Hardware Optimization for Transformers -Suhas

Different categories of Hardware Optimization Overview

1. Pipelining

- **Overview**: Pipelining involves overlapping different computational stages or tasks, such as data transfer and processing. In deep neural networks like transformers, this approach can significantly improve efficiency by balancing workloads and reducing idle times for different computational units.
- **Examples**:
  - PipeBERT: Divides BERT into subgraphs for different processor clusters, achieving load balance and efficiency.
  - Length Adaptive Co-design: Adjusts pipeline stages based on input sequence lengths to optimize resource allocation and throughput.

2. Optimizing Matrix Multiplication

- **Overview**: Matrix multiplication is a core operation in transformers. Optimizing these operations can lead to substantial performance gains.
- **Examples**:
  - Hardware Accelerator by Lu et al.: Uses a systolic array for efficient GEMM operations in MHA and FFN blocks.
  - DFX: A multi-FPGA accelerator for GPT-2, optimizing matrix operations for different stages of the model.

3. Skipping Redundant or Trivial Computations

- **Overview**: By identifying and avoiding computations that don't significantly impact the final output or are repeated, efficiency can be greatly improved.
- **Examples**:
  - ELSA: Estimates attention scores without full computation.
  - DOTA: Predicts attention scores to reduce computation wastage.
  - A^3 and Chen et al.'s techniques: Focus on approximating or avoiding computations with little to no impact on final results.

4. Dataflows for Exploiting Reuse

- **Overview**: Choosing the right dataflow in hardware accelerators can enhance data reuse and reduce memory accesses.
- **Examples**:
  - Output Block Stationary (OBS): Reduces memory access by broadcasting at block and vector levels.
  - Sanger and SALO: Focus on maximizing data reuse for sparse attention models and long input sequences, respectively.
  - ViTCoD: Optimizes dataflow for fixed sparse attention patterns in ViTs.

5. Block-Circulant Matrix for Reducing Weight Storage

- **Overview**: The block-circulant matrix approach compresses weight matrices, reducing storage requirements and computational complexity.
- **Example**:
  - Ftrans: Improves BCM for transformers, balancing on-chip and off-chip storage to enhance performance and reduce model size.

Detailed Paper analysis

1. Pipelining

1.1 PipeBERT

https://dl.acm.org/doi/abs/10.1007/s11265-022-01814-y
Objective: This paper introduces a method for efficiently running BERT (Bidirectional Encoder Representations from Transformers) models, specifically on big.LITTLE processor architectures.

big.LITTLE Architecture: This type of processor architecture combines high-performance cores ("big") with energy-efficient cores ("LITTLE"). The architecture is designed to balance computational power and energy efficiency.

Methodology:

- **Subgraph Division:** The BERT network is divided into two subgraphs. This division is based on the characteristics of the network and the capabilities of the big and LITTLE clusters.
- **Mapping to Processors:** Each subgraph is mapped to either the big or the LITTLE cluster of the processor. This mapping uses the CPU's "affinity" feature, which allows specific parts of a program to be assigned to certain processors.
- **Latency-aware Binary Search Algorithm:** This algorithm is a key feature of the approach. It dynamically adjusts the allocation of operations between the big and LITTLE clusters. The aim is to maintain a balanced load across the clusters.
- **Performance:** This method is found to outperform traditional binary search and brute-force methods in terms of throughput (how much processing can be done in a given time) and energy efficiency.

Applications: The technique is particularly effective on systems like the HiKey970, demonstrating enhanced throughput and energy efficiency.

1.2 Length Adaptive Co-design -A Length Adaptive Algorithm-Hardware Co-design of Transformer on FPGA Through Sparse Attention and Dynamic Pipelining

https://arxiv.org/pdf/2208.03646.pdf

Objective: This paper addresses inefficiencies that occur due to variable input sequence lengths in transformer models.

Challenges Addressed:

- Transformer models often face inefficiencies when dealing with input sequences of varying lengths.
- Traditional CPU and GPU implementations may not handle these variations efficiently.

Methodology:

- **Prediction of Attention Scores:** The method involves predicting relative attention scores at low precision. This step reduces the computational load.
- **Look-Up Table (LUT) for Multiplication:** A LUT is used for efficiently computing the most significant attention scores.
- **Pipeline Stages in Encoder:** The encoder of the transformer model is split into three pipeline stages. This division optimizes the use of on-chip memory and helps in reducing the impact of the 'memory wall' effect (a limitation in computer system design where the speed of memory lags behind the processor speed).
- **Handling Sparse Attention:** The technique is adept at managing sparse attention scenarios. It dynamically reconfigures resource allocation based on the sequence length.
- **Performance:** The approach leads to an improvement in throughput and energy efficiency. It achieves this with minimal loss in accuracy, making it more efficient than traditional CPU and GPU implementations.

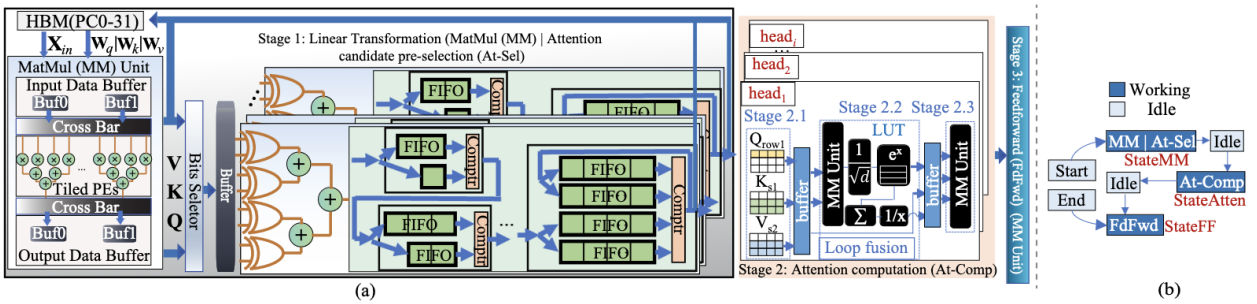


Figure 2: (a) Sparse attention on FPGA; (b) State machine.

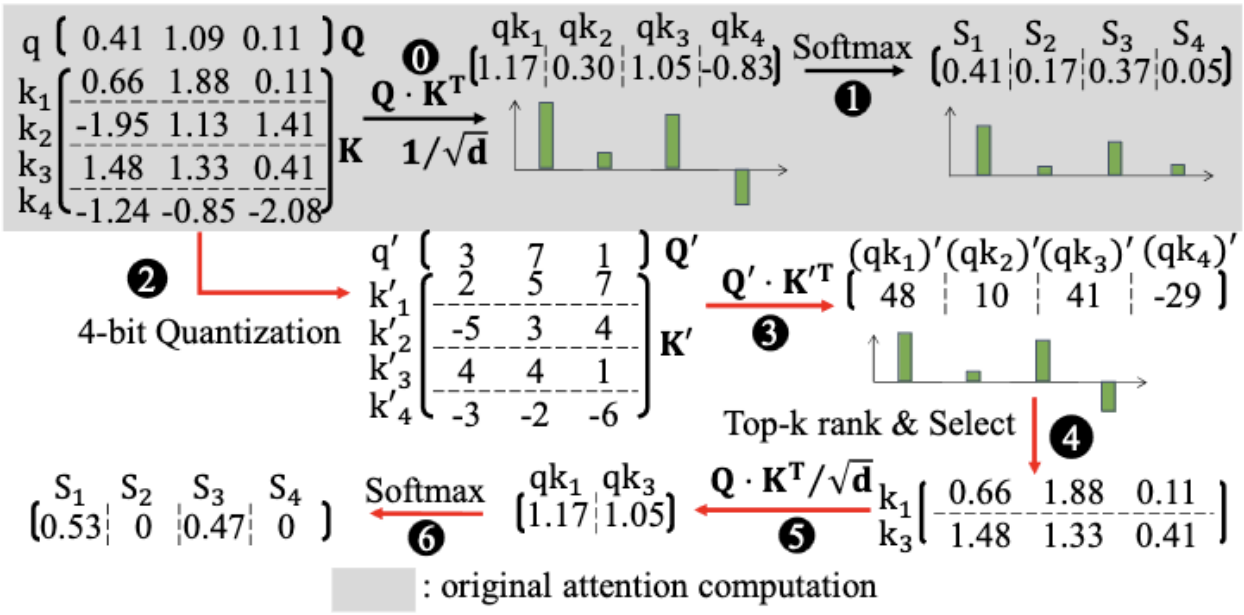


Figure 3: Candidate selection from quantized result.

2. Optimizing Matrix-Multiplication

2.1 Optimizing Matrix-Multiplication Hardware Accelerator for Multi-Head Attention and Position-Wise Feed-Forward in the Transformer

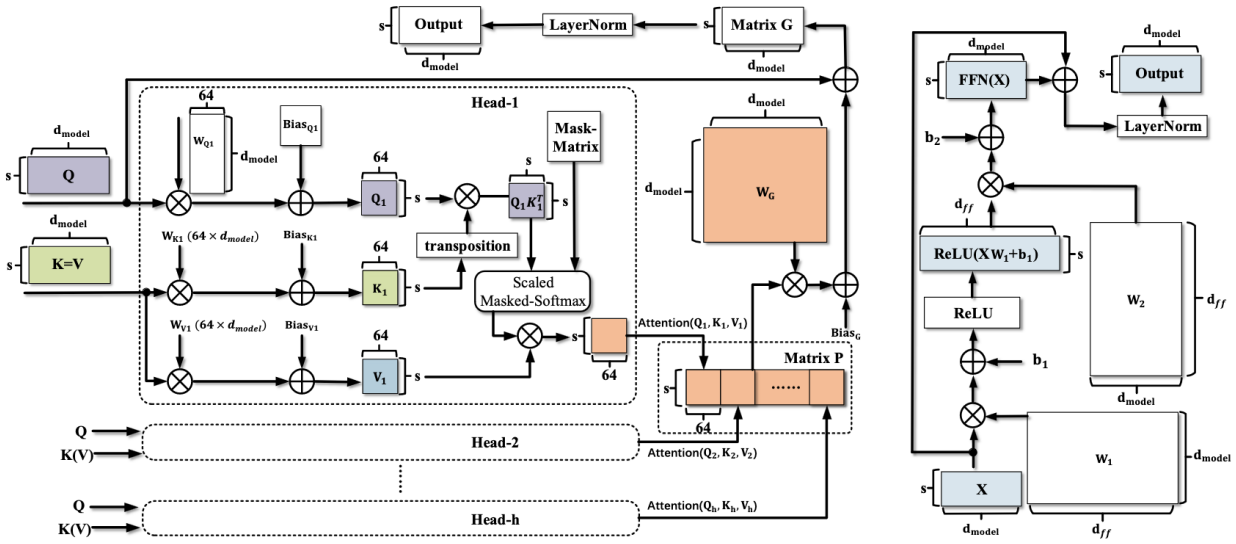
https://arxiv.org/pdf/2009.08605.pdf

Objective: This paper introduces a specialized hardware accelerator specifically designed for the Multi-Head Attention (MHA) and Feed-Forward Network (FFN) blocks of transformers.

Key Innovations:

- **Division of Weight Matrices:** The primary innovation lies in the division of large weight matrices into smaller segments. This allows most General Matrix Multiply (GEMM) operations to be executed efficiently.
- **Systolic Array Utilization:** These divided matrices are processed using a systolic array, a network of processors that perform matrix operations in a coordinated fashion. The size of the systolic array is kept manageable, which is crucial for resource efficiency.
- **Softmax Computation:** The method for computing softmax (a crucial operation in transformers for normalizing attention scores) is optimized. Instead of using Look-Up Tables (LUTs) and multipliers, they opt for a linear approximation method, which enhances computational efficiency.
- **FPGA Implementation:** The implementation of this hardware accelerator on Field-Programmable Gate Arrays (FPGAs) demonstrates a substantial speedup compared to traditional GPU-based methods, validating the effectiveness of their approach.

Impact: This innovation in hardware design optimizes the use of resources and improves the efficiency of executing key operations in transformer models, which is beneficial for applications requiring high computational throughput.



(a) The MHA ResBlock. (b) The FFN ResBlock.

Fig. 3. Matrix Operations in the MHA and the FFN ResBlocks. Note that all the multiply operations marked in this figure are dealing with cross products.

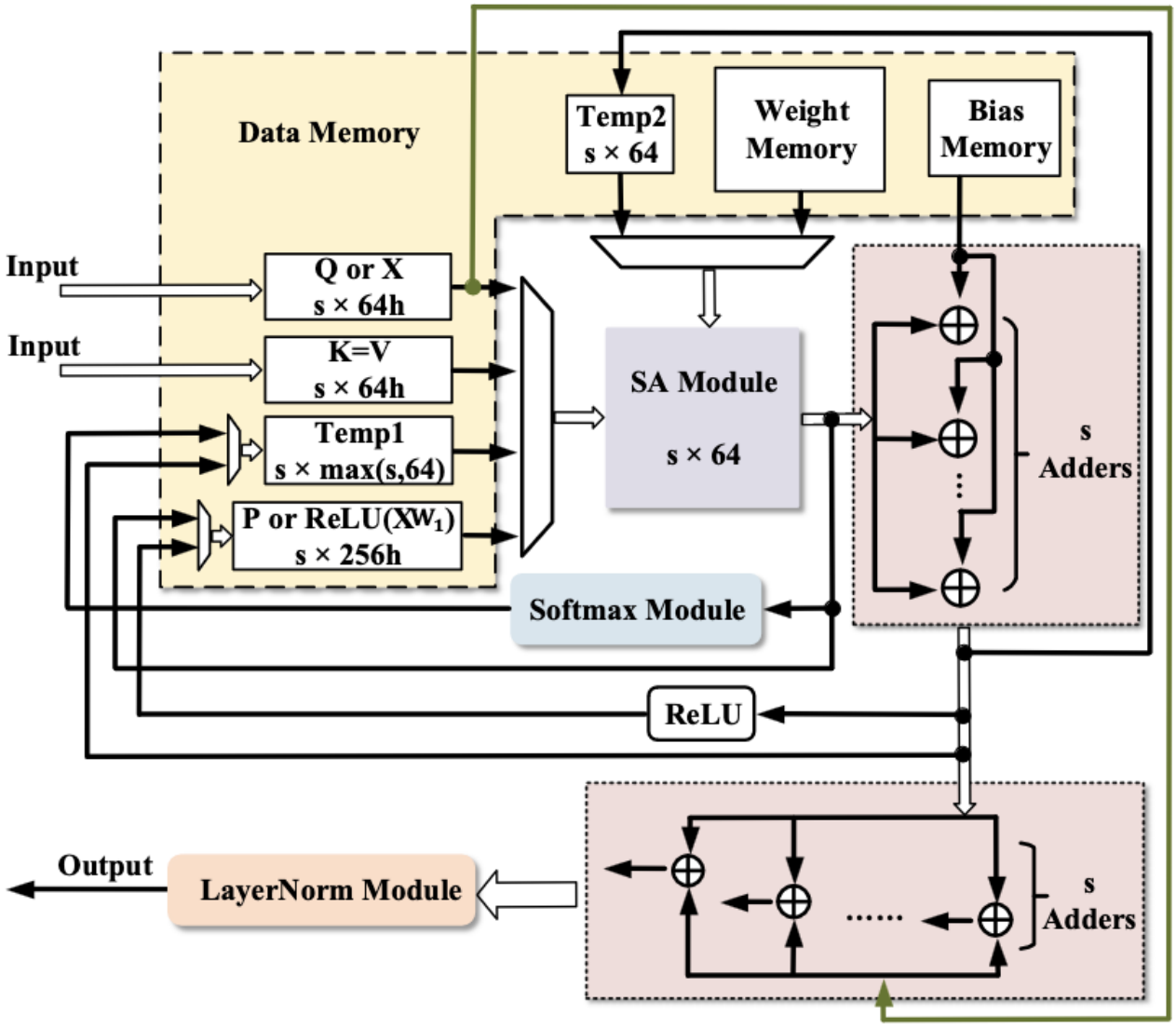
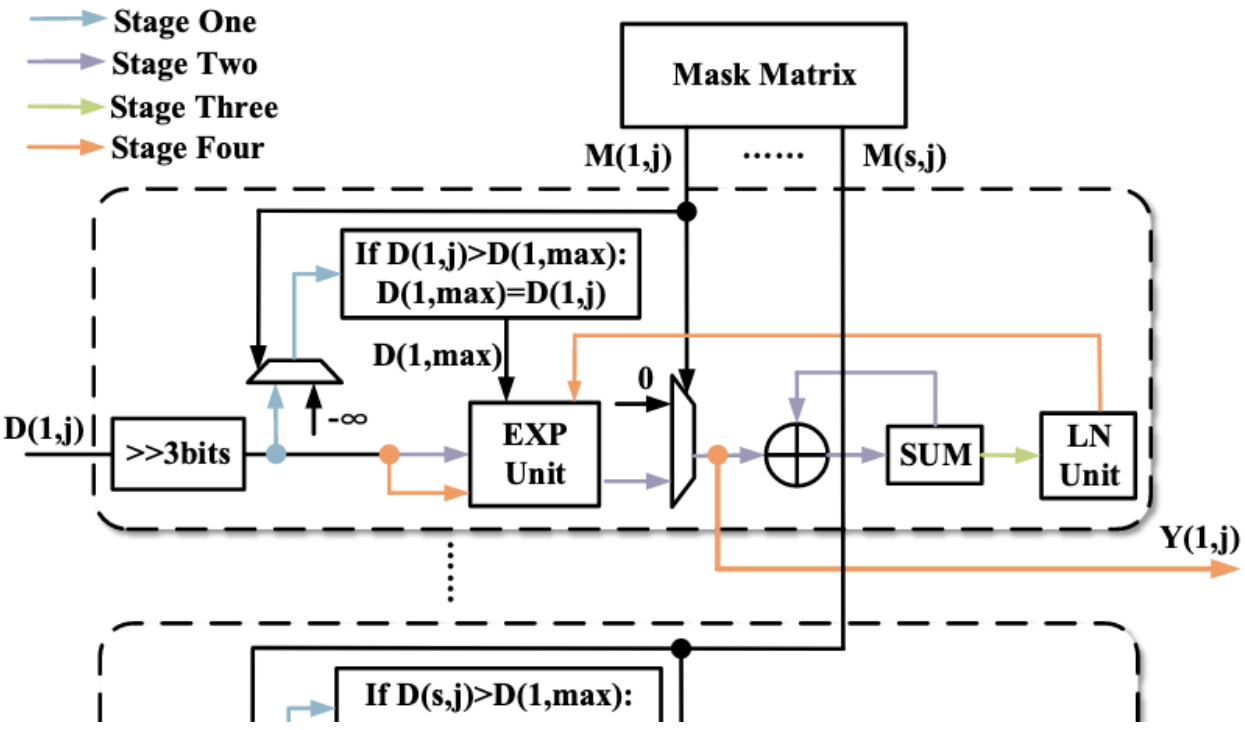


Fig. 5. The top-level architecture of our design.



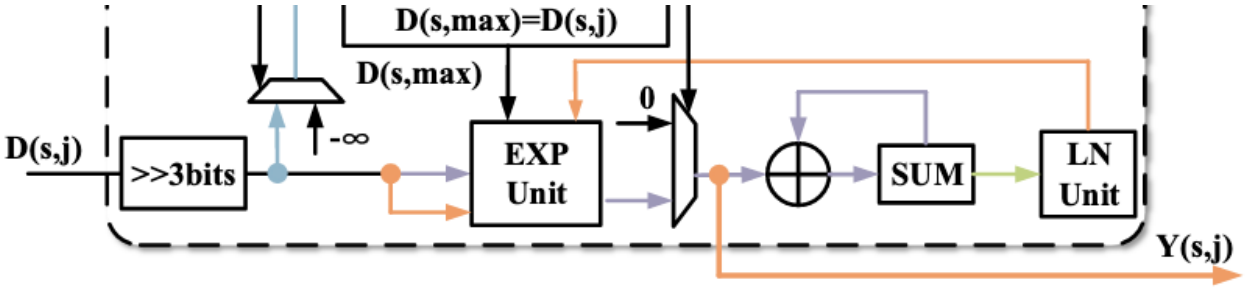


Fig. 6. The architecture of Softmax module. The ">>" denotes right shift operation.

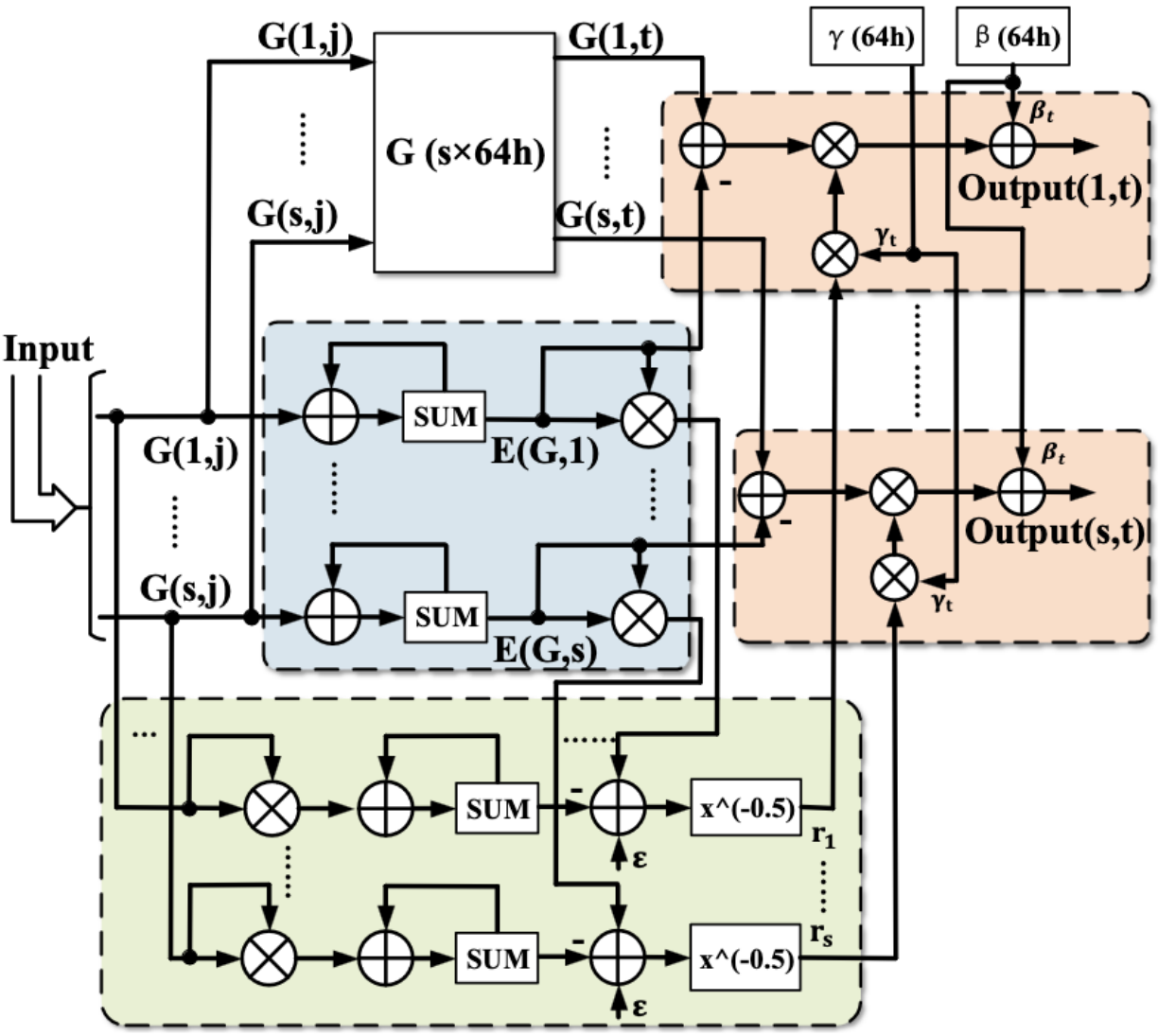


Fig. 8. The architecture of LayerNorm module.

2.2 DFX by Hong et al.

https://www.researchgate.net/publication/363765406_DFX_A_Low-latency_Multi-FPGA_Appliance_for_Accelerating_Transformer-based_Text_Generation

Objective: The DFX framework is aimed at efficiently handling large-scale text-generation models, particularly focusing on GPT-2.

Framework Features:

- **Multi-FPGA Accelerator:** DFX is a multi-FPGA accelerator, designed to manage the heavy computational demands of large text-generation models.
- **Intra-Layer Model Parallelism:** Recognizing the different computational needs of the summarization and generation stages in GPT-2, DFX employs intra-layer model parallelism. This means that different stages of the model are processed in parallel, enhancing efficiency.
- **Optimized Compute Cores:** The compute cores in DFX are specifically optimized for processing individual tokens, a key aspect of the GPT-2 model.
- **Unique Instruction Set Architecture (ISA):** DFX features a specialized ISA that includes matrix, vector, DMA (Direct Memory Access), and router instructions. These are tailored to efficiently execute various operations required by the GPT-2 model.
- **Memory Allocation Strategy:** The framework uses High Bandwidth Memory (HBM) for storing weight matrices, which are frequently accessed, and places less frequently accessed data in DDR memory. This strategic memory allocation contributes to improved performance and energy efficiency.
- **Comparison with GPU Setups:** DFX shows significant improvements in performance and energy efficiency compared to traditional GPU setups, making it a promising solution for large-scale model processing.

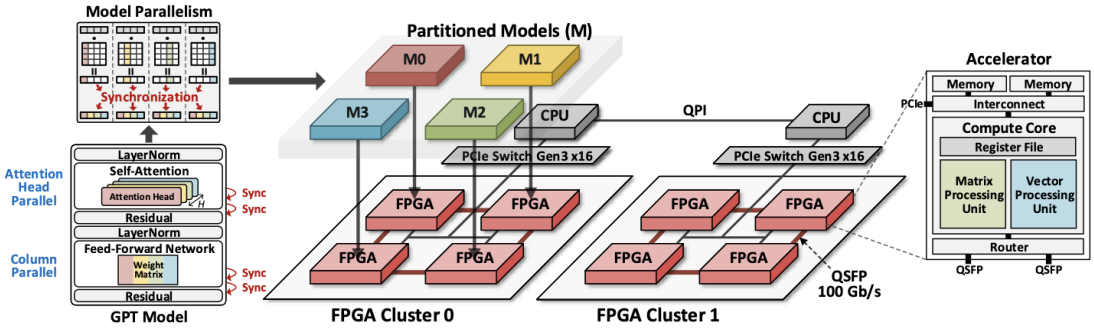


Figure 5. Overall DFX appliance architecture. Left is the illustration of GPT-2 model parallelism. Center is the mapping of the partitioned models into the DFX appliance. Right is the accelerator's microarchitecture.

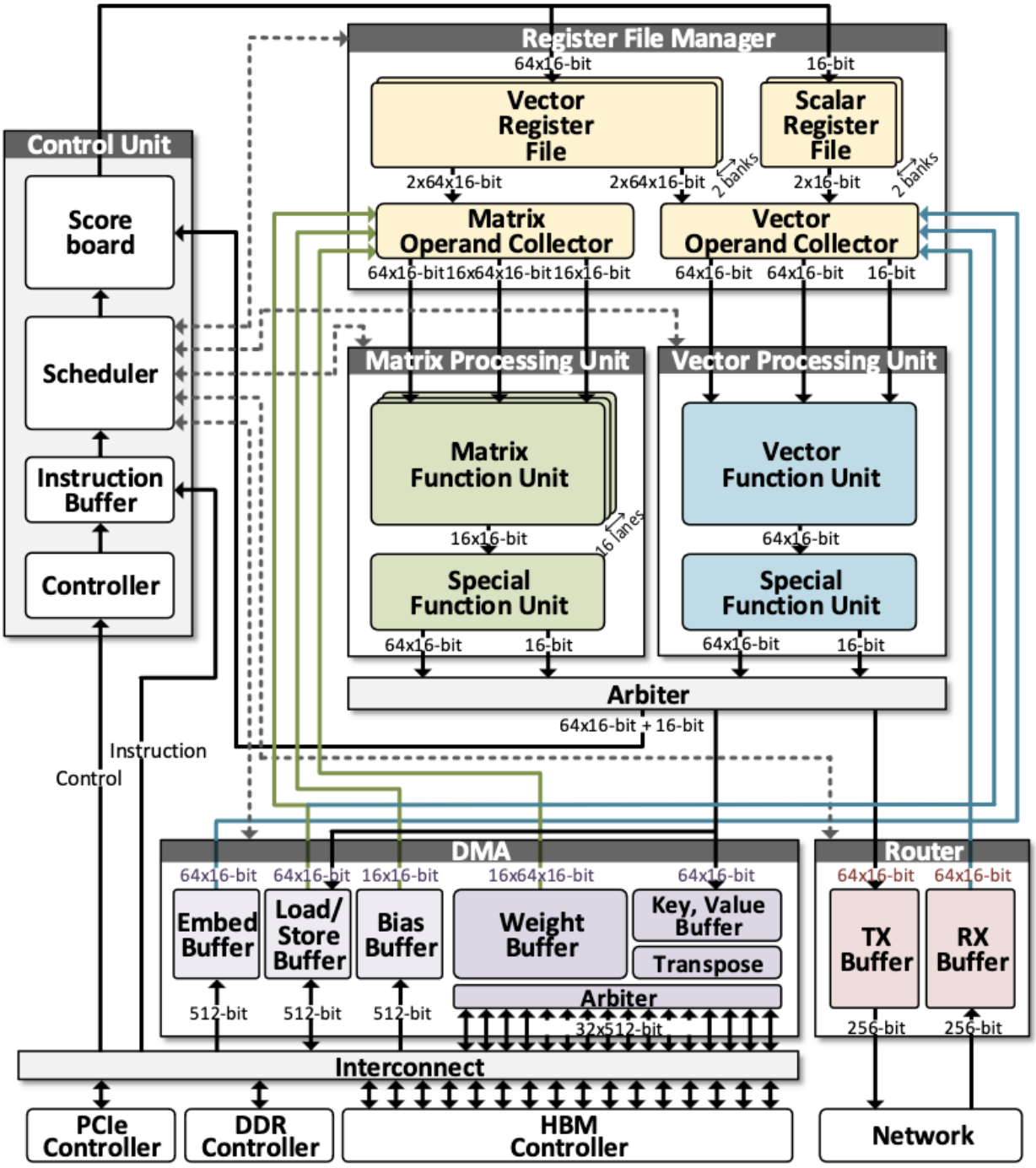


Figure 7. DFX compute core microarchitecture.

3. Skipping Redundant or Trivial Computations

3.1 ELSA-Hardware-Software Co-design for Efficient, Lightweight Self-Attention Mechanism in Neural Networks

https://taejunham.github.io/data/elsa_isca21.pdf

Objective: ELSA aims to accelerate attention operations in neural networks by preselecting keys that are likely to yield high attention scores, thereby avoiding the necessity of performing full computations.

Methodology:

- **Approximating Vector Similarity:** The core method involves approximating the similarity between vectors using the Hamming distances of their hashed versions. Hashing is a process of converting data into a fixed-size string, which makes it easier to compare and compute.
- **Combination of Techniques:** ELSA uses a combination of techniques, including hashing, cosine function approximation (a way to estimate the cosine similarity between vectors), and threshold-based filtering. This combination helps efficiently identify significant relationships or interactions in the data.
- **Reducing Computational Overhead:** By using these techniques, ELSA significantly reduces the computational overhead involved in attention mechanisms, which can be quite resource-intensive.

Performance and Impact:

- **Model Accuracy:** Despite the reduction in computational complexity, ELSA maintains the accuracy of the model.
- **Accelerator Performance:** The proposed hardware accelerator for ELSA demonstrates significant improvements in both speed and energy efficiency compared to conventional GPU implementations.
- **Application:** This method is particularly useful in scenarios where fast processing of large-scale neural networks is crucial, such as in real-time applications.

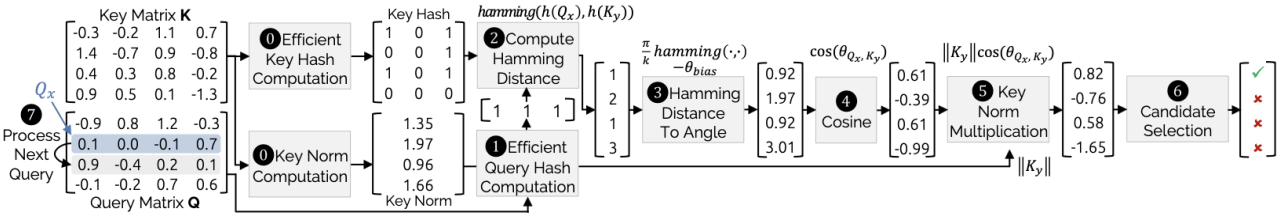


Fig. 4. Approximate Self-attention Algorithm

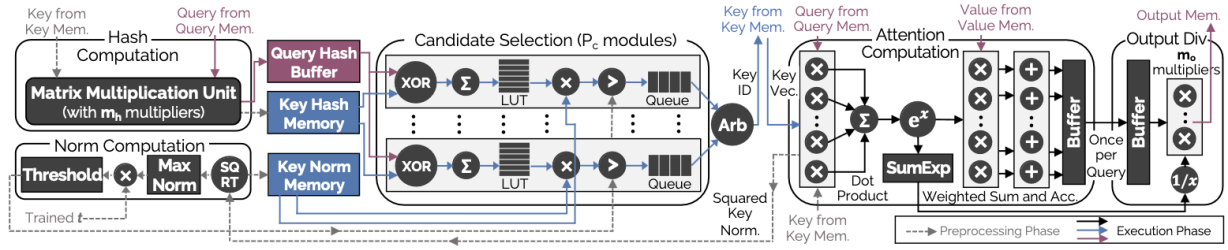


Fig. 7. ELSA Pipeline Block Diagram

3.2 DOTA

https://dl.acm.org/doi/pdf/10.1145/3503222.3507738

Objective: DOTA focuses on reducing computation wastage in the fully connected attention layers of neural networks.

Key Innovations:

- **Prediction of Attention Scores:** DOTA predicts attention scores by employing low-rank transformations (LRT) of the Q (query) and K (key) matrices, which are essential components of the attention mechanism.
- **Masking Weak Attention Connections:** By identifying weak or less significant attention connections, DOTA masks these connections to reduce unnecessary computations.
- **Dual Optimization:** A notable feature of DOTA is its optimization of both the network parameters and the LRT parameters. This dual optimization allows the model to adapt to sparse attention graphs (where only a few connections are significant) while preserving its accuracy.
- **Handling Task Complexities:** The approach facilitates efficient management of different task complexities, adapting dynamically to the requirements of the task at hand.

Performance and Impact:

- **Efficiency Gains:** DOTA results in significant gains in efficiency over traditional GPU processing, making it a valuable approach for handling complex neural network tasks.
- **Application:** It's particularly beneficial for large-scale models where attention mechanisms can become a computational bottleneck.

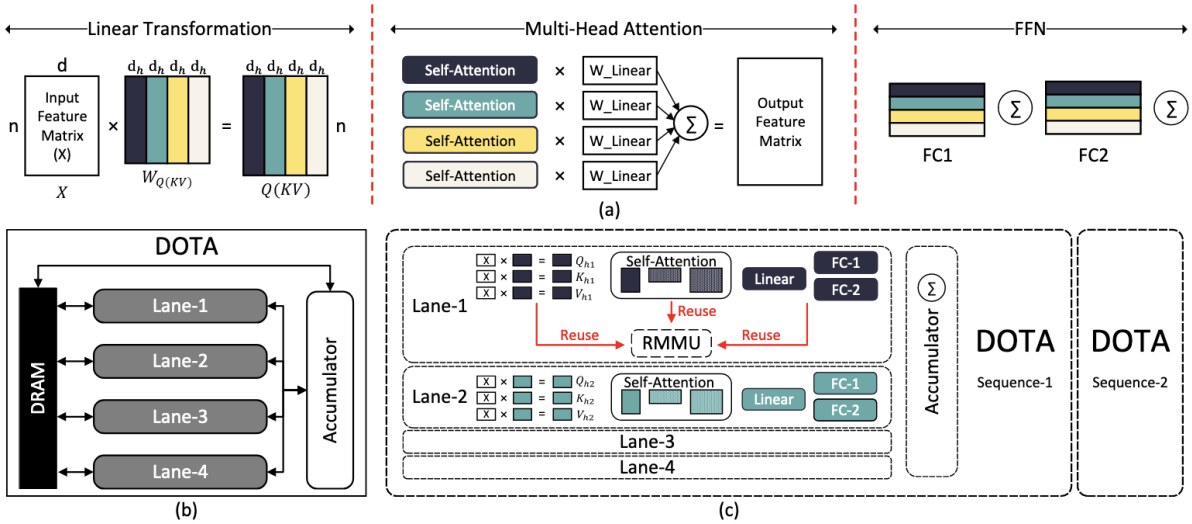


Figure 5: DOTA system design. (a) The abstraction of a single encoder block. We divide each encoder into three sequential stages. Each stage contains multiple GEMM operations that can be further cut into chunks (represented by different colors) and mapped to different compute Lanes. (b) Overall system design of DOTA. Each compute Lane communicates with off-chip DRAM for input feature. The intermediate results are summed up in the Accumulator. (c) Computation mapping between the algorithm and hardware. Each DOTA accelerator processes one input sequence, and each Lane computes for one chunk (color).

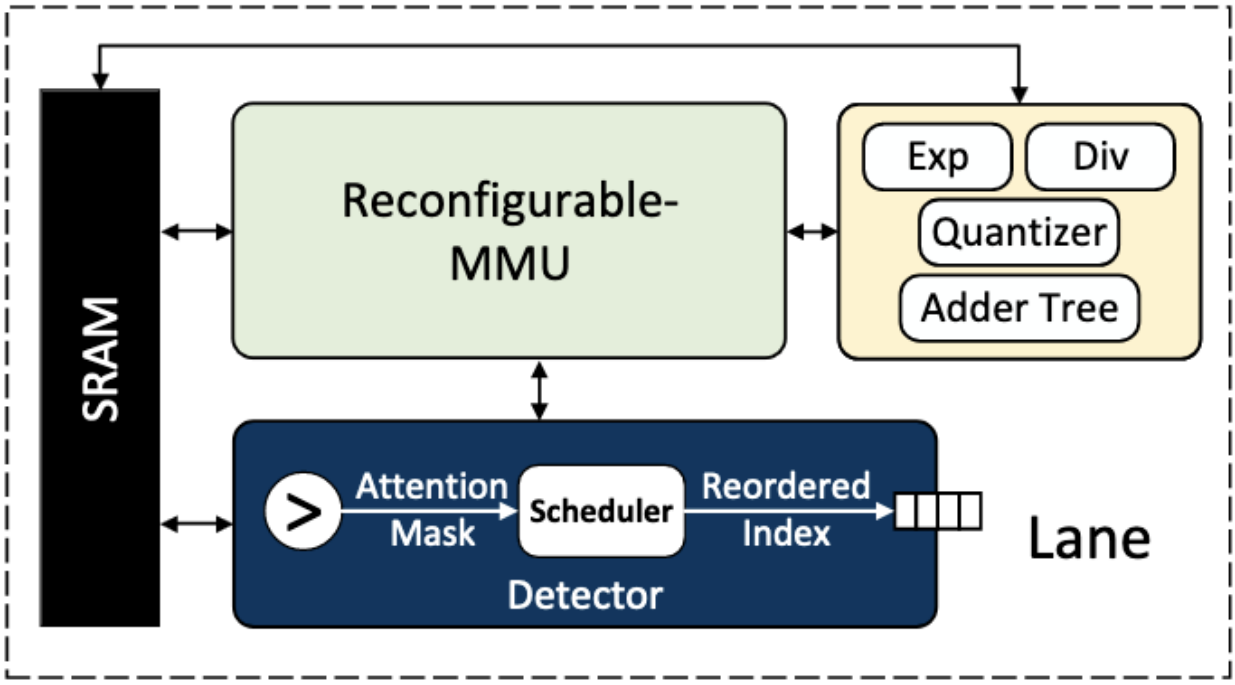


Figure 6: Architecture of each compute Lane.

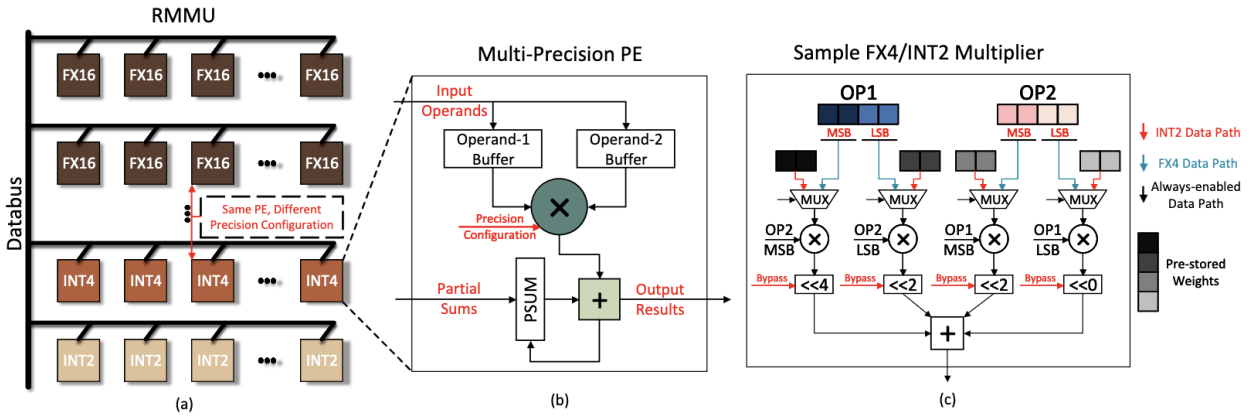


Figure 7: Design of the Reconfigurable Matrix Multiplication Unit. (a) RMMU is composed of a 2D PE array, where each row can be configured to a specific computation precision. (b) Each PE is a multi-precision MAC unit. (c) A sample FX4/IN2 multi-precision multiplier. The key is to build up high precision multiplication data path with low precision multipliers. In low precision mode, we split and multiply the input operands with pre-stored weights and perform in-multiplier accumulation. Therefore, the computation throughput is quadratically improved while input/output bit-width are kept the same as high precision mode.

3.3 A^3 (Approximate Attention Acceleration)

https://arxiv.org/pdf/2002.10941.pdf

Objective: A^3 aims to speed up attention computations in neural networks by employing approximation techniques.

Methodology:

- **Preprocessing the Key Matrix:** The first approximation scheme involves preprocessing the key matrix to estimate a 'rough-score' for each row. This process helps the system identify which relationships might be significant without performing the full computation.
- **Selective Attention Score Computation:** The second scheme computes attention scores precisely but only for selected rows. After calculating these scores, a threshold is applied to decide which scores are relevant enough to be passed to the softmax function, which is a standard part of the attention mechanism responsible for normalizing attention scores.
- **Computational Load Reduction:** By focusing only on essential parts of the attention mechanism, A^3 effectively reduces the computational load, making the process more efficient.

Performance and Impact:

- **Efficiency Improvements:** The proposed accelerator for A^3 demonstrates significant improvements in both performance and energy efficiency over traditional CPU and GPU setups.
- **Application:** This method is particularly beneficial for applications requiring efficient processing of large-scale neural networks, such as real-time systems.

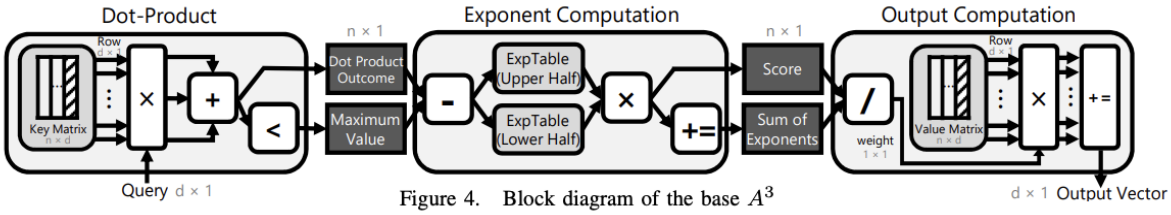


Figure 4. Block diagram of the base A^3

3.4 Dynamic ViT Inference-Enabling and Accelerating Dynamic Vision Transformer Inference for Real-Time Applications

https://arxiv.org/pdf/2212.02687.pdf

Objective: This research explores how semantic segmentation models can be optimized by selectively bypassing computations.

Key Innovations:

- **Selective Layer Skipping:** The paper demonstrates that it's possible to achieve substantial latency reductions by selectively skipping layers or adjusting the number of input/output channels in the model.
- **Minimal Impact on Accuracy:** Despite these modifications, the authors show that the impact on model accuracy is minimal.
- **Dynamic Inference Method:** They propose a dynamic inference method that adjusts the computation graph based on resource constraints. This method allows for flexible optimization of performance without necessitating extensive model retraining.

Impact and Application:

- **Resource-Constrained Environments:** This approach is particularly useful in scenarios where computational resources are limited or where there is a need to balance computational load and accuracy dynamically.
- **Flexibility in Performance Optimization:** The dynamic nature of the proposed method offers a versatile approach to enhancing model performance, making it applicable to a wide range of real-world scenarios.

3.5 DiVIT - Algorithm and architecture co-design of differential attention in vision transformer

https://dl.acm.org/doi/10.1016/j.sysarc.2022.102520
Objective: DiVIT aims to improve the efficiency of computations in the Multi-Head Attention (MHA) block of vision transformers by leveraging the locality of neighboring patches.

Methodology:

- **Delta Encoding:** The core innovation is 'delta encoding,' which focuses on the differences between adjacent patches rather than the patches themselves. This approach is based on the observation that neighboring patches in images often have similar features.
- **Reduced Non-Zero Bits Processing:** By focusing on differences, DiVIT significantly reduces the number of non-zero bits that need to be processed. This reduction is crucial for enhancing the efficiency of bit-serial multipliers, which are a type of hardware used for performing multiplication operations in neural networks.
- **Architecture Design:** The architecture of DiVIT is specifically designed to exploit patch locality, optimizing the processing of image data in vision transformers.

Performance and Impact:

- **Speed and Energy Efficiency:** The proposed architecture leads to substantial gains in speed and energy efficiency, especially when compared to traditional CPU and GPU implementations.
- **Application:** DiVIT is particularly beneficial for applications that require fast and energy-efficient processing of large volumes of image data.

3.6 AxBy-ViT Reconfigurable Approximate Computation Bypass for Vision Transformers

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9806143
Objective: AxBy-ViT addresses the inefficiency of handling trivial operations in Vision Transformers by approximating operands near zero or ±1.

Key Innovations:

- **Observation of Gaussian Distribution:** The authors note that operands in ViT often follow a Gaussian distribution, indicating a high concentration of values around zero.
- **Approximate Matching Techniques:** Two approximate matching techniques are introduced:
  - **Zero Approximation:** Adjusting the threshold for zero approximation helps to handle values close to zero more efficiently.
  - **±1 Approximation:** Truncating mantissa bits for ±1 approximation deals with values close to ±1, balancing computational reduction and accuracy loss.
- **Tailored Optimization Strategies:** Different blocks within the Vision Transformer architecture exhibit varying tolerances to these approximations. This observation leads to the development of tailored optimization strategies for each block, maximizing efficiency while minimizing the impact on accuracy.

Impact and Application:

- **Computational Reduction:** By optimizing the handling of trivial operations, AxBy-ViT achieves a balance between computational reduction and accuracy loss.
- **Diverse Applications:** This technique can be applied across various settings where Vision Transformers are used, enhancing their efficiency without significantly compromising performance.

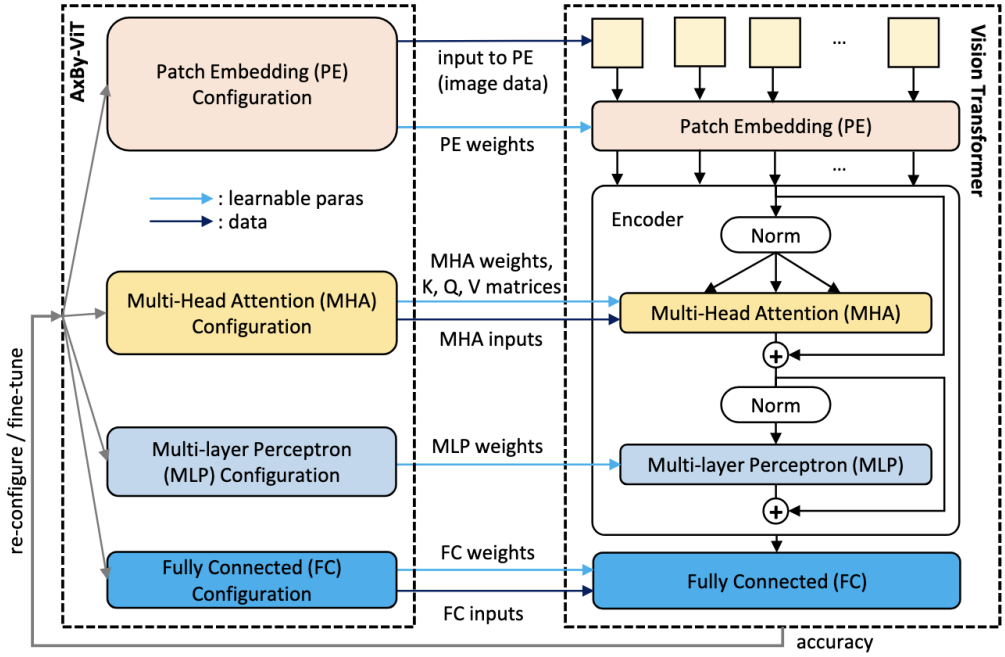


Fig. 1. Approximate trivial computation bypass targets of **AxBy-ViT** inside a vision transformer. **AxBy-ViT** focuses on four components of a vision transformer: patch embedding (PE), multi-head attention (MHA), multi-layer perceptron (MLP) and fully connected layer (FC).

4. Dataflows for Exploiting Reuse

4.1 Output Block Stationary (OBS)-An FPGA-Based Transformer Accelerator Using Output Block Stationary Dataflow for Object Recognition Applications

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9848824

Objective: OBS introduces a novel dataflow pattern for transformers, aiming to reduce memory accesses for inputs and weights.

Methodology:

- **Block Division of Matrix Multiplications:** The key strategy in OBS is to divide matrix multiplications into smaller block computations. This division enables more efficient handling of data within the computations.
- **Two-Level Broadcasting:** A significant feature of OBS is its implementation of two-level broadcasting, which substantially reduces the memory bandwidth requirements. Broadcasting refers to the method of handling data across different units of a processor.
- **Focus on Transformer Models:** The OBS pattern is particularly effective for transformer models, where the reuse of weights is less common compared to Convolutional Neural Networks (CNNs).

Performance and Impact:

- **Comparison with Output Stationary (OS) Dataflow:** While OBS shows a slight reduction in hardware utilization compared to the OS dataflow, it demonstrates substantial improvements in energy consumption and overall efficiency.

- **Application:** OBS is beneficial for applications involving transformers, especially where energy efficiency is a critical concern.

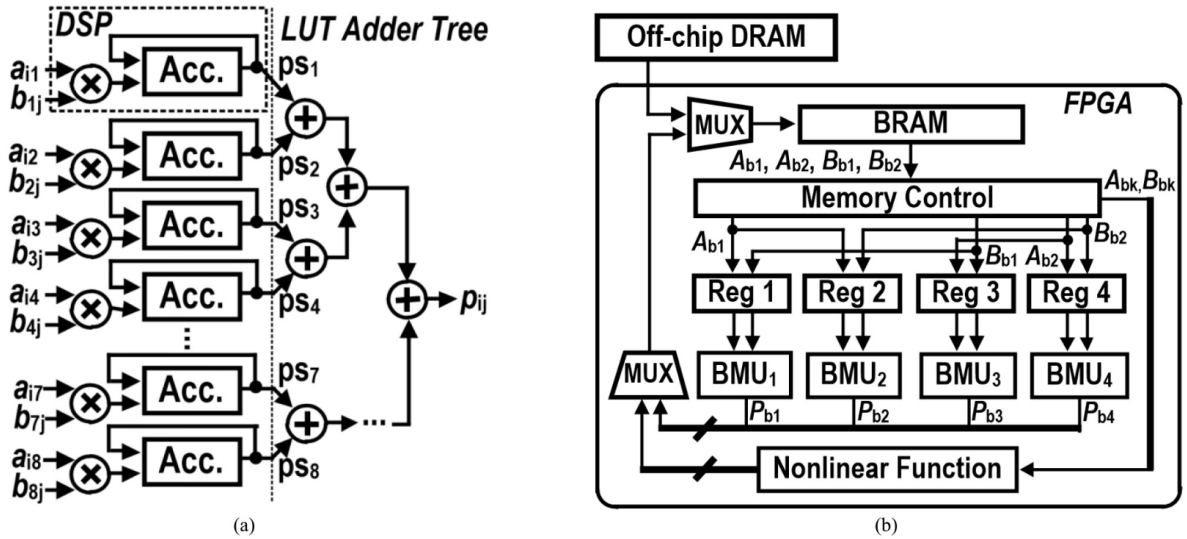


Fig. 5. The block diagram of (a) the DPE, and (b) the accelerator.

4.2 Sanger -A Co-Design Framework for Enabling Sparse Attention using Reconfigurable Architecture

https://dl.acm.org/doi/pdf/10.1145/3466752.3480125

Objective: Sanger aims to accelerate sparse attention models by optimizing the structure of attention masks and dataflow.

Key Innovations:

- **Transforming Attention Masks:** Sanger transforms unstructured attention masks into structured blocks. This restructuring leads to a more uniform distribution of computation across the processing elements (PEs) of the hardware.
- **Score-Stationary Dataflow:** A novel 'score-stationary' dataflow is introduced, which involves keeping sparse scores in PEs until all computations are completed. This strategy reduces decoding overheads and unifies two key operations in sparse attention models: Sparse-Dense-Dense Matrix Multiplication (SDDMM) and Sparse Matrix-Matrix Multiplication (SpMM).
- **Reduced Model Size without Accuracy Loss:** This approach allows for a significant reduction in the model size without compromising accuracy.

Performance and Impact:

- **Improvements Over Traditional CPUs and GPUs:** Sanger shows marked performance improvements over traditional CPU and GPU setups, making it a valuable approach for handling sparse attention models.

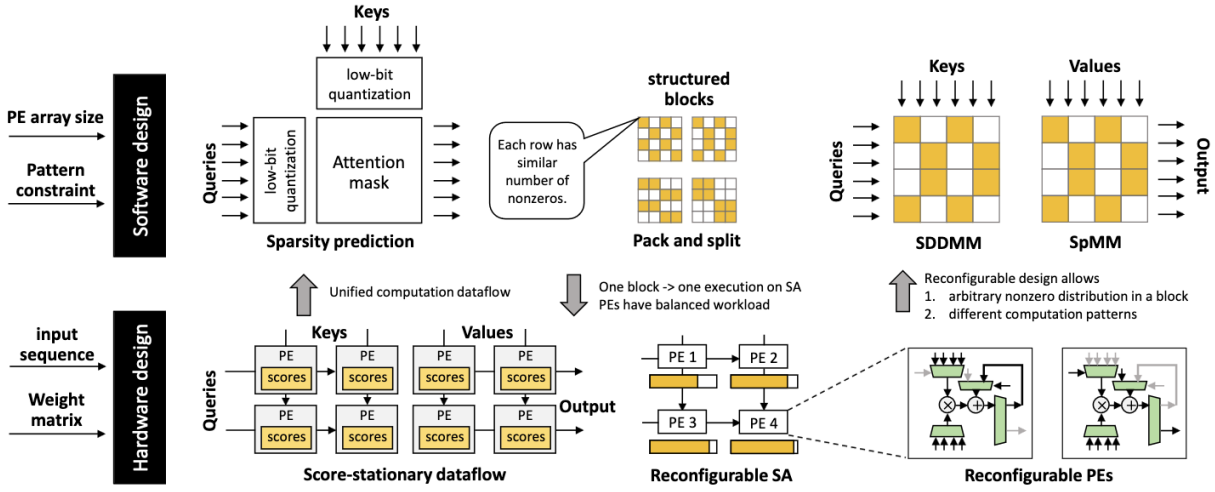


Figure 2: Sanger framework overview.

4.3 SALO -an efficient spatial accelerator enabling hybrid sparse attention mechanisms for long sequences

https://dl.acm.org/doi/10.1145/3489517.3530504
Objective: SALO focuses on addressing the computational challenges posed by long input sequences in transformers, particularly by optimizing data reuse.

Key Innovations:

- **Reorganization of Q Matrix:** SALO reorganizes the Query (Q) matrix to expose data reuse opportunities in dilated window attention, a mechanism similar to sliding window attention. This reorganization is crucial for making the computation process more efficient.
- **2D Systolic Array Architecture:** The architecture features a 2D systolic array, which is an arrangement of processors that perform matrix operations in a coordinated way. This architecture efficiently computes the softmax function, a key component in the attention mechanism.
- **Handling Global Attention:** SALO includes a global Processing Element (PE) row/column in its architecture, specifically designed to handle global attention, which is important for considering long-range dependencies in data.

Performance and Impact:

- **Speedups and Energy Savings:** The design of SALO leads to significant speedups and energy savings when compared to traditional CPUs and GPUs. This efficiency is particularly noticeable in models like ViL and Longformer, which are designed to handle long input sequences.
- **Application:** SALO is beneficial for applications involving transformers that process long sequences, where computational efficiency and energy usage are critical factors.

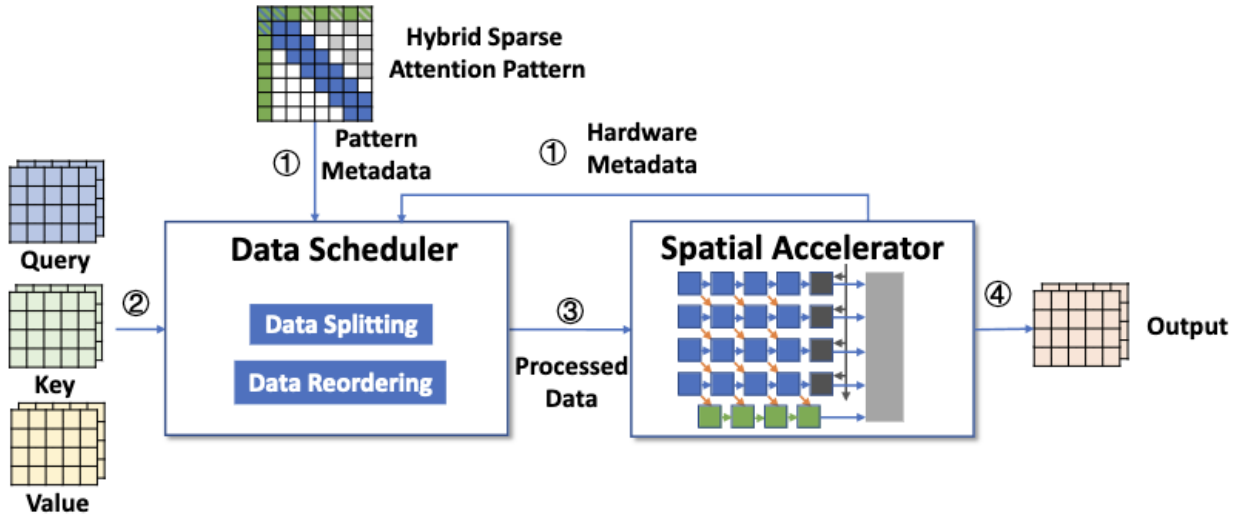


Figure 3: The framework overview of SALO

4.4 ViTCoD - Vision Transformer Acceleration via Dedicated Algorithm and Accelerator Co-Design

https://arxiv.org/pdf/2210.09573.pdf
Objective: ViTCoD aims to mitigate data movement and under-utilization issues in Vision Transformer (ViT) accelerators by implementing fixed sparse attention patterns.

Methodology:

- **Clustering Query-Key Pairs:** The method involves clustering query-key pairs into denser and sparser layouts. This clustering facilitates more efficient computation and data compression.
- **Auto-Encoder for Data Compression:** An auto-encoder is used to compress the data effectively, reducing the size of the information that needs to be processed.
- **Separate Engines for Dense and Sparse Layouts:** The accelerator design includes distinct engines for handling dense and sparse layouts. This specialization optimizes data reuse and minimizes the need for off-chip memory accesses.
- **Fixed Sparse Attention Patterns:** By enforcing these patterns, ViTCoD addresses the inefficiencies that typically arise in ViT accelerators.

Performance and Impact:

- **Performance Improvements Over CPUs and GPUs:** ViTCoD demonstrates significant performance improvements over traditional CPU and GPU implementations, especially in scenarios characterized by high attention sparsity.
- **Application:** This approach is particularly relevant for applications using Vision Transformers where attention sparsity is a major consideration.

Overall Impact: Both SALO and ViTCoD contribute significantly to enhancing the efficiency of transformer and Vision Transformer models. SALO improves data processing for long input sequences, while ViTCoD optimizes data movement and computation in sparse attention scenarios. These advancements are crucial for the deployment of large-scale neural networks in real-world applications where processing speed, data efficiency, and energy consumption are key concerns.

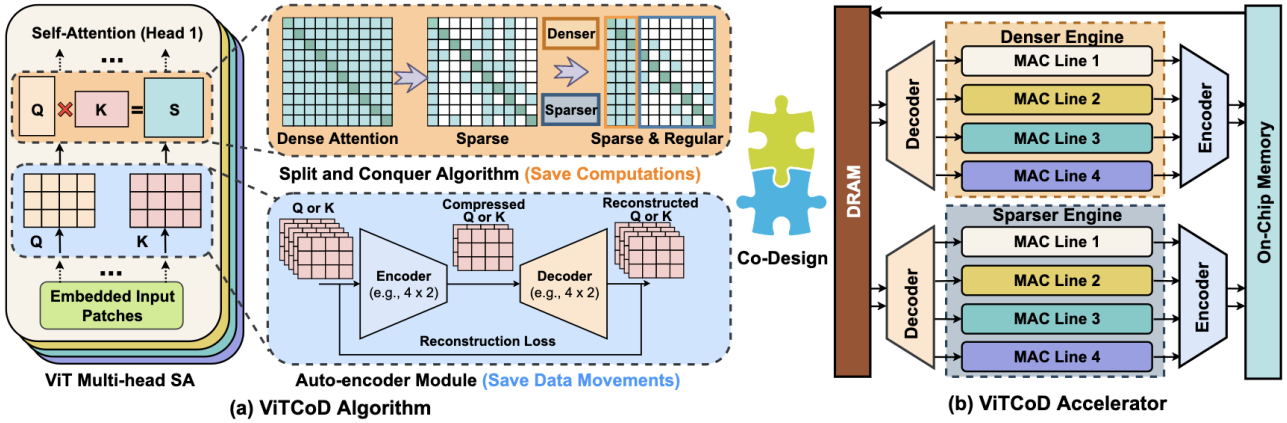


Fig. 5. An overview of ViTCoD, the first algorithm-accelerator co-design framework dedicated to sparse ViTs.

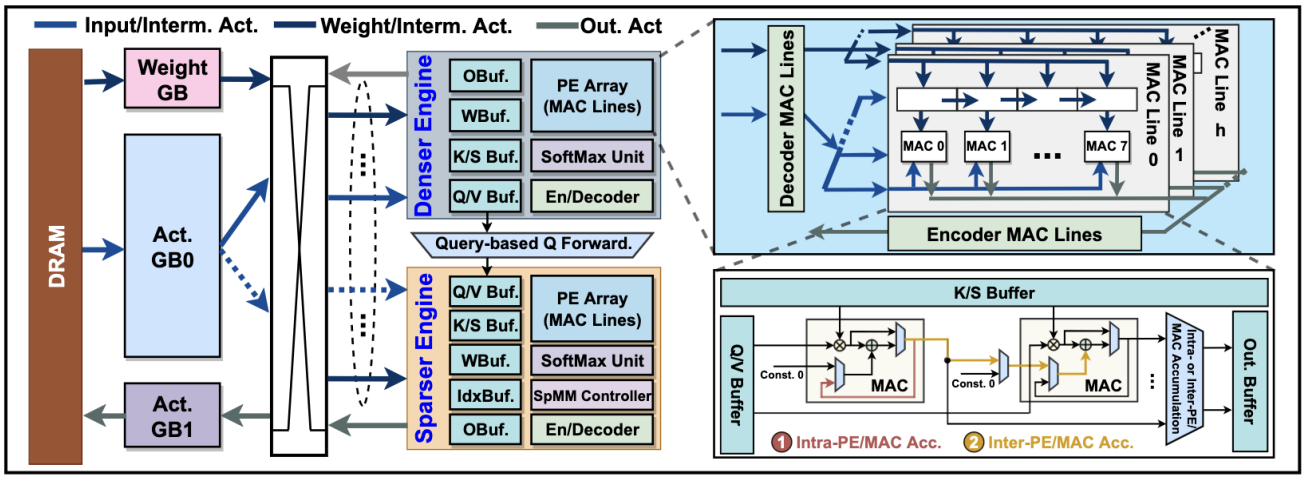


Fig. 12. Illustrating the micro-architecture of our ViTCoD accelerator.

5. **Block-Circulant Matrix for Reducing Weight Storage

5.1 Ftrans- Energy-Efficient Acceleration of Transformers using FPGA

https://arxiv.org/pdf/2007.08563.pdf

Objective: Ftrans aims to improve the compression of transformer models, making them more efficient in terms of memory usage and computational requirements, while maintaining model accuracy.

Methodology:

- **Enhanced Block-Circulant Matrix Method:** The traditional block-circulant matrix (BCM) method is used for compressing neural network models by representing weight matrices with circulant matrices. Ftrans builds upon this method by not only considering the first row/column of these matrices but also incorporating a representation of the remaining rows/columns. This approach allows for a more accurate representation of model parameters.
- **Strategic On-Chip Storage:** Ftrans strategically stores the encoder and decoder layers of the transformer model on-chip, reducing the need for off-chip communication which is often a bottleneck in terms of speed and energy efficiency. Meanwhile, the embedding layer, which typically requires more memory, is kept off-chip.
- **Specialized Processing Elements (PEs):**
  - **Matrix-Vector Multiplication:** PEs are specialized for efficient matrix-vector multiplication, a common and computationally intensive operation in transformer models.

- **FFT/IFFT Operations:** The implementation includes PEs specifically for Fast Fourier Transform (FFT) and Inverse FFT (IFFT) operations. FFT and IFFT are used in the BCM method for efficient computation of circulant matrix operations.

Performance and Impact:

- **Model Size Reduction:** Ftrans allows for a significant reduction in the model size, which is crucial for deploying transformers in memory-constrained environments.
- **Improved Performance and Energy Efficiency:** The system demonstrates improved performance and energy efficiency over traditional CPU-based setups. This improvement is particularly relevant for applications requiring real-time processing or operating under energy constraints.

Application: Ftrans is beneficial for applications that use transformer models, especially where there are limitations in terms of memory and computational resources. By compressing the model without significant loss in accuracy, Ftrans enables the deployment of sophisticated neural networks in more constrained environments, such as edge computing devices.