RECONFORMER: A Multi-Level Run-Time Reconfigurable System-on-Chip for Accelerating Transformers

Je Yang, Gabriele Tombesi, Joseph Zuckerman and Luca P. Carloni
{je.yang, gtombesi, jzuck, luca} @ cs.columbia.edu
Department of Computer Science · Columbia University in the City of New York, New York, NY, 10027

Abstract—**Recent advances in neural network design have emphasized attention-based Transformer models, which deliver state-of-the-art accuracy across applications in natural language processing and computer vision. However, the computational demands of Transformers pose significant challenges for deployment in latency-sensitive scenarios and resource-constrained devices, resulting from Transformers' quadratic scaling to sequence length in attention mechanism and extensive data movement. Many works have tried to alleviate these limitations while failing to give comprehensive solutions that consider the heterogeneous characteristics within Transformers.**

In this paper, we propose RECONFORMER, a system-on-chip that accelerates Transformers through multi-level reconfigurability. At component level, RECONFORMER introduces reconfigurable processing elements by incorporating range-based approximations for non-linear functions, resulting in 21% reduced logic usage per accelerator. At the system level, RECONFORMER has a run-time coarse-grained reconfigurability, which provides adaptive parallelism strategies and dynamic cache-coherence mode to address diverse kernel requirements. As a result, RECONFORMER achieves up to 5.54× speedup in multi-head attention and 3.61× in end-to-end performance improvement over a static system. RECONFORMER attains remarkable performance compared to other platforms, resulting in 13.03× and 5.30× higher efficiency than edge and server CPUs, respectively. Even compared to an edge GPU, RECONFORMER achieves 1.79× improvement in efficiency, highlighting our system as a compelling alternative for edge deployment.

Index Terms—**Natural Language Processing; Attention; Domain-Specific Accelerator; Reconfigurable Accelerator; System-on-Chip; FPGA;**

I. INTRODUCTION

Natural language processing has seen rapid advancements in recent years, driven by attention-based models such as Transformer [1], BERT [2], and GPT [3]. These models have demonstrated substantial performance improvements compared to traditional approaches relying on convolutional and recurrent neural networks, with BERT even surpassing human-level accuracy in challenging sentence-classification tasks. Unfortunately, the enhanced accuracy of Transformers comes at a significant cost in computational efficiency. The attention mechanism in Transformers is inherently complex, exhibiting quadratic scaling with input sequence length and requiring substantial data movement, resulting in low arithmetic intensity. In practical terms, these complexities manifest themselves with a GPT-2 model taking 370 milliseconds to generate a 30-word sentence on a NVIDIA TITAN GPU - approximately two orders of magnitude slower than the 6-millisecond image classification of MobileNet-V2 [4]. Such inefficiencies hinder the deployment of Transformers on embedded devices, where computational power, memory, and bandwidth are limited.

Given these constraints, a comprehensive understanding of the design choices for Transformers on edge devices is paramount. Previous works optimizing Transformers with algorithmic approaches, such as quantization, compression, and approximation techniques, often sacrifice model accuracy [5, 6]. Architectural solutions like distributed parallel computing have been proposed to reduce memory bottlenecks while requiring heavy synchronization across devices [7, 8]. Other works have focused on fusing multiple operations in the attention module to minimize external memory accesses to intermediate values [9, 10]. Unfortunately, these efforts often target specific aspects of the efficiency challenge, missing the broader need for adaptability in heterogeneous workloads across diverse kernel types.

Transformers involve complex computational patterns – from matrix-matrix operations to non-linear/element-wise computations – requiring careful orchestration of parallelism and operation fusion to reach ideal performance. Furthermore, a dedicated cache-management approach is critical to optimize system-level data reuse because memory-access patterns vary heavily in Transformers. In BERT, for example, as the sequence length increases from 128 to 512, the portion of memory operations for vector-matrix multiplication is nearly halved, while each memory and computation operation of matrix-matrix multiplication increases by 2.8× and 3.3×, respectively [11]. Thus, a robust solution must address these diverse needs, particularly for applications on embedded platforms where maximizing the capabilities of limited resources is critical.

In this paper, we propose the RECONFORMER system-on-chip (SoC), which accelerates Transformer-based models by dynamically exploiting both component-level and system-level reconfigurability. Our SoC addresses the diverse computational and memory demands of Transformer models through:

- **Component Reconfigurability.** We introduce *reconfigurable processing elements* (RPE), which are designed by decomposing the non-linear functions to general element-wise operations through our range-based approximation. With reusable operation units and reconfigurable dataflows for each operation type, RECONFORMER reduces logic requirements by 21% per accelerator, thus supporting high scalability.

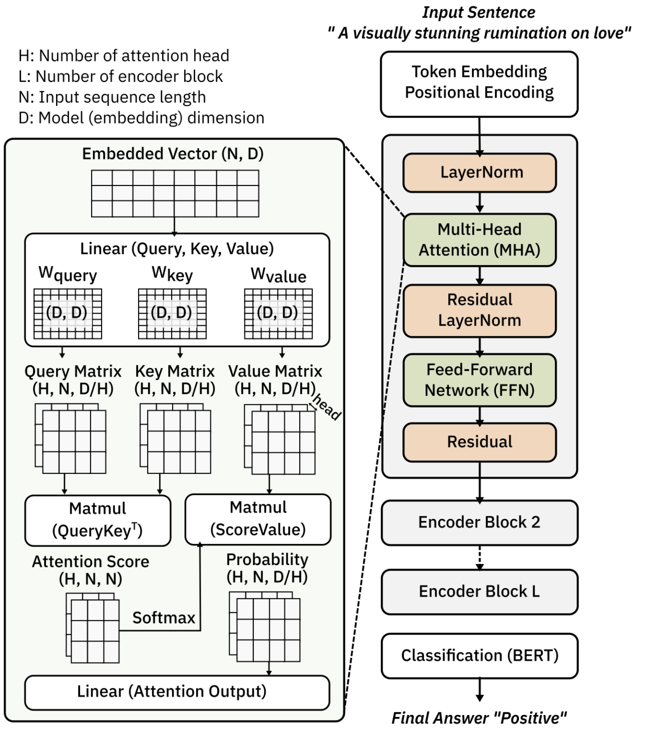


Fig. 1. Transformer encoder structure for sentence-classification task, with an example of N=3, D=8, and H=2.

- **System Reconfigurability.** ReconFormer delivers run-time coarse-grained system reconfigurability by providing adaptive parallelism and dynamic cache management. This is the first work that leverages *accelerator cache-coherence modes* for Transformers, yielding up to 5.54× speedup for multi-head attention and 3.61× for end-to-end performance compared to the static system.

For BERT benchmark applications, the ReconFormer SoC achieves 730.18× and 1.67× higher throughput, as well as 13.03× and 5.30× higher efficiency, compared to edge-class CPU and server-class CPU platforms, respectively. Even compared to an edge GPU, we attain 1.79× higher efficiency, highlighting our SoC as a promising solution for edge deployments of Transformers.

II. Background

A. Attention-Based Language Models

The Transformer architecture has become the foundation for various natural language processing tasks. Fig. 1 shows how the Transformer encoder performs sentence-classification tasks, including a *multi-head attention (MHA)* module and a *feed-forward network (FFN)* module, each followed by a layer normalization operation and a residual connection. First, the input sentences are converted into vector representations ($N \times D$) through token embedding and positional encoding, where N and D represent the input sequence length and model dimension, respectively. Then, the MHA module performs linear projection of this embedded vector by multiplying it with three different weight matrices ($W_{query}, W_{key}, W_{value}$) and yields three different matrices (query, key, and value). Then, these three matrices are split head-wise into H chunks, with each chunk having a dimension of $N \times D/H$. For each of H different attention heads, matrix multiplication and softmax operations are performed to find similarities within a given sentence. Then, all matrices from the attention heads are concatenated and projected with W_{out} to compute the final attention output. While the MHA module consists of four vector-matrix operations ($W_{query}, W_{key}, W_{value}$ and W_{out}) and two matrix-matrix operations (query×key^T and attention score×value), the FFN module is a relatively simple block, involving two fully-connected layers with larger dimensions and a *Gaussian Error Linear Unit (GELU)* activation. The operations within an encoder block are repeated L times, where L indicates the number of encoder blocks.

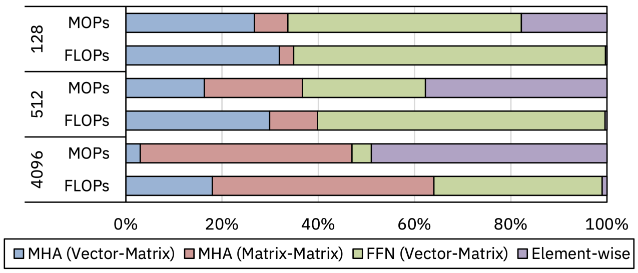


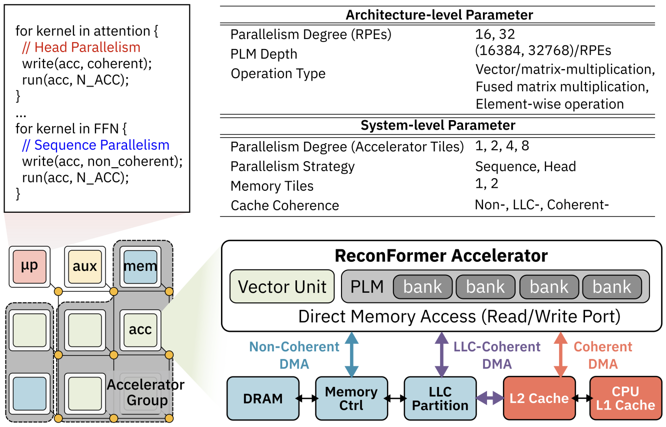
Fig. 2. Per-Layer FLOPs and MOPs for the BERT-Base with sequence lengths of 128, 512, and 4096. In our analysis, we ignore the maximum sequence length 512 for the standard BERT model.

B. Heterogeneity of Transformer

To identify performance bottlenecks in Transformer models, we profile the number of floating-point operations (FLOPs) and the total number of memory operations (MOPs) that are required when executing the Transformer architecture. We conduct our characterization study using BERT-Base on a single NVIDIA GPU by employing PyTorch 2.3.1. Fig. 2 presents the per-layer breakdown of FLOPs and MOPs according to the sequence length for the BERT-Base encoder. For shorter sequences, vector-matrix operations in MHA and FFN dominate FLOPs and MOPs. However, with longer sequences, matrix-matrix multiplications become the primary source of FLOPs, while element-wise operations contribute most to MOPs. This variation across layers and sequence lengths highlights the need for adaptable hardware architectures. To address these heterogeneous computational demands, ReconFormer incorporates (1) reconfigurable processing elements as accelerators that efficiently handle both matrix and non-linear/element-wise operations and (2) run-time coarse-grained reconfigurability for system strategies like adaptive parallelism and dynamic cache-coherence mode. By adopting a multi-level reconfigurability, ReconFormer provides a flexible and scalable solution to optimize performance and energy efficiency in accelerating Transformer-based workloads.

III. ReconFormer System-on-Chip

We designed our ReconFormer system as a tile-based heterogeneous system-on-chip (SoC). Fig. 3 shows a 3×3-



```
for kernel in attention {
  // Head Parallelism
  write(acc, coherent);
  run(acc, N_ACC);
}
...
for kernel in FFN {
  // Sequence Parallelism
  write(acc, non_coherent);
  run(acc, N_ACC);
}
```

Architecture-level Parameter	
Parallelism Degree (RPEs)	16, 32
PLM Depth	(16384, 32768)/RPEs
Operation Type	Vector/matrix-multiplication, Fused matrix multiplication, Element-wise operation
System-level Parameter	
Parallelism Degree (Accelerator Tiles)	1, 2, 4, 8
Parallelism Strategy	Sequence, Head
Memory Tiles	1, 2
Cache Coherence	Non-, LLC-, Coherent-

Fig. 3. An example of 3×3 tiled RECONFORMER system-on-chip with multi-level reconfigurability.

tile instance of the RECONFORMER SoC with 4 instances of RECONFORMER accelerators.

A. Coarse-Grained Reconfigurable System-on-Chip

The RECONFORMER SoC integrates multiple accelerator and memory tiles along with a host processor tile and an auxiliary tile through a 2D-mesh network-on-chip. Multiple accelerator and memory tiles can be dynamically configured as an `accelerator group`, depending on the network traffic of each kernel in a Transformer. RECONFORMER leverages the cache hierarchy to reduce the energy and latency associated with external memory accesses. For each kernel, we dynamically select one of three accelerator coherence mode, as classified in the literature [12]. `Non-coherent` accelerators access DRAM via direct memory access (DMA), bypassing the cache hierarchy. `LLC-coherent` accelerators route DMA requests to the last-level cache (LLC), where hits are returned directly to the accelerator; since the accelerator is only coherent with the LLC, software must flush private caches before its execution begins. `Coherent` accelerators also send their requests to the LLC, but full hardware coherence is maintained, and the LLC may need to recall or invalidate data in private caches. The accelerators are designed based on a loosely-coupled accelerator approach to execute coarse-grain computational kernels by operating on large amount of data [13]. They are invoked by software applications through a device driver, which writes run-time system parameters, such as operation type, operation dimension, cache-coherence mode, number of accelerators, etc., for each coarse-grained kernel. Fig. 3 characterizes the design flexibility of the RECONFORMER SoC by summarizing the architecture component-level and system-level parameters.

B. Accelerator Design

Fig. 4 illustrates the microarchitecture of the RECONFORMER accelerator. We designed and optimized it using SystemC and high-level synthesis (HLS) with the Catapult HLS tool. The design includes four main SystemC processes that communicate through a highly customized, multi-bank,

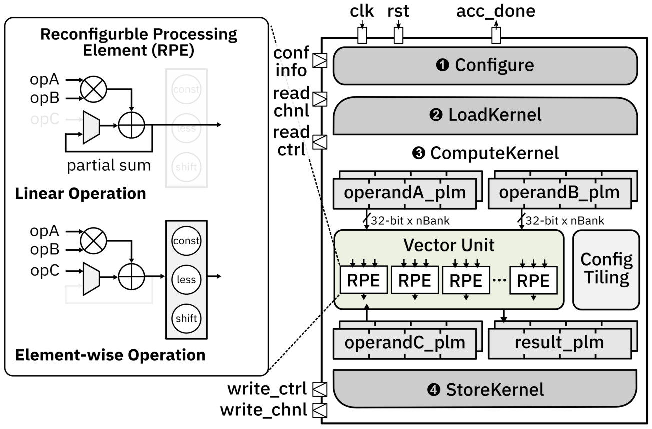


Fig. 4. Microarchitecture of the RECONFORMER accelerator.

multi-port private local memory (PLM) [14]. The first of these processes, `Configure`, sets up the accelerator's pipeline through memory-mapped registers, allowing a high degree of configurability for each invocation. Next, the `LoadKernel` sends DMA read requests to an external DMA engine, storing the fetched data in the PLM. It employs double-buffering to pipeline the load operations with subsequent compute and store operations. Once the required chunk of kernel data is fetched, the `ComputeKernel` process reads data from the PLMs for each operand, and performs the target computations using a vector unit that has a customized reconfigurable datapath. Finally, the `StoreKernel` issues DMA write requests to the external DMA engine, transferring the computed data by reading from the result PLM.

The `ComputeKernel` process is responsible for the main operation of the Transformer. First, `ConfigTiling` calculates the required iteration for memory and computation tiling based on the dimension of each kernel, vector length, and PLM capacity. Then, a 1-dimensional vector of *reconfigurable processing elements (RPEs)* performs the main workload, vector/matrix-matrix multiplication or element-wise operation, with fixed-point precision. Considering the relatively low operation intensity of the Transformer, we set the number of PLM banks equal to the vector length to hide data feeding latency. At design time, both the number of RPEs and the size of the PLM are configurable. In Section IV-B, we explore a design space where the number of RPEs is either 16 or 32, and the size of a single PLM is either 16,384 words (64 KB) or 32,768 words (128 KB), respectively.

C. Range-Based Approximation for Non-linear Functions

Exponential functions and Gaussian Error Linear Units (GELU) are two types of non-linear element-wise operations commonly used in Transformer. To address the resource-intensive nature of these operations, we propose a range-based approximation approach and integrate this approximation technique into the RPE's datapath.

Fig. 5 illustrates the input data distribution for both the exponential functions in MHA and the GELU function in the FFN across BERT-Base's encoder layers. The heat maps

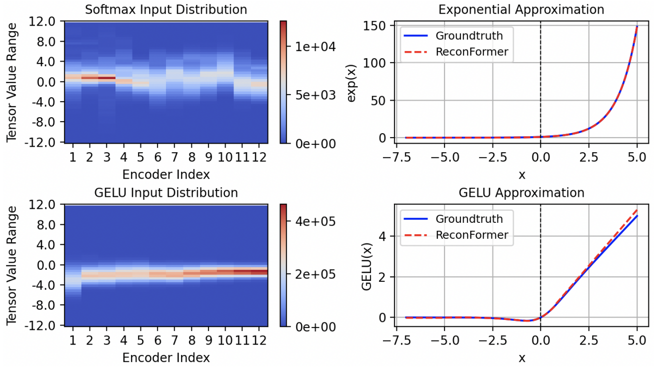


Fig. 5. Input distribution and results of proposed range-based approximation.

TABLE I
END-TO-END ACCURACY FOR BERT WITH PROPOSED ACCURACY. BLEU SCORE IS USED FOR THE MEASUREMENT.

	Baseline	**Range-based Approximation**
BERT-Base	89.10%	89.07%
BERT-Large	72.87%	72.95%

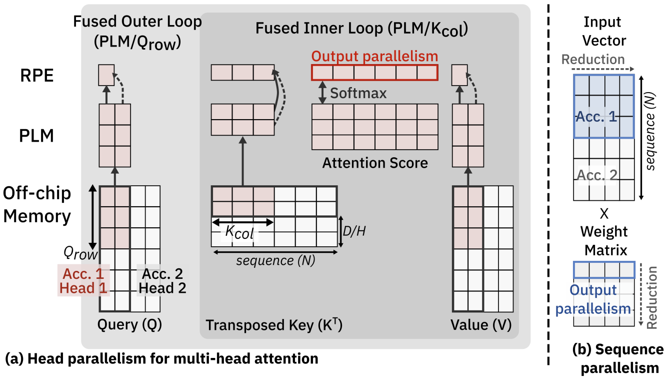
show the input frequency for each range of tensor values, with red indicating higher frequency and blue indicating lower frequency. We observe that the input value of each function lies primarily within specific ranges, such as from -4 to 0 for the GELU function. Based on this insight, we designed range-specific approximation strategies for these non-linear functions to achieve both accuracy and resource efficiency.

To approximate the exponential function, we employ a linear transformation followed by a biased exponentiation method. Specifically, we map the input range into a linear domain, apply an integer transformation using $a = 1/ln(2)$, and introduce two biases b, c to minimize the mean squared error against the true exponential values. The approximation is represented by $exp(x) = 2^{(ax+b)} + c$, which can be easily implemented by using a shifter as we exploit fixed-point precision. For the GELU approximation, we employ a piecewise approximation approach that uses a quadratic polynomial fit for specific segments within the critical range and a linear approximation for outlying values.

Table I lists the end-to-end accuracy of the BERT model with our proposed approximation method. While showing similar precision to the ground truth, this method requires 42K fewer LUTs and 19K fewer registers than employing a non-linear function provided by the Catapult HLS library when targeting an FPGA implementation. Given that our design consumes over 200K LUTs and 250K registers per accelerator, these 21% savings are crucial for scaling the RECONFORMER architecture by accommodating multiple accelerators.

D. Reconfigurable Processing Element (RPE) and Adaptive Parallelism

Using the proposed range-based approximation approach, we simplify resource-intensive non-linear functions into more efficient element-wise operations. We designed an RPE that integrates these element-wise operations with a multiply-and-accumulate dataflow for linear computations, such as vector or matrix multiplications. As shown in Fig. 4, the RPE adapts its dataflow based on the operation type. The linear operations use an output-stationary dataflow to accumulate results. The same multipliers and adders are reused for element-wise operations, with additional components like comparators and shifters to support our range-based approximated non-linear functions.



(a) Head parallelism for multi-head attention

(b) Sequence parallelism

Fig. 6. Inter-/intra-accelerator level adaptive parallelism and operation fusion.

Since RECONFORMER supports all operations in a single RPE, we can employ various parallelism schemes with maximum hardware utilization. We strategically apply different parallelism techniques at both inter-accelerator and intra-accelerator levels. At the inter-accelerator level, we utilize head parallelism for multi-head attention by performing the matrix multiplication for each head independently. For other operations, such as vector-matrix multiplications and element-wise operations, we adopt sequence parallelism to maximize cache locality for weight matrices by setting the LLC size sufficiently large to store parameters for each kernel in Transformer. At the intra-accelerator level, we leverage output parallelism among the RPEs for all operations.

Fig. 6 illustrates how we tile the data and feed it to the RPEs by using adaptive parallelism tailored to each kernel. In the multi-head attention layer (Fig. 6(a)), we divide multiple heads to apply the softmax function row-wise across the full sequence length. The query matrix (Q) is pre-partitioned along the head dimension at the inter-accelerator level, and then the segments of Q_{row} are fetched to fit within the PLM's capacity. Meanwhile, the transposed key matrix (K^T) is divided along the column dimension K_{col}. To compute the attention scores, we prioritize iteration on the K_{col} dimension for PLM/K_{col} times to generate the results for the entire sequence length. After computing the multiple row-wise segments of attention scores, we directly apply softmax and perform the final matrix-matrix multiplication with the value matrix (V). We iterate these outer fusion loops for PLM/Q_{row} times. Fusing these two matrix multiplications and softmax without storing intermediate values dramatically minimizes the cost of accessing external memory because the intermediate attention score size has a quadratic relationship with sequence length. For general linear operations (Fig. 6(b)), we adopt sequence parallelism

TABLE II
RECONFORMER RESOURCE USAGE PER TILE.
ACCELERATOR TILE IS CONFIGURED (RPES, PLM WORD)

Component (Total)	**LUT** (4,085,760)	**FF** (195,527)	**BRAM** (2,160)	**DSP** (3,840)
Accelerator (16, 16384)	176,642	195,527	161	528
Accelerator (16, 32768)	171,790	183,819	321	530
Accelerator (32, 16384)	240,422	252,090	161	748
CPU Tile	53,141	40,285	49.5	27
Memory Tile	20,525	21,557	100.5	0
IO Tile	8,935	12,032	32	0

and configure the LLC size sufficiently large to fit the weight matrix of a single kernel. By prioritizing sequence parallelism over output parallelism, we maximize cache locality and make full use of the available cache capacity.

IV. EVALUATION

A. *Experimental Setup*

Workload. We evaluate the efficiency of our proposed RECONFORMER with the fundamental backbone of the language model, BERT [2]. We focus on BERT-Base ($L = 12, D = 768, H = 12$) and BERT-Large ($L = 24, D = 1024, H = 16$), which have 110M and 340M parameters, respectively.

System Implementation. We prototyped our RECONFORMER SoC on a proFPGA UltraScale+ XCVU19P board by leveraging ESP [15], an open-source platform for designing heterogeneous SoCs with tile-based architectures [16]. We integrated four accelerator tiles, two memory tiles – each with a LLC partition – and a single tile each for CPU and IO. Table II summarizes the resource usage of each tile and accelerator configuration, denoted (`#RPEs`, `PLM Words`).

B. RECONFORMER *Ablation Study*

Intra/Inter-Accelerator Parallelism. We evaluate how the intra-accelerator parallelism (number of RPEs), inter-accelerator parallelism (number of accelerators), and memory bandwidth (number of memory tiles) impact the performance across different kernels. For a fair comparison, we also assess the resource utilization of each system as the percentage of the total resources of the FPGA used by all accelerators and memory tiles, as summarized in Table II.

Fig. 7 illustrates the latency-area trade-offs across various degrees of both intra- and inter-accelerator parallelism for each kernel, annotated with the dimension of the largest operand size for each operation. We evaluated the cycle count for each kernel using three distinct accelerator configurations by varying intra-accelerator parallelism and on-chip capacity, starting from the baseline design `(RPE 16, PLM 16384)`. Although both the baseline and `(RPE 32, PLM 16384)` accelerators have the same on-chip memory capacity, the latter incorporates a greater number of PLM banks with reduced depth, providing higher output parallelism, but with reduced data availability along the reduction dimension. In particular, performance with `(RPE 32, PLM 16384)` is even slower than the baseline across all kernels, indicating that reducing transactions along the reduction dimension is more critical than maximizing output generation. This observation highlights the importance of selecting a parallelism strategy tailored to the operation's requirements, further supporting the efficacy of our adaptive parallelism approach by prioritizing sequence parallelism over output parallelism.

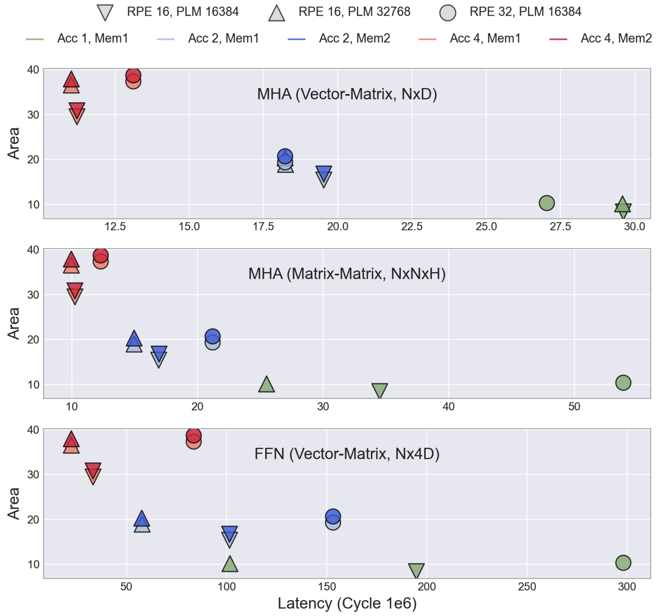


Fig. 7. Latency-area trade-offs for the Transformer kernel with different accelerator designs.

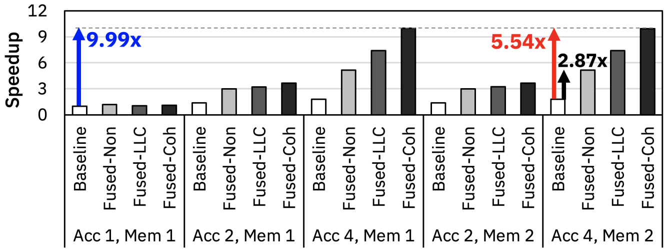


Fig. 8. Performance improvement of multi-head-attention kernel with various memory optimization strategies.

Intra-accelerator parallelism exerts limited influence on kernels with moderate compute and memory requirements, such as vector-matrix multiplications in MHA. However, performance varies significantly across configurations for low-operation-intensity kernels, such as matrix multiplications in MHA, and for larger-dimension kernels. Additionally, while increasing the number of memory tiles generally reduces latency across all kernels, this improvement is minimal relative to the associated increase in resource consumption.

Operation Fusion and Cache Management. Fig. 8 shows the impact of operation fusion and cache management on a MHA module. We evaluate the speedup across different memory optimization strategies: `Baseline` (no fused operations

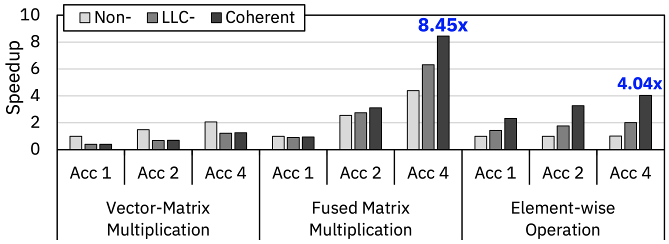


Fig. 9. Performance improvement of each Transformer kernel with coherence mode.

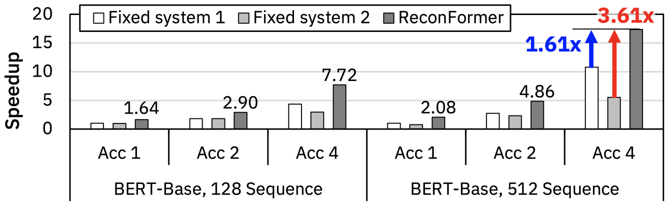


Fig. 10. End-to-end performance improvement with ReconFormer multi-level reconfigurability. Each fixed system 1 and 2 consists of (RPE 16, PLM 16384) and (RPE 32, PLM 16384) without coherence mode while ReconFormer consists of (RPE 16, PLM 32768) with mixed coherence mode.

TABLE III
Comparing ReconFormer to other platforms, **B** and **L** of throughput row and efficiency row represent the performance of BERT-Base and BERT-Large, respectively.

		Edge GPU	Edge CPU	Server CPU	**ReconFormer**
Compute Unit		1.2GHz @ 1,024 Cores	1.7GHz @ Single Core	3GHz @ 24 Cores	100MHz @ 4×16 RPEs
On-chip Memory		1MB	16KB	38MB	4×1.3MB
Off-chip Bandwidth		80 GB/s GDDR5	22.5GB/s DDR4	89.6GB/s DDR4	22.5GB/s DDR4
Power (W)		Max 40	0.254	52.25	14.23
Throughput	B	31.55	0.017	7.42	12.40
(seq/s)	L	10.13	0.001	4.18	8.16
Efficiency	B	1.19	0.067	0.16	0.87
(seq/s/W)	L	0.32	0.038	0.08	0.57

and non-coherent accelerators), `Fused-Non` (fused operations non-coherent accelerators), `Fused-LLC` (fused operations and LLC-coherent accelerators), and `Fused-Coh` (fused operations and coherence accelerators) for each accelerator-memory configuration. Operation fusion substantially reduces external memory access overhead for intermediate features, achieving a speedup of 2.87× in the configuration with four accelerator tiles and two memory tiles. Furthermore, ReconFormer enhances performance by switching the accelerator coherence mode, particularly when the number of accelerators increases. In this case, the query, key, and value matrices are partitioned and become compact enough to fit within the LLC, which improves hit rates and enhances the performance of the coherent mode. Consequently, with the four-accelerator, fused, and coherent configuration, ReconFormer is 9.99× faster than the baseline single accelerator, which has a non-fused, and non-coherent configuration. Even compared to a four-accelerator setup without fusion and coherence, ReconFormer shows a substantial improvement, achieving a 5.54× speedup.

Fig. 9 shows how dynamically choosing the accelerator coherence mode affects the performance of each kernel in Transformers. Notably, vector-matrix multiplication does not demonstrate any performance improvement from cache usage for the following reasons. First, sequence parallelism, which is employed at the inter-accelerator level, duplicates the model parameters. The substantial kernel size resulting from this duplication can lead to cache thrashing and increased contention among multiple accelerators. In contrast, the memory-dominant kernels with longer sequence length, like fused MHA and element-wise operation, attain 8.45× and 4.04× speedup compared to a non-coherent single accelerator. These gains eventually lead to end-to-end speedups as shown in Fig. 10. Based on previous results, ReconFormer selects the coherent mode for MHA and element-wise operation kernel while using the non-coherent mode for the remaining kernels. This dynamic management enables the ReconFormer SoC to achieve up to a 3.61× speedup compared to a static configuration using the same number of accelerators in long-sequence scenarios.

C. System-Level Performance Evaluation

Based on the previous ablation study, we set the optimal configuration for the ReconFormer SoC to achieve peak performance as follows: four accelerator tiles with 16 RPEs and 32768 PLM words, two memory tiles each with 0.4MB of LLC, and using the coherent mode for the fused MHA kernel and element-wise operations. To compare the performance and energy consumption of our optimal SoC with state-of-the-art platforms, we evaluate the BERT model with an edge GPU (NVIDIA Quadro T2000), an edge CPU (Ariane RISC-V Core), and a server CPU (Intel i9-13900K). Table III lists the specifications and performance for the above platforms. For a fair comparison, we use a common metric *normalized throughput* (sequence per second) [17]. Although our SoC has limited compute cores and on-chip memory capacity compared to other platforms, it leverages adaptive reconfiguration mechanisms to dynamically optimize the architecture selection. As a result, for the BERT-Base benchmark applications, ReconFormer SoC attains major improvements of 730.18× and 1.67× higher throughput and 13.03× and 5.30× higher efficiency compared to edge CPU and server CPU platforms, respectively. Especially, the 1.79 × improvement in efficiency over the edge GPU for larger model demonstrates that our ReconFormer SoC can be a competitive alternative for edge deployments of Transformers.

D. Scalability and Generalizability

Fig. 11 shows the resource utilization and layout of the optimal configuration of the ReconFormer SoC on FPGA. Overall, it uses 24.5% of LUTs, 13.6% of Registers, 56.5% of DSP, and 81% of BRAM. The balanced utilization of resources

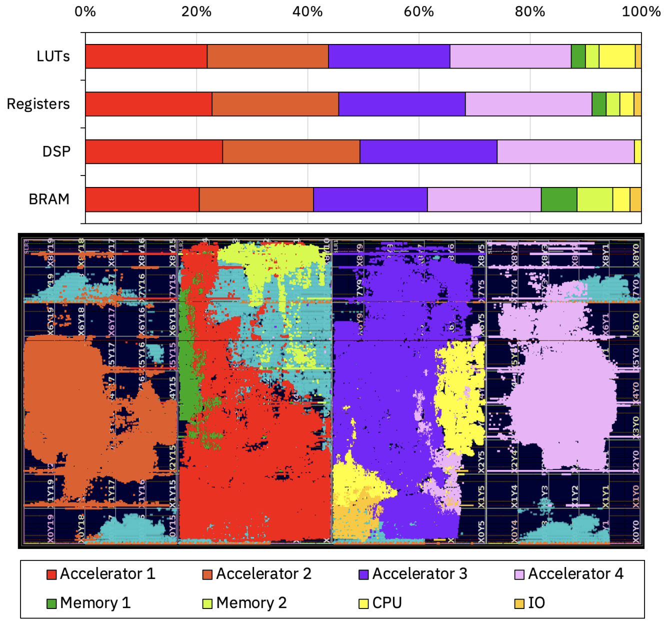


Fig. 11. Resource utilization and layout on FPGA of RECONFORMER with optimal SoC configuration.

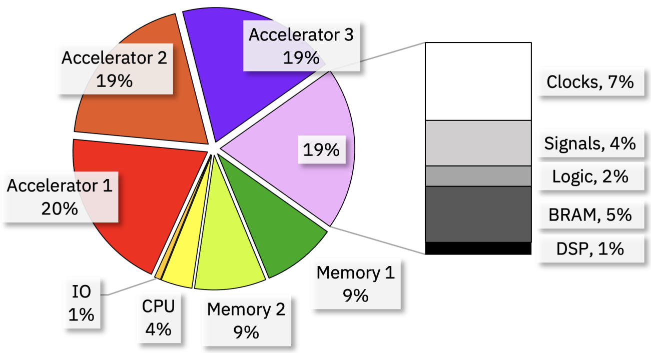


Fig. 12. Power breakdown of RECONFORMER with optimal SoC configuration per each tile and module.

across accelerators suggests that our SoC is well-optimized and can scale without immediate resource contention. Fig. 12 shows the power breakdown of the SoC with optimal configuration. It consumes less dynamic power both in terms of logic power (0.61W) and BRAM power (2.23W); each accelerator tile contributes approximately 1.3 W (19%) and the memory tiles consume only 0.6 W each (9%). The low resource use and power dissipation of the CPU confirm that our SoC is designed to offload computation to accelerators, which is a key for scalability.

Although our benchmarks focused mainly on discriminative models such as BERT, we would like to emphasize that RECONFORMER's reconfigurability is broadly applicable to any Transformer-based models. These models typically feature MHA and FFN modules, both of which benefit significantly from RECONFORMER's system-level dynamic cache management and component-level adaptive datapaths. Moreover, generative models that involve decoding processes with key-value caching in cross-attention are expected to benefit even more from RECONFORMER's dynamic cache-coherence modes, enabling efficient reuse of intermediate results.

TABLE IV
COMPARING SCALABILITY OF RECONFORMER TO OTHER MULTI-CORE ACCELERATOR FOR TRANSFORMERS.

	DFX [7]	DeTransformer [8]	RECONFORMER
Platform	Xilinx U280	Raspberry Pi	UltraScale+ XCVU19P
Benchmark	GPT [3]	BERT [2]	BERT [2]
Scalability	1.57×(2 devices) 2.23×(4 devices)	2.81×(4 devices)	1.77×(2 cores) 4.71×(4 cores)

V. RELATED WORK

Efforts to accelerate Transformers span both algorithmic and architectural strategies. Compression and quantization techniques optimize the model itself to reduce size and attention complexity [4, 5, 18]–[20]. To alleviate the complexity with longer sequence, various approximations of non-linear function in Transformer have been proposed [21]–[23]. FlashAttention [9] and Flat [10] have leveraged operation fusion in MHA layers as architectural solutions. However, these methods address only specific efficiency challenges, often without accounting for the heterogeneous nature of the Transformer workloads, which encompass a mix of compute- and memory-bound operations across diverse kernel types.

Other researchers have focused on distributed parallel computing to scale Transformer models [6]–[8, 11]. For instance, DFX [7] introduces custom instructions and dataflow, efficiently combining multi-device hardware with model parallelism. While these approaches reduce memory bottlenecks, they require complex synchronization across devices, making it difficult to exploit parallelism within Transformers effectively. To address synchronization challenges, DeTransformer [8] highlights block parallelism by reconstructing Transformer layers and retraining the model, achieving up to a 2.81× speedup compared to standard tensor parallelism. In contrast, our approach leverages multi-level reconfigurability without model retraining, providing a flexible and holistic solution for edge devices while surpassing ideal speed by achieving up to a 4.71× improvement. Table IV summarizes the scalability of the multi-core accelerator for Transformers, implying that our system reconfigurability can help the accelerator attain an ideal speedup.

VI. CONCLUSION

In this paper, we propose RECONFORMER, a multi-level reconfigurable system-on-chip designed to accelerate Transformers on resource-constrained edge devices. RECONFORMER exploits a reconfigurable processing element with a range-based approximation approach to reuse the operation units and reconfigure the dataflow for each operation type, thus achieving a 21% reduction in logic usage. In addition, our coarse-grained system-level reconfigurability, which includes adaptive strategies for intra- and inter-accelerator parallelism and dynamic selection of coherence modes, provides a 5.54× speedup in multi-head attention and a 3.61× in end-to-end performance improvement over a static system. Ultimately, when accelerating BERT-Base, the RECONFORMER SoC achieves 730.18× and 1.67× higher throughput values and 13.03× and 5.30× higher efficiency values compared to edge and server CPUs, respectively. By reaching a 1.79× improvement in efficiency over an edge GPU, we show that our RECONFORMER SoC is a promising solution for the edge deployment of Transformers.

ACKNOWLEDGMENT

This work is partially supported by a DOE award (A#: DE-SC0024458) and a Columbia Center of Artificial Intelligence Technology (CAIT) Award.

REFERENCES

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[2] J. Devlin, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[3] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[4] H. Wang, Z. Zhang, and S. Han, "Spatten: Efficient sparse attention architecture with cascade token and head pruning," in *IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pp. 97–110, IEEE, 2021.

[5] T. Yang, D. Li, Z. Song, Y. Zhao, F. Liu, Z. Wang, Z. He, and L. Jiang, "Dtqatten: Leveraging dynamic token-based quantization for efficient attention architecture," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 700–705, IEEE, 2022.

[6] R. Pope, S. Douglas, A. Chowdhery, J. Devlin, J. Bradbury, J. Heek, K. Xiao, S. Agrawal, and J. Dean, "Efficiently scaling transformer inference," *Proceedings of Machine Learning and Systems*, vol. 5, pp. 606–624, 2023.

[7] S. Hong, S. Moon, J. Kim, S. Lee, M. Kim, D. Lee, and J.-Y. Kim, "Dfx: A low-latency multi-FPGA appliance for accelerating transformer-based text generation," in *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 616–630, 2022.

[8] Y. Wei, S. Ye, J. Jiang, X. Chen, D. Huang, J. Du, and Y. Lu, "Communication-efficient model parallelism for distributed in-situ transformer inference," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–6, 2024.

[9] T. Dao, D. Fu, S. Ermon, A. Rudra, and C. Ré, "Flashattention: Fast and memory-efficient exact attention with IO-awareness," *Advances in Neural Information Processing Systems*, vol. 35, pp. 16344–16359, 2022.

[10] S.-C. Kao, S. Subramanian, G. Agrawal, A. Yazdanbakhsh, and T. Krishna, "Flat: An optimized dataflow for mitigating attention bottlenecks," in *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pp. 295–310, 2023.

[11] S. Kim, C. Hooper, T. Wattanawong, M. Kang, R. Yan, H. Genc, G. Dinh, Q. Huang, K. Keutzer, M. W. Mahoney, Y. S. Shao, and A. Gholami, "Full stack optimization of transformer inference: a survey," *arXiv preprint arXiv:2302.14017*, 2023.

[12] J. Zuckerman, D. Giri, J. Kwon, P. Mantovani, and L. P. Carloni, "Cohmeleon: Learning-Based Orchestration of Accelerator Coherence in Heterogeneous SoCs," in *Proceedings of the IEEE/ACM Symposium on Microarchitecture (MICRO)*, 2021.

[13] E. G. Cota, P. Mantovani, G. Di Guglielmo, and L. P. Carloni, "An analysis of accelerator coupling in heterogeneous architectures," in *Proceedings of the Design Automation Conference (DAC)*, pp. 1–6, 2015.

[14] P. Mantovani, G. Di Guglielmo, and L. P. Carloni, "High-level synthesis of accelerators in embedded scalable platforms," in *Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 204–211, 2016.

[15] P. Mantovani, D. Giri, G. Di Guglielmo, L. Piccolboni, J. Zuckerman, E. G. Cota, M. Petracca, C. Pilato, and L. P. Carloni, "Agile SoC development with open esp," in *Proceedings of the International Conference On Computer Aided Design (ICCAD)*, pp. 1–9, 2020.

[16] L. P. Carloni, "The case for embedded scalable platforms," in *Proceedings of the Design Automation Conference (DAC)*, pp. 17:1–17:6, June 2016.

[17] "NVIDIA Data Center Deep Learning Product Performance AI Inference — developer.nvidia.com." https://developer.nvidia.com/deep-learning-performance-training-inference/ai-inference. [Accessed 12-11-2024].

[18] G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth, and S. Han, "Smoothquant: Accurate and efficient post-training quantization for large language models," in *International Conference on Machine Learning*, pp. 38087–38099, PMLR, 2023.

[19] J. Lin, J. Tang, H. Tang, S. Yang, W.-M. Chen, W.-C. Wang, G. Xiao, X. Dang, C. Gan, and S. Han, "AWQ: Activation-aware weight quantization for on-device llm compression and acceleration," *Proceedings of Machine Learning and Systems*, vol. 6, pp. 87–100, 2024.

[20] J. Park, M. Kang, Y. Han, Y.-G. Kim, J. Shin, and L.-S. Kim, "Token-picker: Accelerating attention in text generation with minimized memory transfer via probability estimation," in *Proceedings of the Design Automation Conference*, pp. 1–6, 2024.

[21] J. R. Stevens, R. Venkatesan, S. Dai, B. Khailany, and A. Raghunathan, "Softermax: Hardware/software co-design of an efficient softmax for transformers," in *Proceedings of the Design Automation Conference (DAC)*, pp. 469–474, IEEE, 2021.

[22] G. Shen, J. Zhao, Q. Chen, J. Leng, C. Li, and M. Guo, "Salo: an efficient spatial accelerator enabling hybrid sparse attention mechanisms for long sequences," in *Proceedings of the Design Automation Conference*, pp. 571–576, 2022.

[23] J. Yu, J. Park, S. Park, M. Kim, S. Lee, D. H. Lee, and J. Choi, "NN-LUT: neural approximation of non-linear operations for efficient transformer inference," in *Proceedings of the Design Automation Conference (DAC)*, pp. 577–582, 2022.