© 2026 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

This is the author's accepted version of a paper published in: M. Wabnitz et al., "ConfASR: A Conformer Block Accelerator for Speech Recognition Optimized for Edge Devices," 2026 31st Asia and South Pacific Design Automation Conference (ASP-DAC), Lantau, Hong Kong, 2026, pp. 147-153, doi: 10.1109/ASP-DAC66049.2026.11420567

ConfASR: A Conformer Block Accelerator for Speech Recognition Optimized for Edge Devices

Malte Wabnitz, Max Nilovic, Finn Scholz, Dominik Friedrich, Christian Lanius, Jie Lou, and Tobias Gemmeke
Chair of Integrated Digital Systems and Circuit Design, *RWTH Aachen University*
Aachen, Germany
wabnitz@ids.rwth-aachen.de

***Abstract*—Attention-based neural networks, like transformers, have significantly improved automatic speech recognition (ASR). Adding convolution operations to transformers results in conformers, which enable better learning of local dependencies and reduce word error rate. We introduce ConfASR, the first conformer block accelerator designed for efficient ASR inference on edge devices. Our system is optimized both in terms of algorithms and hardware to support all transformer operations and additional features required for conformers, including depthwise-separable convolution and learned positional encoding. We propose a hardware-friendly normalization, shared scaling factors for non-linear functions, and an efficient dataflow with a shared MAC array that keeps all activations on chip.**
Implemented in a $22\,\mathrm{nm}$ FDSOI technology, ConfASR operates at $250\,\mathrm{MHz}$ with a power consumption of $359\,\mathrm{mW}$, and a die area of $1.19\,\mathrm{mm}^2$. It performs over 900 times faster than necessary for real-time streaming requirements. This makes the architecture suitable not only for ASR but also for other transformer-based applications. ConfASR reduces latency by over $4\times$ and power consumption by $16\times$ during real-time use compared to previous solutions, while supporting more functionality.

***Index Terms*—Automatic Speech Recognition, Transformer, Attention, Convolution, ASIC, Edge Devices**

I. Introduction

The transformer architecture [1] has revolutionized performance across a range of natural language processing (NLP) tasks, including automatic speech recognition (ASR) [2]–[4], machine translation (MT) [5], and large language models (LLMs) [6], [7]. Despite their success, transformers come with substantial computational and energy demands during inference. Since inference is performed repeatedly in real-world deployments, its cumulative energy consumption can eventually surpass that of training. For instance, ChatGPT's estimated energy usage in January 2023 exceeded $1500\,\mathrm{MW\,h}$, surpassing even the energy consumed during its training [8]. This growing energy footprint raises environmental concerns and highlights the urgency of developing more efficient hardware solutions [9], [10].

Transformers are typically deployed on GPU-based cloud computing platforms. However, this approach introduces additional latency, increases overall energy consumption due to data movement and communication overhead, and raises privacy and security concerns. While GPUs offer flexibility, they are significantly less energy-efficient than specialized ASICs [11]. This motivates the development of optimized hardware accelerators to enable on-device execution, particularly for energy-constrained devices.

Previous works have developed hardware accelerators for transformer networks, highlighting challenges like the computational demand of the attention mechanism [12] and the complexity of non-linear functions such as normalization and softmax [13], [14]. However, most implementations target isolated components rather than full transformer models. Other FPGA-based solutions [15], [16] offer flexibility but lack the energy efficiency and performance inherent in ASICs. Additionally, some approaches forego hardware sharing, leading to increased area [14]. In ASR, the conformer - an extension of the transformer - outperforms other end-to-end models [2], yet optimized hardware support is missing.

To address this gap, we propose an algorithm-hardware co-optimized ASIC accelerator for the conformer block. The main contributions of this work are:

- We introduce ConfASR, the first accelerator capable of executing all conformer block operations. It implements an efficient dataflow with a single Multiply-Accumulate (MAC) array that supports fully connected layers, point- and depthwise convolution, multi-head self-attention, and positional encoding.
- We apply a hardware-friendly mean absolute (MA) normalization that uses only $50\,\%$ of the input tensor, avoiding the costly squared root computation of layer normalization.
- We propose the use of shared scaling factors over all conformer blocks for nonlinear functions like softmax, and gated linear unit (GLU) which allows for the reuse of look-up tables (LUTs).
- We implemented and evaluated the complete architecture in a $22\,\mathrm{nm}$ FDSOI technology resulting in a power of $359\,\mathrm{mW}$ at $250\,\mathrm{MHz}$, and an area of $1.19\,\mathrm{mm}^2$. The latency is reduced by $4\times$ and for real-time use, the power consumption is $16\times$ lower compared to previous works.

This work was supported in part by the German Federal Ministry of Education and Research under Grant 03ZU2106CA (NeuroSys), and in part by the German Federal Ministry for the Environment, Nature Conservation, Nuclear Safety and Consumer Protection under Grant 67KI32006A (RESCALE).

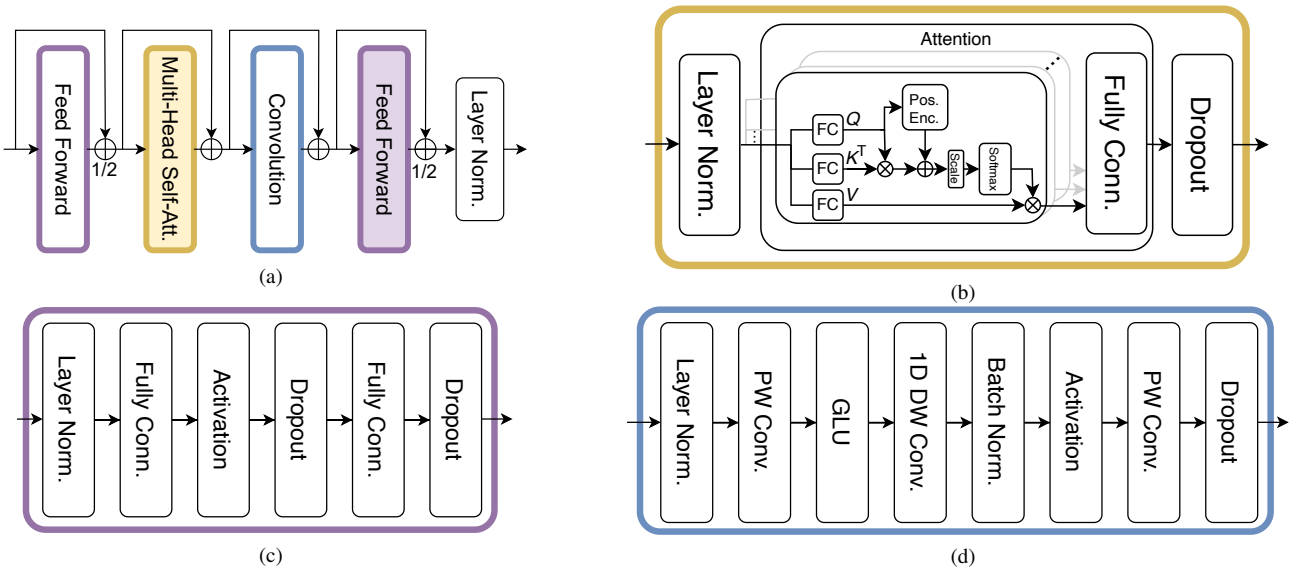

Fig. 1: Conformer block components: (a) Network architecture for a complete conformer block, using only the filled blocks builds the standard transformer; (b) Multi-head self-attention module; (c) Feed-forward module; (d) Convolution module.

II. Background

This work is based on the conformer network as introduced by Gulati et al. [2]. We focus on the encoder, consisting of N consecutive conformer blocks. The network architecture of one conformer block is shown in Fig. 1a. It comprises all operations of a standard transformer block (filled in Fig. 1a [1]), and further incorporates an additional feed-forward (FF) module before the multi-head self-attention (MHSA) operation, and a convolution (CONV) module between the MHSA and the final FF module. All these modules employ residual connections, and the block ends with a final layer normalization. Fig. 1b to 1d depict these sub-modules in detail. Each of these modules begins with a layer normalization and ends with a dropout operation. In the next sections, we denote the feature dimension as d and the sequence length as s.

A. Multi-Head Self-Attention Module

The MHSA module with its attention operation is depicted in Fig. 1b. After the layer normalization, the input is split into h heads of dimension $d_\text{h} = d/h$. In each head, these inputs are processed by three separate fully connected layers to produce query Q, key K, and value V. Multiplying QK^T yields an $s \times s$ matrix. Unlike to the standard transformer, the conformer incorporates a learned positional encoding [17] inside the MHSA module instead of sinusoidal encoding at the encoder and decoder input. Positional encoding involves multiplying Q^T with a pre-computed matrix of size s^3. The result is added to QK^T. Finally, the values are scaled and then passed to the softmax to compute the final attention score. These scores serve as weights for multiplication with the values V. The outputs of all heads are concatenated and forwarded to a final fully connected layer.

B. Feed Forward Module

The FF module consists of two fully connected layers, as shown in Fig. 1c. The first layer transforms input features from dimension d to the intermediate feed-forward size d_ff, while the second layer converts them back to size d.

C. Convolution Module

The CONV module, depicted in Fig. 1d, utilizes depthwise separable convolutions. The initial pointwise convolution maps the inputs to dimension $2d$. A GLU [18] splits the input into two matrices G_1, G_2 along the feature dimension. The output is computed as the element-wise multiplication of these matrices $GLU(M) = G_1 \bigodot \sigma(G_2)$, where $\sigma(x)$ denotes the sigmoid function. This feeds into the depthwise separable operation comprising a 1D depthwise convolution, batch normalization, activation function, and finally another pointwise convolution.

III. Algorithm Exploration

In this section, we investigate the impact of various algorithmic modifications on the network accuracy to identify an optimal configuration for the hardware implementation. Our algorithmic exploration utilizes an open-source toolkit featuring an ASR conformer implementation [19]. We started with a standard floating-point network as our baseline.

TABLE I: Algorithmic exploration for hardware-friendly conformer configurations.

	d_ff	Norm	p (%)	WER (dev-clean)
Baseline	2048	LN (w/ bias)	100	2.58 %
C1-FP	512	LN (w/ bias)	100	2.95 %
C2-FP	2048	MA (w/o bias)	100	2.65 %
C3-FP	512	MA (w/o bias)	50	2.65 %
C3-INT8	512	MA (w/o bias)	50	3.32 %

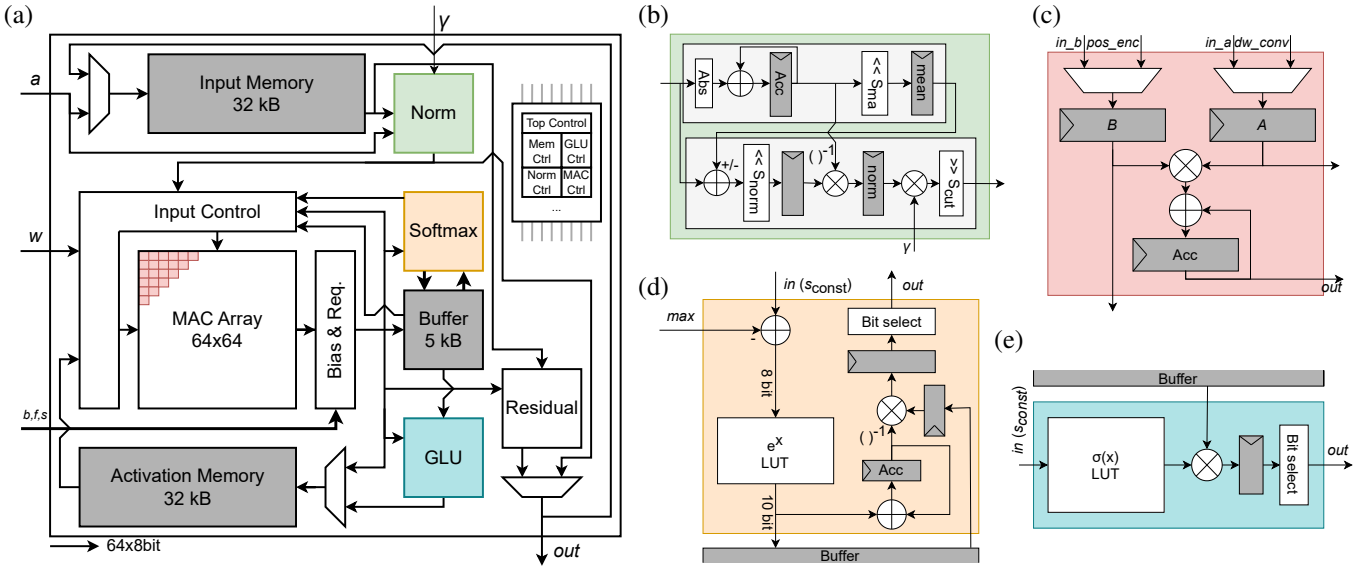

Fig. 2: Hardware Architectures: (a) System-level architecture of the design with all modules; (b) Normalization module; (c) Single processing element of the MAC array; (d) Softmax module; (e) Gated linear unit (GLU) module.

For the input feature extraction, we compute Mel coefficients with a frame stride of $10\,\mathrm{ms}$, followed by $6\times$ convolutional sub-sampling. At the output, we use an autoregressive decoder. The encoder comprises $N = 12$ conformer blocks and ReLU as an activation. The exploration is based on the LibriSpeech dataset [20]. With this baseline setup, we achieve a word error rate (WER) of $2.58\,\%$, as detailed in Table I.

Common values for the feature dimension d are 144, 256, or 512, with $d_{\mathrm{ff}} = 4d$ [2]. For our baseline implementation with $d = 512$, this leads to a large dimension of $d_{\mathrm{ff}} = 2048$. To reduce memory requirements, we trained the network with $d_{\mathrm{ff}} = 512$. As shown in Table I, reducing d_{ff} from 2048 to 512 increases WER slightly to $2.95\,\%$ (Conformer C1-FP).

Computing square roots for layer normalization in integer-quantized networks presents significant challenges for efficient hardware implementations [14], [15]. We replaced the layer normalization in the conformer with a MA normalization. As a further optimization, we utilize only the first $p = 50\,\%$ of input tensors for the mean computation and omit the bias in the final linear mapping. As shown in Table I, conformer architecture C2-FP substitutes the default normalization with a floating-point MA norm without bias, resulting in a slight WER increase to $2.65\,\%$ compared to the baseline. Adjusting d_{ff} to 512 and using only half of the inputs for mean computation maintains the WER, as shown with architecture C3-FP.

Further exploration involved post-training quantization for an integer-only design [21]. Matrix-matrix multiplications were performed using INT8 format, biases were quantized to align with the scaling factors. Retaining nonlinear functions in floating-point form resulted in a WER of $3.32\,\%$ (C3-INT8). This C3-INT8 network is utilized as the baseline for subsequent hardware implementations.

IV. Hardware Implementation

The system-level architecture is shown in Fig. 2a. It can either execute individual modules, transformer, or conformer block operations. We use three register files depicted in gray to store activations, the input and activation memory, and a buffer. For the hardware implementation, we use a sequence length of $s = 64$. Aligning s with $s = d_{\mathrm{h}}$ facilitates hardware-friendly tiling and maximizes MAC array utilization. Considering the feature extraction, this supports an input length of approximately $3.8\,\mathrm{s}$, consistent with streaming ASR methods using chunked attention [22].

a) Memories: The input and activation memories consist of s register files with 1R/1W macros, each containing d memory words. Two distinct read/write access patterns are employed. The first pattern is shown in Fig. 3 where each shade indicates the simultaneous access in one cycle. Each row corresponds to a dedicated register file, allowing parallel reading and writing of s words. The second pattern is similar but incorporates tiling into h sections along dimension d.

b) Normalization: The integer MA module is depicted in Fig. 2b. It comprises the MA computation (top) followed by the normalization (bottom) involving a division and the multiplication by γ. Normalization cancels out pre-existing

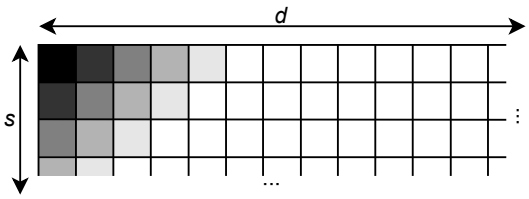

Fig. 3: Memory access scheme.

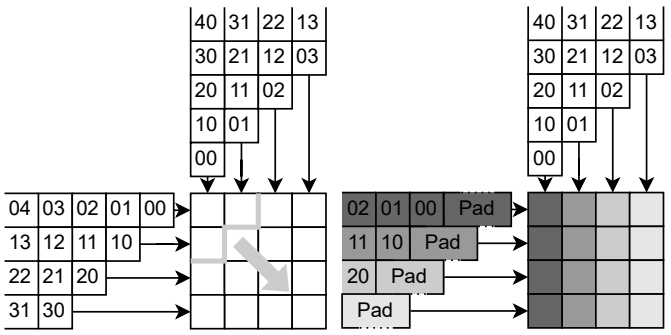

Fig. 4: Default output-stationary dataflow for matrix-matrix multiplications (left), flow for 1D depthwise convolution (right), the numbers indicate indices in the 2D arrays.

quantization scaling factors. However, intermediate results are requantized using scaling factors that are integer powers of 2 resulting in shift operations as shown in Fig. 2b. Extensive exploration atop the C3-INT8 network (Table I) determined the shift values S as $S_{\text{ma}} = 2$, $S_{\text{norm}} = 7$, and $S_{\text{cut}} = 8$, achieving a WER of $4.18\,\%$ for dev-clean.

c) MAC Array: The MAC array comprises 64×64 processing elements (PEs). Each PE has several inputs as depicted in Fig. 2c. An output stationary design is used, with the output data stored in the accumulator (Acc). This requires the design to load every weight only once. Additionally, the output stationary flow allows us to exploit a transposing characteristic when feeding back the output of the MAC array to its input at the top (see Section V). The gray modules are clock-gated registers. Each of the two input operators for the multiplication, A and B, have two possible inputs, depending on the operation mode. For the linear layers that require a standard matrix-matrix multiplication, input activations are applied to in_a, weights to in_b. The alternative inputs are dw_conv used to support depthwise convolution and pos_enc for the positional encoding. For the standard matrix multiplication, data is applied starting from the top-left PE, propagating through the array to the bottom right corner as shown in Fig. 4 (left). After 64 cycles, 64 INT8 words of activations and weights each are applied in every cycle. A and B are forwarded each cycle to the right and the bottom PE. For the depthwise convolution, the input activations of each row i from the left are broadcasted to all PEs of column i (Fig. 4, right). The weights are propagated by one PE each cycle.
For the positional encoding, each row of Q^{T} is multiplied with its individual $s \times s$ matrix. For this, the rows of the MAC array are enabled consecutively while the weights are directly being applied to this active row.

d) Softmax: Implementing a hardware-friendly solution for the softmax function, including its exponential component, was crucial. Typically, each of the 12 conformer blocks in the network has a unique scaling factor associated with the inputs to the softmax, presenting challenges for integer-based implementations tailored to each block. Post-training network exploration demonstrated that employing a shared scaling factor (q_{const}) across all conformer blocks is feasible, facilitating the use of a single shared integer LUT. Moreover, previous methods involved identifying and subtracting the maximum input value to confine exponential outputs within $(0, 1]$ [13], [15], necessitating double reading of activations. The adoption of a constant scaling factor offers an additional advantage by allowing the use of a fixed maximum value.
Exploring various configurations resulted in a WER of $3.48\,\%$ for C3-INT8 (Table I), with $max = 32$. The LUT is designed to output $10\,\text{bit}$ words. A shared buffer stores these values during accumulation and retrieves them for final division. Outputs are rounded, truncated, and clipped, effectively producing a fixed-point representation (Bit selection). The resulting hardware module is depicted in Fig. 2d.

e) GLU: The GLU function is implemented similarly to the softmax, utilizing an $8\,\text{bit}$ LUT with a shared scaling factor across all blocks. The architecture is depicted in Fig. 2e. Given that the input to the GLU is $s \times 2d$, managing access patterns is essential to minimize memory usage. Thus, tiling is applied to output matrices as 64×64 blocks in an alternating fashion. We re-use the buffer to store tile t. Then, the corresponding second tile $t + h$ is computed for the sigmoid function. These values are combined with previously stored ones and then forwarded to be stored in tile t within the activation memory. This approach achieves a WER of $3.75\,\%$ for C3-INT8.

f) Bias & Requantization: The requantization module is inspired by SwiftTron [14], featuring multiplication followed by bit selection. We enhanced this module by incorporating an initial addition to accommodate bias and rounding to the nearest value during requantization. Additional linear operations, such as batch normalization in the convolution module [23], can also be merged into this module.

V. Dataflow

The architecture, along with its control logic, provides high versatility. A hierarchical structure of finite-state machines is controlled by instruction words at the input, allowing flexibility in processing order and enabling operations to be skipped. This adaptability allows the accelerator to support various networks beyond the specified conformer. In Fig. 5, the timing diagram for an entire conformer block operation is presented with $s = 64$, $d = d_{\text{ff}} = 512$, and $h = 8$. The design can be equivalently scaled to other dimensions by adapting the MAC array and memory sizes as all the key operations stay the same. The design ensures that the weights for fully connected, convolution and attention operations only need to be accessed once per inference.

a) Feed Forward Module: The first fully connected layer multiplies input $A \in \mathbb{R}^{s \times d}$ by weight matrix $W \in \mathbb{R}^{d \times d_{\text{ff}}}$. Here, W is divided into h tiles $W_i \in \mathbb{R}^{d \times d_h}, i \in \{1, ..., h\}$, producing output $O_i \in \mathbb{R}^{s \times d_h}$. After requantization and activation, these tiles are stored in the activation memory. The second fully connected layer is computed similarly but passes its output through a residual connection back to the input memory.

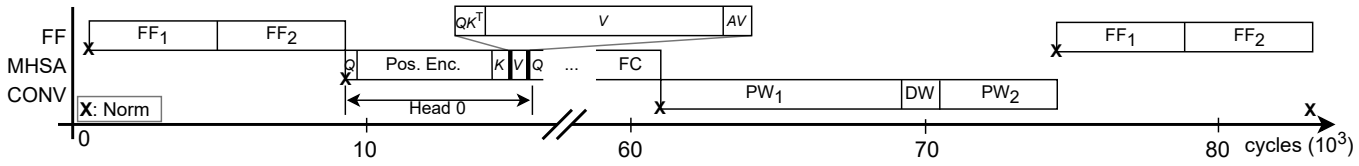

Fig. 5: Timing diagram for the computation of an entire conformer block operation on ConfASR.

b) Multi-Head Self-Attention Module: In the MHSA module, heads are processed sequentially. Fully connected layers for Q, K, and V follow similar procedures as above, however, tiling is not necessary as the split into heads achieves equivalent effects on the dimensions. For each head i, the computed matrix Q_i is stored in the buffer. Positional encoding computation follows Section IV, storing results in the activation memory's i-th tile. Upon completing computation of K_i, we exploit the transposing characteristic of the output-stationary array when feeding back its output directly to its top input. K_i is fed back in such a way while Q_i is read from the buffer and applied to the left input of the array, computing $Q_i K_i^T$. Outputs are combined with the previously calculated positional encodings using the adder in the requantization module. The results are forwarded to the softmax module while the MAC array already computes V_i^{T} using the normalization outputs from the top input and transposed weight matrices from left. The softmax outputs - the attention scores A_i - are delayed until V_i is computed and then multiplied with V_i, yielding $A_i V_i$. Outputs are stored again in the activation memory's i-th tile. This process is repeated across all heads. The outputs of the heads are concatenated and forwarded to a final fully-connected layer. After the residual connection, results are stored back in the input memory.

c) Convolution Module: The dataflow for the pointwise convolutions is similar to that of fully connected layers, except for the first pointwise convolution's output dimension $\mathbb{R}^{s\times 2d}$. This output is directed to the GLU and requires a different execution order for the tiles to match the previously described GLU computation (Section IV). Once outputs are stored in the activation memory, depthwise convolution follows, concluding with the second pointwise convolution.

d) MA Computation: The MA computations indicated by **X** in Fig. 5 refer to the complete execution of the MA operation. After this initial computation, the mean is stored in the clock-gated register and the outputs are recomputed whenever required for the subsequent tiling steps. Data is read from input memory and combined with the mean during this process.

VI. Results

The proposed architecture was designed in System Verilog and implemented using $22\,\mathrm{nm}$ FDSOI technology. Synthesis was performed with Genus, while place-and-route (P&R) was executed using Innovus. Power consumption analysis was conducted with Voltus. The layout can be seen in Fig. 6. The design was simulated after P&R with $250\,\mathrm{MHz}$ at $0.65\,\mathrm{V}$ with data from an ASR reference network.

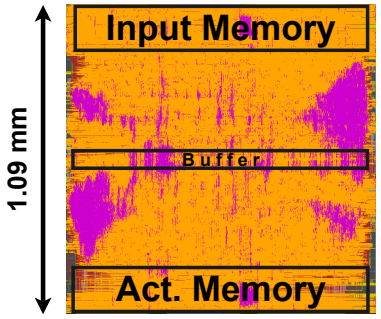

Fig. 6: Layout of ConfASR with input memory (top), buffer (center), and activation memory (bottom).

The area breakdown (Fig. 7, left) reveals that the design is dominated by the MAC array ($51.9\,\%$) and memory units ($35.2\,\%$). The power (Fig. 7, right) shows a similar trend with the MAC array contributing with $78\,\%$ and the memories with $13.3\,\%$ to the total power.

There have been various works on accelerators for attention-based networks, mostly transformers without the conformer extensions. Therefore, we first show a high level comparison of existing accelerators for attention-based networks in Tab. II. While designs that support normalization, feed forward, attention, and softmax operations provide the key functionalities for standard transformers, the conformer also requires positional encoding inside each block, depthwise convolution, and the GLU function. ConfASR is the only accelerator to support all the required operations while keeping power and normalized area in a similar range due to the optimized modules and a high degree of hardware sharing. We introduced hardware sharing at several stages, including the use of a single MAC array,

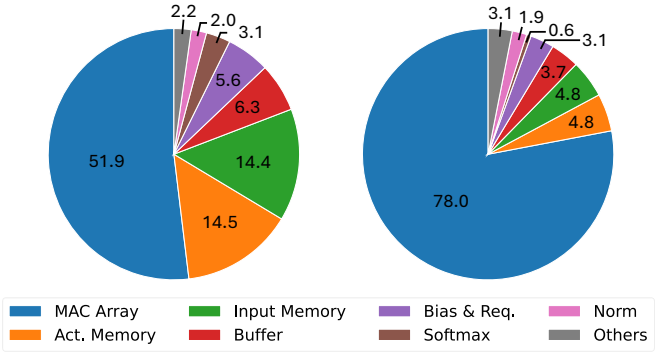

Fig. 7: Area (left), power (right) breakdown in percent.

TABLE II: Overview of accelerators for attention-based networks.

Design	**HPCA'20** [12]	**SOCC'20** [15]	**ISCA'21** [24]	**ISSCC'21** [25]	**ISSCC'22** [26]	**VLSI'22** [27]	**IJCNN'23** [14]	**ISLPED'23** [13]	**ISCAS'23** [16]	**This work**
Name	A^3	-	ELSA	FlexASR	-	-	SwiftTron	ITA	-	ConfASR
Implementation	Synthesis	FPGA	Layout	Tapeout	Tapeout	Tapeout	Synthesis	Layout	FPGA	Layout
Tech. (nm)	40	16	40	16	28	5	65	22	16	22
#PEs	2x64	64x64	528	4x16x16	4x8x16	16x64	Various	16x64	32x64	64x64
Embeddings ($s \times d$)	320×64 (per head)	64×512	512×64 (per head)	-	-	384×1024	256×768, 224×224	64×256	64×512	64×512
W/A Format	FXP4.4	INT8	FXP6.3	FP8	INT12	INT4	INT8	INT8	INT8	INT8
Act. functions	FXP4.4	FXP	FP16	FP8	INT12	FXP	INT32	INT8	INT8	INT8
Network	MemN2N, BERT	Transf.	BERT	LSTM + Attention	(GPT-2)	BERT-L.	RoBERTa, DeiT	I-BERT	Transf., ResNet	Conformer
Task	WikiMovies, SQuAD	BLEU	SQuAD, RACE	LibriSpeech	(LLM)	SQuAD	GLUE, ImageNet	Image Class.	ImageNet	LibriSpeech
Operations										
Normalization		✓		✓		✓	✓		✓	✓
Feed Forward		✓			✓	✓	✓	(✓)	(✓)	✓
Attention	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Pos. Encoding										✓
Softmax	(✓)	✓	(✓)	✓	✓	✓	✓	✓	✓	✓
Depthw. Conv.									✓ (Stan.)	✓
GLU										✓
Freq. [MHz]	1000	200	1000	573	510	1760	143	500	200	250
Area [mm^2]	2.08	-	2.15	8.84	6.82	0.153	273	0.407	-	1.19
Norm. Area	0.63	-	0.65	16.7	4.21	2.96	31.3	0.173	-	1.19
w/o memory	0.14	-	0.37	-	-	-	31.3	0.137	-	0.77
Voltage [V]	-	0.85	-	0.8	1.1	0.67	-	0.80	0.85	0.65
Power [mW]	110	16700	1470	214	273	50.6	33640	121	16900	359

sharing the buffer between MAC array, softmax, and GLU, and reusing the LUTs by exploiting shared scaling factors. Compared to a fully-flat design like SwiftTron, the normalized area with this hardware sharing is reduced by $26\times$.

For a better comparison, we further analyzed specific submodules. To verify the benefits of the optimized normalization, we implemented alternative integer realizations. The proposed MA approach reduces the latency by $25\,\%$ compared to an implementation with bias and $100\,\%$ for the mean. Compared to an also common implementation with a root mean square normalization with binary search for the square root, our approach reduces area by $30\,\%$ and energy by $37\,\%$.

Compared to the optimized softmax module in ITA [13], our implementation with the constant maximum and the shared memory reduces the area by $30\,\%$ when normalized to the same number of parallel computations.

Looking at the impact of these modules on the total design, we see that the optimized normalization and softmax occupy only $2\,\%$ and $3.1\,\%$ compared to contributing with $25\,\%$ and $17\,\%$ to SwiftTron [14]. This efficiency is also reflected in the power consumption (Fig. 7).

For an application-specific evaluation, we perform an ASR-only system-level comparison in Tab. III with results for real-time operation of ConfASR including all 12 conformer blocks. ConfASR with its conformer specific architecture, shows a competitive WER when combining all previously introduced techniques, while decreasing latency. Compared to FlexASR, latency can be reduced more than $4\times$, which leads to an execution that is more than $900\times$ faster than required for real-time streaming operation. Power during real-time operation is reduced about $16\times$ compared to the Apple Watch S7.

TABLE III: Comparison with Other ASR Platforms.

	FlexASR [25]	Apple Wat. S7 (Neural Eng.) [28]	Nvidia H100 [29]	CPU[a,b]	**ConfASR**[a]
Technology	16 nm	7 nm	4 nm	10 nm	22 nm
Dataset	LibriSp.	Custom	LibriSp.	LibriSp.	LibriSp.
WER (%)	9.4	4.76	-	4.95	4.95
Laten. (s)	0.018	0.19	0.0146	9.33	**0.00402**
Power (mW)	214	263	$7 \cdot 10^4$ - $7 \cdot 10^5$	2890	359
RT factor	0.56	0.19	0.0183	2.46	**0.00106**
RT power (mW)	120[c]	50[c]	1280 - 12800[c]	7100	**3.16**

[a] For 12 block encoder

[b] Reference implementation block- and headwise sequentially on a single core of a Intel Xeon Gold 6326 CPU @2.90GHz, power normalized by 64 cores

[c] Normalized with Real-Time (RT) factor

VII. Conclusion

In this work, we presented ConfASR, an advanced conformer accelerator capable of executing all operations within a conformer block. We proposed a comprehensive dataflow utilizing a single MAC array, featuring optimized normalization, nonlinear functions with shared scaling factors, and inherent transposing in the MAC array's feedback path. This architecture and dataflow are also applicable to other transformer networks. ConfASR was implemented and simulated using 22 nm FDSOI technology. Our optimizations achieved more than $4\times$ reduction in latency compared to other ASIC implementations while supporting more functionality. The real-time power consumption is reduced by $16\times$ compared to previous works.

REFERENCES

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit *et al.*, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

[2] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar *et al.*, "Conformer: Convolution-augmented Transformer for Speech Recognition," 2020. [Online]. Available: https://arxiv.org/abs/2005.08100

[3] N. Moritz, T. Hori, and J. Le, "Streaming Automatic Speech Recognition with the Transformer Model," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 6074–6078.

[4] S. Kim, A. Gholami, A. Shaw, N. Lee *et al.*, "Squeezeformer: An Efficient Transformer for Automatic Speech Recognition," in *Advances in Neural Information Processing Systems*, vol. 35. Curran Associates, Inc., 2022, pp. 9361–9373.

[5] Q. Wang, B. Li, T. Xiao, J. Zhu *et al.*, "Learning Deep Transformer Models for Machine Translation," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Jul. 2019, pp. 1810–1822.

[6] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018.

[7] H. Touvron, L. Martin, K. Stone, P. Albert *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[8] M. F. Argerich and M. Patiño-Martínez, "Measuring and improving the energy efficiency of large language models inference," *IEEE Access*, vol. 12, pp. 80 194–80 207, 2024.

[9] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, "Green AI," *Commun. ACM*, vol. 63, no. 12, p. 54–63, Nov. 2020.

[10] D. Patterson, J. Gonzalez, U. Hölzle, Q. Le, C. Liang *et al.*, "The Carbon Footprint of Machine Learning Training Will Plateau, Then Shrink," *Computer*, vol. 55, no. 7, pp. 18–28, 2022.

[11] L. Cheng, Y. Gu, Q. Liu, L. Yang *et al.*, "Advancements in Accelerating Deep Neural Network Inference on AIoT Devices: A Survey," *IEEE Transactions on Sustainable Computing*, pp. 1–18, 2024.

[12] T. J. Ham, S. J. Jung, S. Kim, Y. H. Oh *et al.*, "A^3: Accelerating Attention Mechanisms in Neural Networks with Approximation," in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2020, pp. 328–341.

[13] G. Islamoglu, M. Scherer, G. Paulin, T. Fischer *et al.*, "ITA: An Energy-Efficient Attention and Softmax Accelerator for Quantized Transformers," in *2023 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, 2023, pp. 1–6.

[14] A. Marchisio, D. Dura, M. Capra, M. Martina *et al.*, "SwiftTron: An Efficient Hardware Accelerator for Quantized Transformers," in *2023 International Joint Conference on Neural Networks (IJCNN)*, 2023, pp. 1–9.

[15] S. Lu, M. Wang, S. Liang, J. Lin *et al.*, "Hardware Accelerator for Multi-Head Attention and Position-Wise Feed-Forward in the Transformer," in *2020 IEEE 33rd International System-on-Chip Conference (SOCC)*, 2020, pp. 84–89.

[16] T. Li, F. Zhang, X. Fan, J. Shen *et al.*, "Unified Accelerator for Attention and Convolution in Inference Based on FPGA," in *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2023, pp. 1–5.

[17] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-Attention with Relative Position Representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*, Jun. 2018, pp. 464–468. [Online]. Available: https://aclanthology.org/N18-2074

[18] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language Modeling with Gated Convolutional Networks," in *Proceedings of the 34th International Conference on Machine Learning*, vol. 70. PMLR, 06–11 Aug 2017, pp. 933–941.

[19] A. Zeyer, T. Alkhouli, and H. Ney, "RETURNN as a generic flexible neural toolkit with application to translation and speech recognition," in *Proceedings of ACL 2018, System Demonstrations*. Association for Computational Linguistics, Jul. 2018, pp. 128–133. [Online]. Available: https://aclanthology.org/P18-4022

[20] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 5206–5210.

[21] C. Latotzke, B. Balim, and T. Gemmeke, "Post-Training Quantization for Energy Efficient Realization of Deep Neural Networks," in *2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2022, pp. 1559–1566.

[22] M. Zeineldeen, A. Zeyer, R. Schlüter, and H. Ney, "Chunked Attention-Based Encoder-Decoder Model for Streaming Speech Recognition," in *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2024, pp. 11 331–11 335.

[23] Y. Chen, J. Lou, M. Wabnitz, J. Loh *et al.*, "EDEA: Efficient Dual-Engine Accelerator for Depthwise Separable Convolution with Direct Data Transfer," in *2024 IEEE 37th International System-on-Chip Conference (SOCC)*, 2024, pp. 1–6.

[24] T. J. Ham, Y. Lee, S. H. Seo, S. Kim *et al.*, "ELSA: Hardware-Software Co-design for Efficient, Lightweight Self-Attention Mechanism in Neural Networks," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, 2021, pp. 692–705.

[25] T. Tambe, E.-Y. Yang, G. G. Ko, Y. Chai *et al.*, "9.8 A 25mm2 SoC for IoT Devices with 18ms Noise-Robust Speech-to-Text Latency via Bayesian Speech Denoising and Attention-Based Sequence-to-Sequence DNN Speech Recognition in 16nm FinFET," in *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 64, 2021, pp. 158–160.

[26] Y. Wang, Y. Qin, D. Deng, J. Wei *et al.*, "A 28nm 27.5TOPS/W Approximate-Computing-Based Transformer Processor with Asymptotic Sparsity Speculating and Out-of-Order Computing," in *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 65, 2022, pp. 1–3.

[27] B. Keller, R. Venkatesan, S. Dai, S. G. Tell *et al.*, "A 17–95.6 TOPS/W Deep Learning Inference Accelerator with Per-Vector Scaled 4-bit Quantization for Transformers in 5nm," in *2022 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits)*, 2022, pp. 16–17.

[28] M. Xu, A. Jin, S. Wang, M. Su *et al.*, "Conformer-based speech recognition on extreme edge-computing devices," *arXiv preprint arXiv:2312.10359*, 2023.

[29] "Nvidia - AI pipeline," accessed: 2024-11-18. [Online]. Available: https://developer.nvidia.com/deep-learning-performance-training-inference/conversational-ai