesting flight information. The relevant words are "Boston" as the departure location and "Orlando" as the destination location. The remaining words are labeled as *O* for "outside." The intent is obtained by appending a *<CLS>* token (for "classification") to the beginning of the sentence, which is later passed through an intent head after the final encoder layer (while all the other words get processed by a slot head).

The ATIS (Airline Travel Information Systems) and SNIPS datasets are common benchmarks [3] [4]. ATIS consists of air travel related sentences with 26 different intent classes and 129 slots. SNIPS features more everyday tasks such as asking for the weather and playing music. It contains 10 different intents and 39 slots.

For evaluation on this task, we look at the intent accuracy and slot F1 score. The F1 score is the harmonic mean of the precision and recall, with the positive class being associated with non-*O* labels. This is preferred over token-wise slot accuracy due to the high proportion of *O* labels. Simply

	Intent	*Slots*									
Input sentence	<CLS>	i	need	a	flight	that	goes	from	boston	to	orlando
Ground Truth	flight info	O	O	O	O	O	O	O	departure location	O	destination location
Prediction	flight info	O	O	O	departure location	O	O	O	departure time	O	destination location
Predicted Non-O	-	✗	✗	✗	✓	✗	✗	✗	✓	✗	✓
Correctly predicted Non-O	-	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓

$$\text{Precision} = \frac{\#(\text{correctly predicted non-O tokens})}{\#(\text{predicted non-O tokens})} = \frac{1}{3}$$

$$\text{Recall} = \frac{\#(\text{correctly predicted non-O tokens})}{\#(\text{ground truth non-O tokens})} = \frac{1}{2}$$

$$\text{F1} = \frac{2(\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}} = \frac{2}{5}$$

$$\text{Accuracy} = \frac{\#(\text{correctly predicted tokens})}{\#(\text{all tokens})} = \frac{8}{10}$$

Fig. 4. Example of intent detection and slot filling task with calculation of evaluation metrics.

labeling all tokens as *O* would result in a high accuracy, making it a misleading measure of model performance.

C. Existing work

Early works targeting Transformers were mainly architectures focused on accelerating the attention mechanism [5]. More recently, several accelerators for Transformers have emerged, including both digital and compute-in-memory implementations. They take advantage of techniques such as approximate computing [6], sparsity [7] [8], and varying the amount of computation based on input complexity [9]. All these works focus on larger networks (e.g. BERT) and achieving high throughput, but may not be optimized for running smaller networks that are more appropriate for edge applications.

III. Design Features

Our design combines algorithmic and circuit innovations to create a low-power scalable solution for performing NLP inference on the edge. It can efficiently support different model configurations with a variable-depth adder tree and uses a SuperTransformer model from which submodels of varying sizes can be generated through weight sampling. This enables adapting the model based on changing hardware conditions without the overhead of storing multiple distinct models. A comparator-based word elimination unit with programmable threshold provides another way to trade off energy and model performance.

Fig. 5 shows the system architecture of the proposed design. A 16-bit external memory interface facilitates transfer of model weights, loading the input data, and outputting the final results. The entire model and inputs (up to 32 words) are stored on-chip (in the W/B/I-Mem) so external data accesses are not required during operation. The remaining memory is divided into groups dedicated to storing certain intermediate values. Each memory bank is connected to its own power

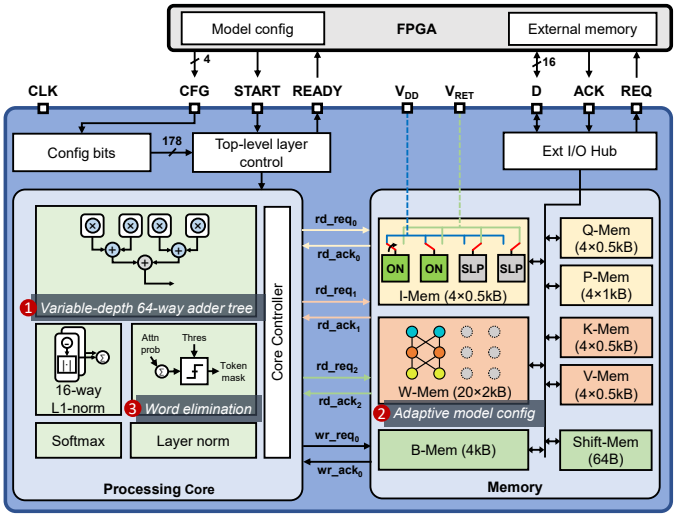


Fig. 5. System architecture of our proposed transformer accelerator.

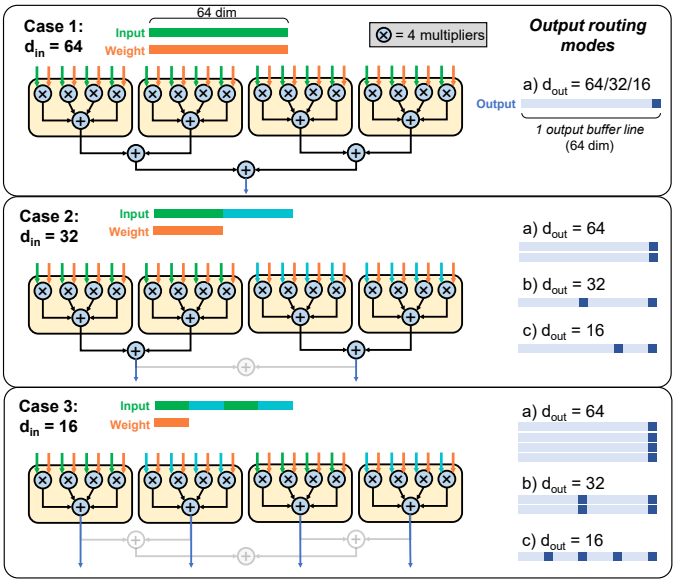


Fig. 6. Variable-depth adder tree and output routing modes to support different model dimensions.

switch, allowing it to be dynamically connected to a lower retention voltage when inactive to minimize leakage. Data is transferred between the memory and processing core through three read channels and one write channel. Each read channel has access to a unique set of memory groups to reduce the amount of multiplexing required. The processing core consists of a reconfigurable 64-way MAC for matrix multiplication, 16-way L1-norm for computing attention score, word elimination unit, softmax for generating attention probabilities, and layer normalization unit. The model configuration is supplied through a serial interface.

A. Variable Depth Adder Tree

We design a variable-depth 64-way adder tree with seven output routing modes to easily support different model dimensions in the hardware (Fig. 6). The input dimension (d_{in}) determines the configuration of the adder tree and weight read pattern, while the output dimension (d_{out}) determines the spacing between consecutive words during writeback. Each dimension can be one of {16, 32, 64}. We use an output stationary dataflow, computing each output in a single cycle. The outputs of the 64 multipliers are combined through a 3-stage adder tree: the first stage adds 16 products together, while the remaining stages add two values at a time. Outputs can be taken at any point in the adder tree to obtain the desired sum, which are then routed to the output buffer. In most cases, only one output buffer line needs to be activated, but up to four lines may be required depending on the configuration. Fig. 7 shows the latency breakdown for three models where all dimensions are equal. The flexible compute unit maintains the throughput across different configurations, as shown by the latency scaling with $d_{in} \times d_{out}$ for most layers. The compute units can also be configured as 64 independent MAC units for softmax, layer normalization, and transposed matrix multiplication.

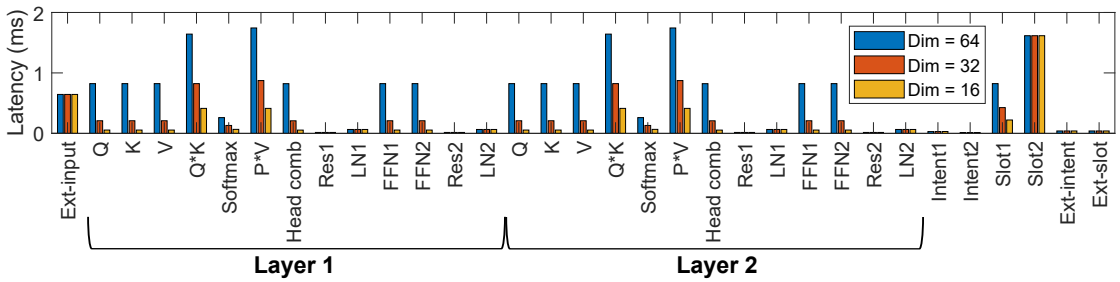


Fig. 7. Latency breakdown with $d_{emb} = d_{QKV} = d_{FFN}$.

B. SuperTransformer Model

The reconfigurable MAC unit is used in conjunction with a Super-Sub-Transformer framework, in which models of different sizes share parameters [10]. This enables energy scalability by varying the model size, but with the storage requirement of only a single full-sized model (Fig. 8). The SuperTransformer is the largest model in the design space and a SubTransformer is a submodel that shares all its parameters with the SuperTransformer. The SuperTransformer has an elastic layer number (n_L), embedding dimension (d_{emb}), QKV dimension (d_{QKV}), and feed-forward network hidden dimension (d_{FFN}).

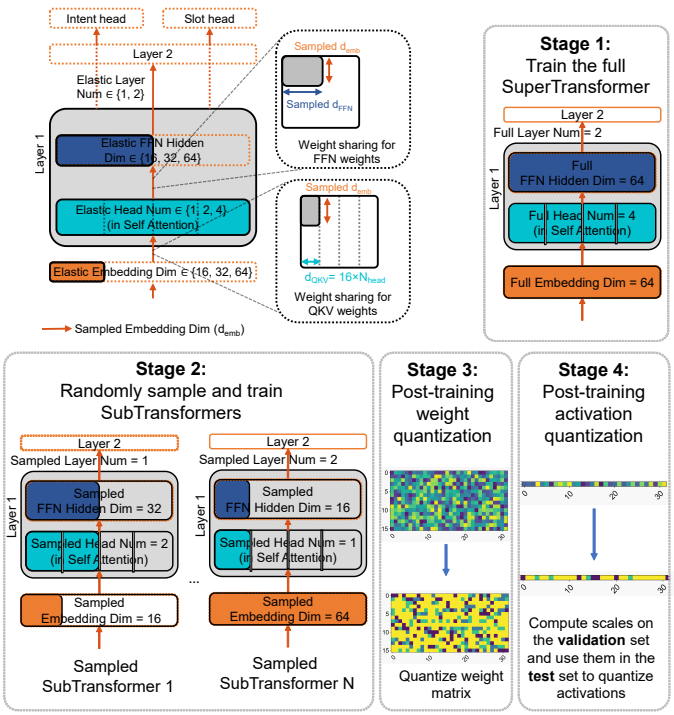


Fig. 8. SuperTransformer architecture and training pipeline.

To ensure each SubTransformer can perform the task properly, we uniformly sample a SubTransformer during each training step and update its parameters. After enough steps, all SubTransformers can be individually used. Our custom SuperTransformer model has 2 layers with d_{emb}, d_{QKV}, and d_{FFN} all equal to 64. For each SubTransformer, n_L can be 1 or 2, and each dimension can be chosen from 16, 32, 64. This gives us 54 (2×3^3) SubTransformers with the footprint of one full model. Practically, not all submodels may be used as several may be of similar size, with varying performance. Software evaluation can determine which submodel has the best accuracy for a given model size. Though we train a small model for our design, this framework can be applied to larger models as well, so long as the number of training cycles is large enough such that each submodel is sampled a sufficient number of times during training. We quantize weights to 4b and activations to 8b. L1-norm is used instead of dot-product attention to reduce multiplications, without accuracy loss.

The SuperTransformer model is stored entirely on-chip in the weight memory, which comprises the majority of the on-chip SRAM. Due to the output stationary dataflow, new weights are read every cycle. Thus, the weight read access energy is a significant fraction of the total energy consumed by the memory. In order to minimize this energy, the weight memory is divided into an array of 5×4 memory banks (5 rows, 4 columns). Only one row of memory banks is active at a time, and the number of columns that need to be active is determined by the sublayer d_{in}.

C. Word Elimination Unit

Besides adjusting model size, we can adjust how many words to process in each sentence. Motivated by the high redundancy in human language, we include word elimination to remove unimportant words. The elimination threshold is tunable, offering another way to trade off accuracy and energy. It is performed after computing attention probabilities in the softmax layer (Fig. 9). Each row of the attention matrix provides a normalized measure of the impact of a particular word to every other word in the sentence. The column-wise sum provides a proxy for the overall importance of a word. The larger the sum, the larger the overall impact. A threshold can be applied on the sum to generate a word mask indicating which words will be eliminated. Compared to other implementations which employ a top-k engine [2] [8], thresholding is more hardware-friendly, but provides less control over the elimination ratio. The word elimination unit consists of 32 parallel accumulators and comparators. During the computation of the attention output ($P*V$), multiplications

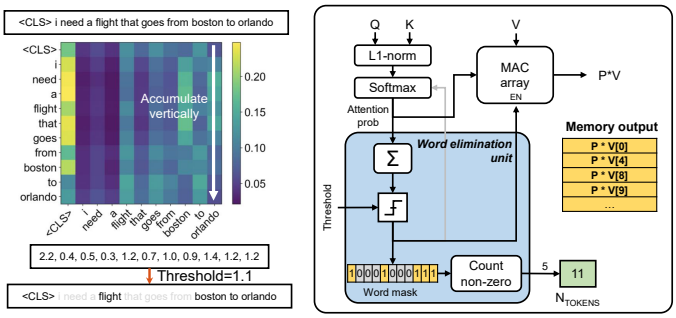


Fig. 9. Detailed operation of word elimination unit.

and writeback are skipped for words for which the word mask is 0. The resulting output memory has no gaps associated with removed words, allowing subsequent computations to be performed without regard to the word mask. Only the register storing the sentence length needs to be updated.

IV. Measurement Results

The design was taped out in TSMC 28nm HPC+ technology and occupies 0.9mm×0.9mm. The die micrograph and test setup are shown in Fig. 10. An FPGA was used to interface between the chip and the PC and the supply voltage and current were measured by Keithley sourcemeters, from which the power consumption was computed. The chip can operate up to 80MHz at 0.68V, consuming 2.62mW. Under these conditions, processing a 32-length input on the full model takes 0.61ms and 1.6μJ (or 20.5 inferences/s/MHz). For our target application, it was not necessary to have a latency much less than 1 ms. Because of this, we set a moderate target maximum frequency during the design phase to ease the effort during synthesis and place-and-route, rather than pushing to the maximum achievable frequency of the technology node. The peak efficiency point occurs at 0.56V and 20MHz, where the energy efficiency reaches 3.8 TOPS/W. A combination of

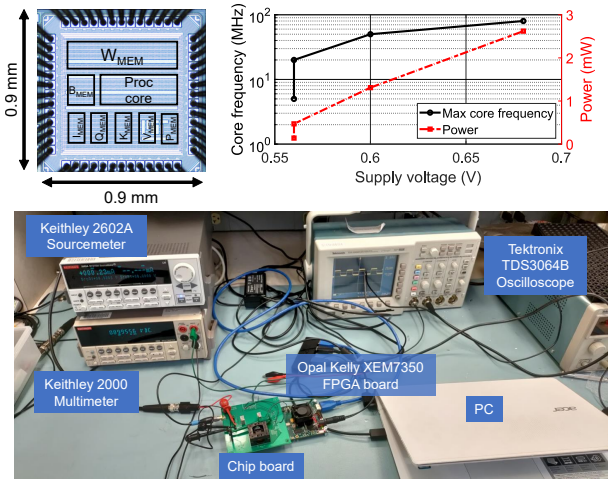
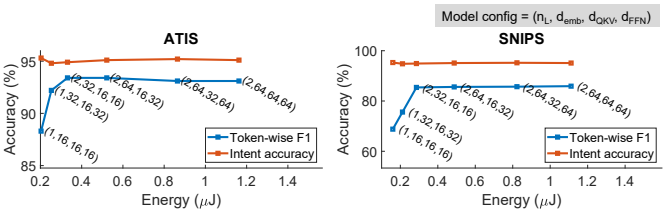


Fig. 10. Die micrograph, voltage-frequency scaling characteristics, and test setup.

SRAM sleep mode and power switches to connect inactive memory banks to a lower retention voltage reduces SRAM leakage by 11×, translating to a 56% reduction in the overall leakage.

Fig. 11 shows the tradeoff between model accuracy and network energy for the ATIS and SNIPS datasets. The accuracy is most sensitive to changes in n_L and d_{emb}, rather than d_{QKV} and d_{FFN}. In general, a two layer SubTransformer performs better than a single layer SubTransformer of similar size. The F1 score remains fairly high, even for submodels less than 1/4 the size of the full model (down to about $(n_L, d_{emb}, d_{QKV}, d_{FFN}) = (2, 32, 16, 16)$). This may be a result of some transfer learning occurring during the training process between the SuperTransformer and SubTransformers from the model sampling. Intent accuracy remains high across configurations. The difference in network energy between the largest and smallest SubTransformers is 5.8× and 6.9× for ATIS and SNIPS, respectively, demonstrating a wide range for scalability.



	n_L	d_{emb}	d_{QKV}	d_{FFN}	#MACs	E (μJ)	t (ms)	F1 (%)	Intent (%)
ATIS	2	64	64	64	2.22M	1.16	2.44	93.1	95.1
	2	32	16	16	0.58M	0.33	0.68	93.4	94.4
	1	32	16	32	0.48M	0.25	0.5	92.2	94.8
	1	16	16	16	0.37M	0.20	0.41	88.3	95.3
SNIPS	2	64	64	64	2.12M	1.11	2.36	85.9	96.8
	2	32	16	16	0.48M	0.29	0.60	85.4	97.5
	1	32	16	32	0.38M	0.21	0.42	75.7	97.1
	1	16	16	16	0.27M	0.16	0.33	68.8	97.4

32 tokens, 20MHz@0.56V

Fig. 11. Model accuracy for various configurations on the SNIPS and ATIS datasets.

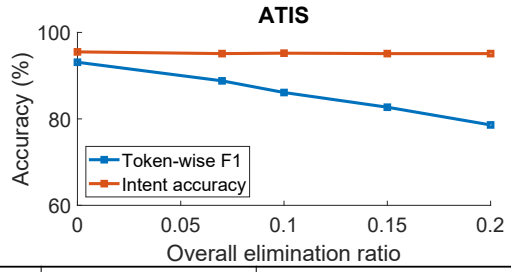
Fig. 12 shows the effect of the overall elimination ratio on the accuracy for the full model on the ATIS dataset. When the overall elimination ratio is increased to 0.2, the F1 score drops by 14.5%, while the energy and latency are reduced by 16%. The accuracy degradation is mainly due to the relatively short length of test inputs (average 16 words), providing fewer opportunities for word elimination. For tasks with longer inputs, less accuracy loss would be expected at higher elimination ratios, due to increased redundancy.

Compared to other transformer accelerators in Table I, ours is the only one that targets lightweight networks that can fit entirely on-chip. This allows it to achieve much lower power and smaller area, making it more suitable for edge devices with limited compute and energy resources. Our peak energy efficiency is generally lower because we do not take as much advantage of sparsity. The performance of BERT-base on a scaled up (12×) version of our design is estimated based on

TABLE I
COMPARISON WITH OTHER WORKS.

	This work	**[6]**	**[7]**	**[8]**	**[9]**
Target application	Transformer	Transformer	Transformer	Multimodal Transformer	Transformer
Implementation	Digital	Digital	Digital CIM	Digital CIM	Digital
Technology (nm)	28	28	28	28	12
Max frequency (MHz)	80	510	240	275	717
SRAM (kB)	56	336	192	128 (CIM)/192 (buffer)	647
Area (mm^2)	0.81	6.82	6.83	6.83	4.6
Power (mW)	0.14–2.6	12–272	27–118	30–153	9–122
Energy efficiency (TOPS/W)	2.6–3.8	1.91–27.56	3.1–20.5	12.1–101.1	3.0–18.1
Word elimination support	Yes	No	No	Yes	No
Model	Custom (ATIS)	GPT-2	BERT (base)	ViLBERT	BERT (base)
Model on-chip	Yes	No	No	No	No
Network latency (ms)	0.61–9.76 (247[1])	-	1684	268–925	682
Network energy (μJ/token)	0.036–0.05 (43.8[1])	-	15.6	2.24–7.72	4000

[1]Estimated performance of BERT-base on a scaled up (12×) version of our design operating at 0.56V and 20 MHz



Threshold L1	Threshold L2	Elim. ratio L1	Elim. ratio L2	Elim. ratio Overall	E (μJ)	t (ms)	F1 (%)	Intent (%)
0	0	0	0	0	1.16	2.44	93.1	95.1
0.6	2.6	0.04	0.09	0.07	1.10	2.31	88.8	95.1
0.65	2.8	0.06	0.14	0.1	1.04	2.17	86.1	95.2
0.7	3.0	0.09	0.20	0.15	1.01	2.10	82.7	95.1
0.75	3.1	0.14	0.26	0.2	0.98	2.05	78.6	95.5

ATIS dataset, 32 tokens, 20MHz@0.56V

Fig. 12. Effect of word elimination on the full model accuracy with the ATIS dataset.

the number of operations. Latency and energy are close to other designs. However, our scaled design would have more on-chip memory than most of the other works, reducing the amount of required external memory accesses, which is not captured in the reported numbers.

V. CONCLUSION

We demonstrate an energy-scalable Transformer accelerator for edge devices. It supports 5.8× (for ATIS) to 6.9× (for SNIPS) scalability in the network latency and energy with a small memory footprint by adopting a custom-trained SuperTransformer model and having reconfigurable hardware to support various model dimensions. The word elimination unit can reduce energy by up to 16%, with a 14.5% drop in F1 score.

ACKNOWLEDGMENT

The authors would like to thank TSMC for funding and the TSMC University Shuttle Program for tapeout support.

REFERENCES

[1] D. Wu, L. Ding, F. Lu, and J. Xie, "SlotRefine: A Fast Non-Autoregressive Model for Joint Intent Detection and Slot Filling," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (Online), pp. 1932–1937, Association for Computational Linguistics, Nov. 2020.

[2] H. Wang, Z. Zhang, and S. Han, "SpAtten: Efficient Sparse Attention Architecture with Cascade Token and Head Pruning," in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, (Los Alamitos, CA, USA), pp. 97–110, IEEE Computer Society, mar 2021.

[3] C. T. Hemphill, J. J. Godfrey, and G. R. Doddington, "The ATIS Spoken Language Systems Pilot Corpus," in *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania*, 1990.

[4] A. Coucke, A. Saade, A. Ball, T. Bluche, A. Caulier, D. Leroy, C. Doumouro, T. Gisselbrecht, F. Caltagirone, T. Lavril, M. Primet, and J. Dureau, "Snips Voice Platform: an embedded Spoken Language Understanding system for private-by-design voice interfaces," *CoRR*, vol. abs/1805.10190, 2018.

[5] T. J. Ham, S. J. Jung, S. Kim, Y. H. Oh, Y. Park, Y. Song, J.-H. Park, S. Lee, K. Park, J. W. Lee, and D.-K. Jeong, "A^3: Accelerating Attention Mechanisms in Neural Networks with Approximation," in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 328–341, 2020.

[6] Y. Wang, Y. Qin, D. Deng, J. Wei, Y. Zhou, Y. Fan, T. Chen, H. Sun, L. Liu, S. Wei, and S. Yin, "A 28nm 27.5TOPS/W Approximate-Computing-Based Transformer Processor with Asymptotic Sparsity Speculating and Out-of-Order Computing," in *International Solid- State Circuits Conference (ISSCC)*, vol. 65, pp. 1–3, 2022.

[7] F. Tu, Z. Wu, Y. Wang, L. Liang, L. Liu, Y. Ding, L. Liu, S. Wei, Y. Xie, and S. Yin, "A 28nm 15.59μJ/Token Full-Digital Bitline-Transpose CIM-Based Sparse Transformer Accelerator with Pipeline/Parallel Reconfigurable Modes," *International Solid- State Circuits Conference (ISSCC)*, vol. 65, pp. 466–468, 2022.

[8] F. Tu, Z. Wu, Y. Wang, W. Wu, L. Liu, Y. Hu, S. Wei, and S. Yin, "MulTCIM: A 28nm 2.24μJ/Token Attention-Token-Bit Hybrid Sparse Digital CIM-Based Accelerator for Multimodal Transformers," pp. 248–250, 2023.

[9] T. Tambe, J. Zhang, C. Hooper, T. Jia, P. N. Whatmough, J. Zuckerman, M. C. D. Santos, E. J. Loscalzo, D. Giri, K. Shepard, L. Carloni, A. Rush, D. Brooks, and G.-Y. Wei, "A 12nm 18.1TFLOPs/W Sparse Transformer Processor with Entropy-Based Early Exit, Mixed-Precision Predication and Fine-Grained Power Management," *International Solid-State Circuits Conference (ISSCC)*, pp. 342–344, 2023.

[10] H. Wang, Z. Wu, Z. Liu, H. Cai, L. Zhu, C. Gan, and S. Han, "HAT: Hardware-Aware Transformers for Efficient Natural Language Processing," in *Annual Conference of the Association for Computational Linguistics*, 2020.