특집논문 (Special Paper)
방송공학회논문지 제30권 제6호, 2025년 11월 (JBE Vol.30, No.6, November 2025)
https://doi.org/10.5909/JBE.2025.30.6.989
ISSN 2287-9137 (Online) ISSN 1226-7953 (Print)

LLM 가속기의 효율적인 행렬 전치를 위한 DMA 적용 방법

이 준 호[a], 안 준 영[b], 김 정 우[a], 서 영 호[a)‡]

Efficient Matrix Transposition in LLM Accelerators via Direct Memory Access (DMA) Integration

Junho Lee[a], Junyoung An[b], Jungwoo Kim[a], and Young-Ho Seo[a)‡]

요 약

대규모 언어 모델(LLMs)의 막대한 파라미터 수와 높은 연산 복잡도는 메모리 대역폭과 하드웨어 자원의 효율적 활용을 하드웨어 가속기 설계의 핵심 과제로 부상시키고 있다. 이러한 문제를 해결하기 위해, 본 연구에서는 자원 효율적인 행렬 전치 모듈이 내장된 DMA(Direct Memory Access)를 설계하여 행렬의 이동과 전치를 동시에 처리하고, 연산자 융합 기법을 통해 중간 메모리 접근을 최소화하는 FPGA 기반 LLM 가속기 설계 방안을 제안한다. 제안하는 가속기는 ZCU104 FPGA 보드에서 구현되어 Llama3.2-1B 모델을 대상으로 성능 평가를 수행하였다. 그 결과, 선행 연구들과 비교하여 50~60% 적은 하드웨어 자원과 20~40% 적은 전력 소모량으로 유의미한 토큰 생성 속도(1.13 token/s)를 달성하였다. 이러한 결과를 통해 제한된 환경에서 자원 효율성과 저지연 특성을 동시에 만족하는 LLM 가속기 설계의 방향성을 제시한다.

Abstract

The enormous parameter count and high computational complexity of large language models (LLMs) have made the efficient utilization of memory bandwidth and hardware resources a key challenge in hardware accelerator design. To address these issues, this study proposes an FPGA-based LLM accelerator design that designs a resource-efficient Direct Memory Access (DMA) module with an integrated matrix transpose unit to simultaneously handle matrix movement and transposition. It also employs operator fusion techniques to minimize intermediate memory accesses. The proposed accelerator was implemented on a ZCU104 FPGA board and evaluated using the Llama3.2-1B model. The results achieved a significant token generation rate (1.13 tokens/s) while using 50‑60% fewer hardware resources and consuming 20‑40% less power compared to prior works. These results suggest a design direction for LLM accelerators that simultaneously satisfy resource efficiency and low-latency features in constrained environments.

Keyword : Accelerator, FPGA, LLM, LLaMA, Matrix Transposition

Ⅰ. 서 론

대규모 언어 모델은 트랜스포머[1] 아키텍처를 기반으로 자연어 처리 분야에 혁신적인 변화를 가져왔다. 그러나 대규모 언어 모델(Large Language Model, LLM)의 추론 과정은 막대한 연산량과 높은 메모리 용량 및 대역폭을 요구하므로[2], 이를 위한 전용 하드웨어 가속기 개발은 필수적이다. 하지만 기존 연구에서는 attention[1] 연산 과정에서 필연적으로 발생하는 행렬 전치 과정에서의 병목은 최적화 대상으로 고려되지 않고 있다.

본 연구에서는 이러한 기존 연구에서의 한계를 극복하고 엣지 FPGA 환경에서 대표적인 오픈소스 LLM인 LLaMA3.2-1B 모델[3]의 효율적인 추론을 구현하는 것을 목표로 한다. 이를 위해 본 논문에서는 다음과 같은 기법들을 활용한 가속기 설계를 제안한다.

(1) 다중 뱅크 기반의 행렬 전치 모듈을 설계하고 DMA (Direct Memory Access) 컨트롤러에 통합하여 데이터 전송과 동시에 실시간 전치를 수행한다. 이를 통해 행렬 전치 과정에서 발생하는 병목을 최소화하고 하드웨어 자원의 효율적 사용을 도모한다.

(2) 행렬 곱셈, RoPE(Rotary Positional Encoding), Soft-max 연산 블록들을 연산자 융합 기법을 통해 통합하여 중간 메모리 접근을 최소화하고 파이프라인 효율을 극대화한다.

Ⅱ. 관련 이론

1. Attention Mechanism

Attention 메커니즘의 핵심은 query, key, value를 통해 입력 시퀀스 내에서 중요한 정보에 선택적으로 집중하는 것이다. 연산 과정에서 생성된 key, value 벡터는 향후 반복적인 사용을 위해 KV 캐시의 형태로 메인 메모리에 저장된다. 그림 1을 보면 key 캐시의 경우, 별도의 전치 작업 없이도 query와의 행렬 곱셈 연산에 적합한 형태로 메모리에 저장되므로 효율적인 접근이 가능하다. 그러나 value 캐시는 연산 시 메인 메모리에 열 단위이 접근이 요구되므로 하드웨어 관점에서 비효율적인 메모리 접근을 초래한다[4]. 따라서 value 캐시에 대해 최적화된 행렬 전치 연산을 수행함으로써 이러한 비효율로 발생하는 병목을 개선하는 것은 중요한 고려 사항이다.

2. 행렬 전치

행렬 전치(Matrix Transposition)는 행렬의 행과 열을 교환하는 기본적인 선형대수 연산으로, 신호 처리와 인공지능 연산에서 광범위하게 활용되고 있어 다양한 하드웨어 모듈이 있다.

그 중 레지스터 배열 기반[5], 시스톨릭 배열 기반[6] 방식은 시프트 레지스터에 데이터를 순차적으로 저장하고 병렬 출력을 수행함으로써 지연 시간을 크게 단축할 수 있다. 그러나 각 열의 모든 요소에 대한 접근이 요구되기 때문에 행렬 크기와 데이터 비트 폭에 비례하는 다수의 플립플롭이 필요하다. 따라서 기존의 전치 연산 하드웨어 구조들은 지연 시간 혹은 자원 효율 측면에서 한계를 가지므로 이러한 제약을 극복하기 위한 전치 모듈의 설계가 필요하다.

Ⅲ. 제안 가속기 설계

1. 전제 가속기 구조

제안하는 LLM 가속기는 Zynq UltraScale+ MPSoC ZCU104[7] FPGA 보드를 기반으로 설계되었다. 그림 2와 같이 Processing System(PS)과 Programmable Logic(PL)을 결합한 구조를 활용하여 시스템 관리 및 제어를 위한 프로

a) 광운대학교 전자재료공학과(Department of Electronic Materials Engineering, Kwangwoon University)
b) 광운대학교 전자공학과(Department of Electronic Engineering, Kwangwoon University)
‡ Corresponding Author : 서영호(Young-Ho Seo)
E-mail: yhseo@kw.ac.kr
Tel: +82-2-940-8362
ORCID: https://orcid.org/0000-0003-1046-395X
※이 논문의 결과 중 일부는 한국방송·미디어공학회 2025년 하계학술대회에서 발표한 바 있음
※본 연구는 2025년도 중소벤처기업부의 기술개발사업[RS-2025-02315950]과 IDEC에서 EDA Tool를 지원받아 수행한 연구임
· Manuscript September 16, 2025; Revised November 4, 2025; Accepted November 4, 2025.

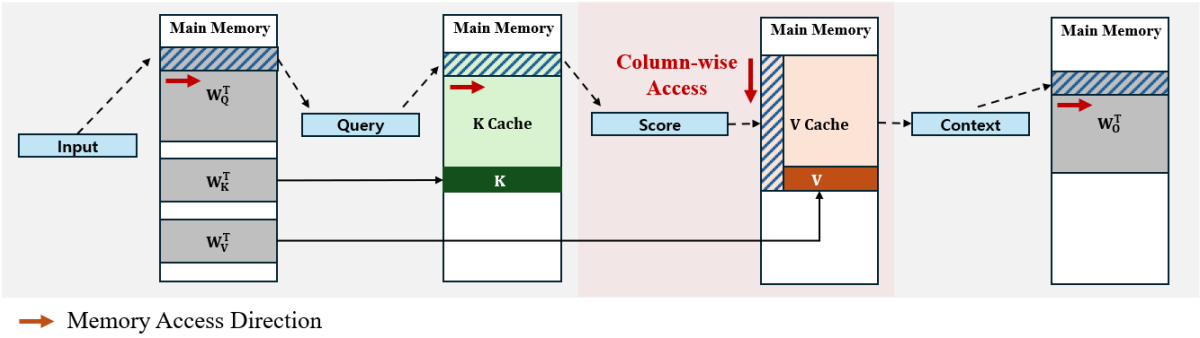

그림 1. Attention 연산 과정에서의 메모리 접근 패턴
Fig. 1. Memory access patterns during attention operations

세서와 LLM 추론을 위한 하드웨어 가속기를 유연하게 통합할 수 있도록 한다.

PS 영역에서 ARM Cortex-A53[8] 프로세서는 호스트 인터페이스 역할을 수행하며 모델 가중치 로딩, PL 영역 가속기에 명령어 전송 및 상태 관리 등의 제어 작업을 담당한다. 또한 토큰화/디토큰화, RMS 정규화 등 PL 영역에서 수행하지 않는 동작을 수행한다.

PL 영역에는 그림 2와 같이 LLM 연산을 가속하는 네 가지 주요 모듈(Controller, DMA, Compute Core, Accumulator)로 구성된다. 내부의 버퍼는 연산 중간값을 저장하거나 행렬-벡터 곱셈 연산에서 반복적으로 사용되는 벡터를 저장한다.

1.1 Controller

Controller는 FPGA의 PS 영역으로부터 연산 유형, 행렬 크기, 현재 토큰의 위치와 같은 정보가 담긴 명령어를 수신하여 DMA와 Compute Core의 동작을 제어한다. 또한, Compute Core로부터 전달되는 valid 신호를 카운트하여 지정된 개수의 결과값이 올바르게 출력되었는지를 판단하거나 DMA 컨트롤러의 상태 레지스터를 모니터링함으로써 명령어에 따른 동작 수행을 확인한다.

1.2 행렬 전치 지원 DMA

데이터 이동 과정에서 실시간 행렬 전치를 위해 전치 모듈이 통합된 DMA 컨트롤러를 사용한다. 이 DMA는 PS

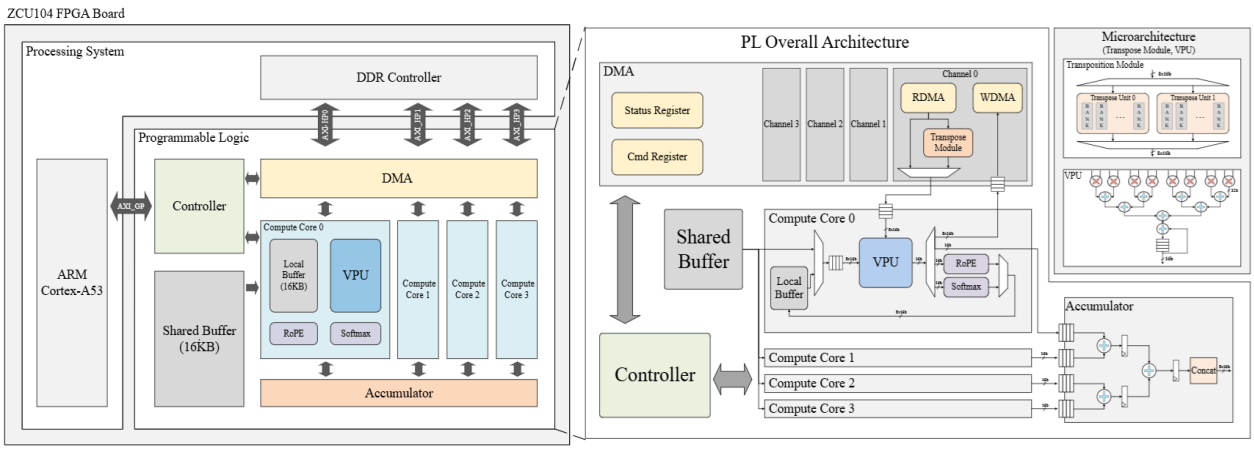

그림 2. 전체 시스템 구조도
Fig. 2. Overall system architecture

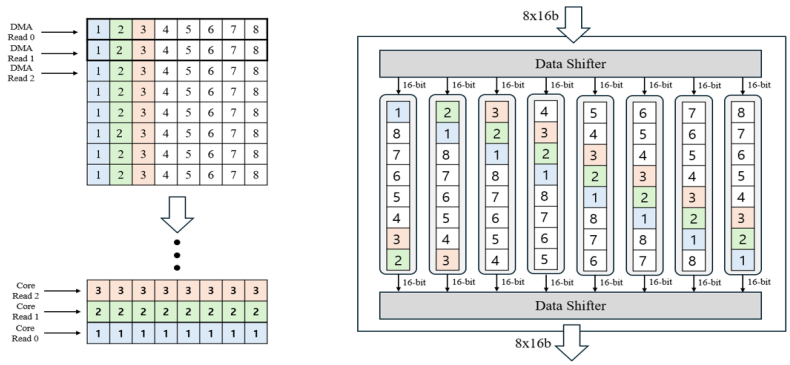

그림 3. 행렬 전치 모듈 동작 원리
Fig. 3. Matrix transpose module operation principle

영역과 PL 영역을 연결하는 4개의 AXI_HP 채널을 통해 각 Compute Core에 필요한 데이터를 전송한다.

행렬 전치 모듈은 기존 전치 하드웨어의 문제들을 해결하기 위해 그림 3과 같은 다중 뱅크 구조[9]를 가지며, 8x8 크기의 단위 행렬을 처리하기 위해 깊이 8을 가지는 8개의 독립적인 메모리 뱅크와 입력 데이터를 적절한 순서로 재정렬하여 저장, 출력하기 위한 Data Shifter로 구성된다. 이러한 구조를 통해, 동일한 열에 속한 데이터가 서로 다른 뱅크에 분산 저장되도록 만들어 전치 행렬 데이터의 병렬 출력이 가능하다. 또한, 제한된 포트 수를 갖는 메모리 블록 동작 특성을 만족하므로 FPGA 내 메모리 매크로(BRAM, Distributed RAM 등) 합성이 가능하다.

1.3 VPU (Vector Processing Unit)

VPU는 지역 버퍼 또는 공유 버퍼에 저장되어 있는 데이터와 DMA로부터 전송되는 가중치, KV 캐시 데이터 간의 행렬-벡터 곱셈 연산을 수행한다. 해당 유닛은 128-bit 채널 버스 너비에 맞춰 16개의 16-bit 데이터를 처리하기 위한 8개의 곱셈기, 이를 합산하는 덧셈기 트리로 구성된다.

2. 데이터 흐름 및 연산 스케줄링

본 가속기에서 attention 연산은 그림 4 (a)에 제시된 순서로 진행된다. 각 헤드 단위로 독립적으로 연산 가능하다는 특성을 활용하여 각 Compute Core가 8개의 attention 헤드

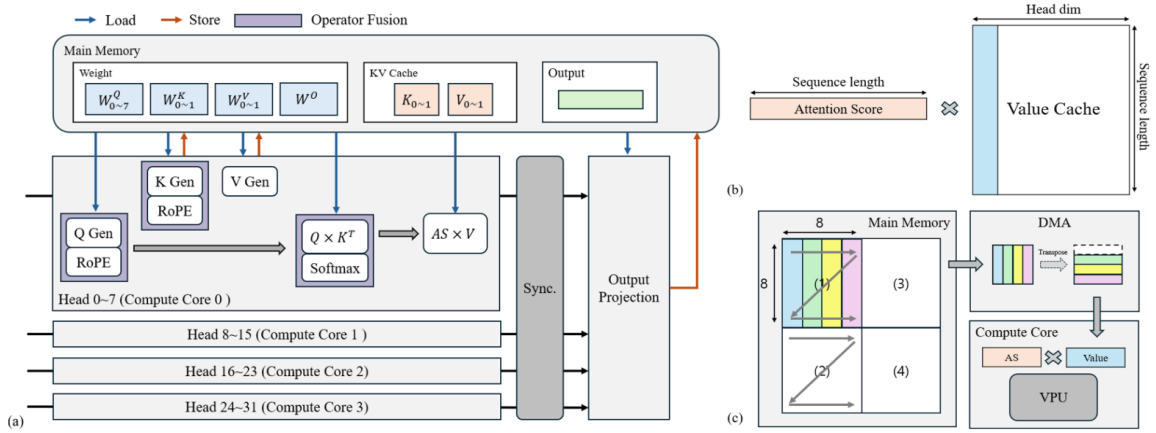

그림 4. (a) Attention 레이어 연산 흐름 (b) Attention score와 value 캐시 사이의 행렬-벡터 곱셈 연산 (c) 행렬 전치 모듈을 통한 연산 과정
Fig. 4. (a) Attention layer computation flow (b) GEMV operation between attention score and value cache (c) Working process through matrix transpose module

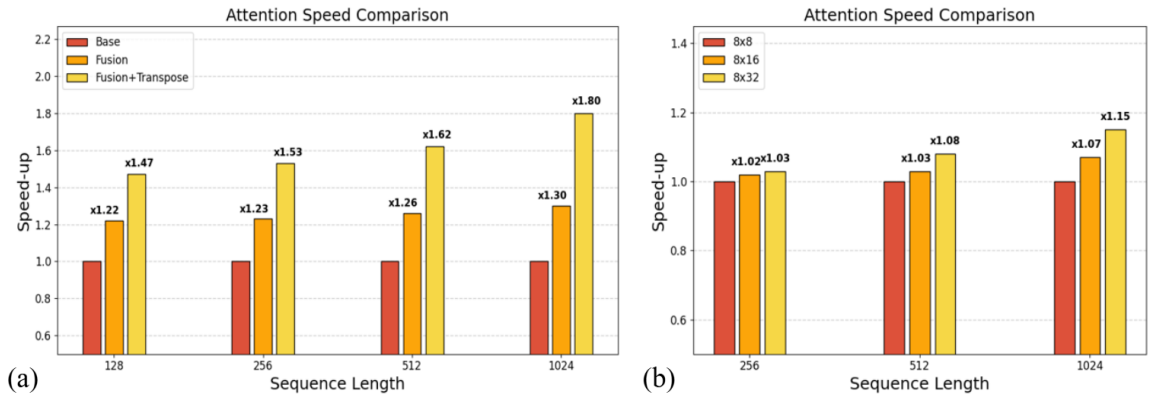

그림 5. (a) 제안 기법 도입 (b) 전치 모듈 크기에 따른 attention 레이어 성능
Fig. 5. Attention layer performance according to (a) the proposed technique and (b) the size of the transposition module

를 병렬적으로 처리한다. 이후 코어 간 동기화와 output projection 단계를 거쳐 결과를 출력한다.

Attention 연산 중 attention score와 value 캐시 간의 행렬-벡터 곱셈 연산에서 전치 모듈이 통합된 DMA를 이용하여 데이터 전송과 행렬 전치를 동시에 처리하며 수행된다. 이 과정은 그림 4 (b), (c)에서 확인할 수 있다. 이를 통해, value 행렬의 전치 과정에서 발생하는 데이터 전송 병목을 완화하고 더블 버퍼링으로 행렬 전치 지연 시간을 은닉한다.

또한, 연산자 융합 기법을 적용하여 메모리 대역폭 병목을 완화하고 데이터 재사용성을 극대화한다. 연산자 융합이란 순차적인 연산들을 연속적인 파이프라인으로 통합하여 중간 결과를 버퍼나 메모리에 저장하는 과정을 제거한다. 이를 통해 attention 레이어 내 query, key 생성과 RoPE, query-key 행렬 곱과 softmax 연산을 융합하여 중간 결과의 메모리 접근을 완전히 제거하였다.

이러한 핵심 최적화 기법들은 비동기 3단계 파이프라인을 기반으로 동작한다. 단계별로 데이터 로드, 연산, 데이터 저장으로 구성되며, Compute Core가 현재 연산을 수행하는 동안 Read DMA는 다음 데이터를 미리 로드하고 Write DMA는 이전 결과를 저장한다. 이를 통해, 메모리 접근으로 인한 Compute Core의 유휴 사이클을 최소화하고 하드웨어 자원 활용률을 극대화한다.

Ⅳ. 실험 결과

1. 실험 환경

본 가속기는 VerilogHDL[10]로 설계되었으며 Xilinx Vivado 2022.2[11]를 사용하여 Zynq UltraScale+ MPSoC ZCU104 FPGA 보드에 구현하였다. 가속기는 125MHz로 동작하며 합성된 회로와 동작 활성도로 추정한 Vivado 전력 리포트에서 4.06W 전력 소모가 보고되었다.

평가에 사용된 LLaMA3.2-1B 모델은 엣지 디바이스 환

표 1. 제안 방식과 선행 연구와의 전치 모듈 자원 사용량 비교
Table 1. Comparison of resource utilization between the proposed method and prior research's transposition module

	Resource				Latency
	LUT	FF(Flip-Flop)	F7 MUX	Dist. RAM	
Register array[5]	684	2330	256	-	8
Systolic array[6]	2072	2057	0	-	8
Proposed (FF Synth.)	1082	2185	130	-	8
Proposed (RAM Synth.)	611	123	0	160	

경을 위해 개발된 경량 언어 모델이다. 본 연구에서는 [12]의 실험 결과를 바탕으로 MMLU(Massive Multitask Language Understanding)[13] 성능 저하를 최소화할 수 있는 16비트 고정소수점 표현(Q7.9 형식)을 채택하였다. 다만, 넓은 범위의 수 표현을 요구하는 RMS 정규화 연산의 특성을 반영하여 해당 가중치는 양자화하지 않고 PS 영역에서 연산을 수행하였다.

2. 제안 기법의 효용성 분석

본 연구에서는 생성형 언어 모델의 효율적인 추론을 위해 실시간 행렬 전치를 수행하는 다중 뱅크 기반 전치 모듈 도입과 연산자 융합 기법을 제안했다. 표 1에 나타난 바와 같이, 다중 뱅크 기반 전치 모듈을 FPGA의 메모리 매크로로 합성(RAM Synth.)한 결과, 기존 방식(FF Synth.) 대비 시간 손실 없이 90% 이상의 플립플롭 자원을 절감할 수 있었다.

그림 5 (a)는 연산자 융합과 전치 모듈의 적용이 attention 연산의 성능에 미치는 영향을 보이고 있다. 두 기법은 모두 생성 토큰의 시퀀스 길이가 증가할수록 그 효과가 두드러짐을 확인할 수 있으며 평균적으로 각각 약 25%, 27%의 연산 속도 향상이 나타났다. 이를 통해 두 제안 기법 모두 attention 연산 성능 향상에 기여함을 확인할 수 있다.

그림 5 (b)는 전치 모듈의 단위 행렬 크기에 따른 attention 연산 성능 및 하드웨어 자원 사용량을 보이고 있다. 단위 행렬의 열 길이를 증가시킴에 따라 시퀀스 길이에 따른 성능의 소폭 향상을 확인할 수 있었다. 그러나 이에 비례하여 전치 모듈의 버퍼 크기, Compute Core의 VPU에서 부분합 저장을 위해 요구되는 버퍼 크기가 선형적으로 증가하였다. 따라서 본 연구에서는 자원 효율성을 고려하여 8x8 단위 행렬 크기를 채택하였다.

3. 선행 연구와의 비교

표 2는 제안하는 가속기(Ours)의 성능을 FPGA 기반 선행 연구와 비교한 결과이다. 표의 주요 항목으로, Device는 가속기가 구현된 FPGA 보드를, Resource는 하드웨어 자원 사용량을 나타낸다. Quant는 가중치 양자화 비트 수를, Token/s는 초당 토큰 생성을 나타내는 처리량 지표이다. 임베디드 디바이스 환경에서 LLaMA3-1B 모델의 하드웨어 가속을 다룬 선행 연구를 확인할 수 없어, 불가피하게 모델 규모가 유사한 연구(TinyLLaMA, BitNet)와의 결과를 비교하였다.

표 2는 임베디드 FPGA 기반 생성형 언어 모델 가속기를 제안한 선행 연구들과 본 연구의 각종 성능 지표를 비교한 결과이다. 본 연구에서 제안한 가속기는 W8 정밀도를 사용하는 LlamaF[9]와 비교할 때, 더 높은 정밀도로 연산을 수행함에도 불구하고 플립플롭은 약 3.2배, LUT는 약 4배 적은 하드웨어 자원을 사용한다. 전력 소모 또한 4.06W로 약 20% 낮다. W4 정밀도를 사용하는 SECDA[8]와 비교해서는 약 1.95배 빠른 처리 속도를 보여준다. 이는 본 논문이 제안하는 아키텍처가 높은 연산 정밀도(W16)를 유지하면서도 효율적인 하드웨어 자원 사용과 낮은 전력 소모로 경쟁력 있는 추론 속도를 달성할 수 있음을 보여준다.

한편, token/s 지표는 일부 연구를 제외하면 비교적 낮은 수준을 보였다. 이는 본 연구에서 W16 양자화를 사용한 것이 원인임으로, 향후 더 고도화된 양자화 기법 적용을 통한 성능 개선의 잠재력이 충분함을 시사한다.

표 2. 선행 연구들과의 성능 비교
Table 2. Performance comparison with prior research

	Device	Resource				Power	Task	Quant.	Token/s
		LUT	FF	BRAM	DSP				
SECDA[14]	PYNQ -Z1	-	-	-	-	-	TinyLLaMA	W4	0.58
LlamaF[15]	ZCU102	164k	171k	223	528	5.08	TinyLLaMA	W8	1.33
TeLLMe[16]	KV260	109k	156k	206	356	6.72	Bitnet	W1.58	9.1
Ours	ZCU104	41k	53k	128	64	4.06	LLaMA3-1B	W16	1.13

V. 결 론

본 연구는 하드웨어 자원이 제한적인 FPGA 보드 환경에서 1B 파라미터 규모의 대규모 언어 모델 추론을 효율적으로 수행하기 위한 가속기 구조를 제안하고 그 유효성을 검증하였다. 핵심적으로, 실시간 행렬 전치를 지원하는 DMA 컨트롤러를 설계하여 동적으로 생성되는 KV 캐시의 비효율적인 메모리 접근을 완화하고 다중 뱅크 기반 전치 모듈 설계로 높은 자원 효율성을 달성하였다. 또한, attention 연산 과정의 주요 연산들을 연산자 융합 기법으로 통합하여 중간 메모리 접근을 최소화하고 파이프라인 효율을 극대화하였다.

실험 결과, 제안하는 가속기는 선행 연구들과 비교했을 때, 30~60% 적은 하드웨어 자원과 20~40% 적은 전력 소모량을 확인할 수 있었다. 초당 토큰 생성 시간에 경우, 1.13 token/s의 비교적 낮은 수치를 보이고 있지만 양자화 방식을 통해 충분히 개선될 여지가 있다.

본 연구는 효율적인 행렬 전치를 통한 메모리 병목 최소화가 엣지 FPGA 환경에서 구현된 LLM 가속기에 효과적으로 적용될 수 있음을 시사하며, 제한된 환경에서 자원 효율성과 저지연 특성을 동시에 만족하는 LLM 가속기 설계의 방향성을 제시한다.

참 고 문 헌 (References)

[1] A. Vaswani, et al., “Attention is all you need,” Proceeding of Advances in Neural Information Processing Systems (NeurIPS), Vol.30, 2017.
doi: https://doi.org/10.48550/arXiv.1706.03762

[2] Q. Pan, H. Cao, Y. Zhu, J. Liu, and B. Li, “Contextual Client Selection for Efficient Federated Learning Over Edge Devices,” IEEE Transactions on Mobile Computing, Vol.23, No.6, pp.6538–6548, June 2024.
doi: https://doi.org/10.1109/TMC.2023.3323645

[3] A. Grattafiori, et al., “The llama 3 herd of models,” arXiv preprint arXiv:2407.21783, 2024.
doi: https://doi.org/10.48550/arXiv.2407.21783

[4] Hong, Seongmin, et al. “Dfx: A low-latency multi-fpga appliance for accelerating transformer-based text generation.” 2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO). Chicago, USA, pp. 1–13, 2022.
doi: https://doi.org/10.1109/MICRO56248.2022.00051

[5] Z. Zhou, and Z. Pan, “Effective hardware accelerator for 2d dct/idct using improved loeffler architecture,” IEEE Access, Vol.10, pp.11011-11020, 2022.
doi: https://doi.org/10.1109/ACCESS.2022.3146162

[6] H. Genc, et al., “Gemmini: Enabling systematic deep learning architecture evaluation via full-stack integration,” Proceeding of 2021 58th ACM/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, pp.1293-1298, 2021.
doi: https://doi.org/10.1109/DAC18074.2021.9586216

[7] Xilinx, Inc., Zynq UltraScale+ MPSoC ZCU104 Evaluation Kit, Xilinx, Inc., 2021.

[8] Arm Ltd., ARM Cortex-A53 MPCore Processor Technical Reference Manual, ARM DDI 0500F, 2018.

[9] S. Ma, et al., “MT-DMA: A DMA controller supporting efficient matrix transposition for digital signal processing,” IEEE Access, Vol.7, pp.5808-5818, 2018.
doi: https://doi.org/10.1109/ACCESS.2018.2889558

[10] IEEE, IEEE Standard for Verilog Hardware Description Language, IEEE Std 1364-2005, 2006.

[11] Xilinx, Inc., Vivado Design Suite User Guide: Design Flows Overview (UG973), Xilinx, Inc., 2022.

[12] J. Kim, S. Yoon, et al., “Fixed-Point Arithmetic Analysis for Development of LLaMA 3 On-Device Accelerator,” Journal of Broadcast Engineering, Vol.29, No.4, pp.498-509, July 2024.
doi: https://doi.org/10.5909/JBE.2024.29.4.498

[13] D. Hendrycks, et al., “Measuring Massive Multitask Language Understanding,” arXiv preprint arXiv:2009.03300, 2020.
doi: https://doi.org/10.48550/arXiv.2009.03300

[14] J. Haris, et al., “Designing efficient llm accelerators for edge devices,” arXiv preprint arXiv:2408.00462, 2024.
doi: https://doi.org/10.48550/arXiv.2408.00462

[15] H. Xu, Y. Li, and S. Ji, “Llamaf: An efficient llama2 architecture accelerator on embedded fpgas,” Proceeding of 2024 IEEE 10th World Forum on Internet of Things (WF-IoT), Yokohama, Japan, pp.1-6, 2024.
doi: https://doi.org/10.1109/WF-IoT62078.2024.10811385

[16] Y. Qiao, et al., “TeLLMe: An Energy-Efficient Ternary LLM Accelerator for Prefilling and Decoding on Edge FPGAs,” arXiv preprint arXiv:2504.16266, 2025.
doi: https://doi.org/10.48550/arXiv.2504.16266

저 자 소 개

이 준 호

- 2020년 3월 ~ 현재 : 광운대학교 전자재료공학과(학사과정)
- ORCID : https://orcid.org/0009-0005-3155-5257
- 주관심분야 : 하드웨어 설계, 딥러닝

안 준 영

- 2020년 3월 ~ 현재 : 광운대학교 전자공학과(학사과정)
- ORCID : https://orcid.org/0009-0008-5063-8615
- 주관심분야 : 하드웨어 설계, 딥러닝

김 정 우

- 2024년 2월 : 광운대학교 전자재료공학과 졸업(공학사)
- 2024년 3월 ~ 현재 : 광운대학교 전자재료공학과 일반대학원(석사과정)
- ORCID : https://orcid.org/0009-0003-0913-0709
- 주관심분야 : 하드웨어 설계, 딥러닝, 3D 영상 처리

서 영 호

- 1999년 2월 : 광운대학교 전자재료공학과 졸업(공학사)
- 2001년 2월 : 광운대학교 일반대학원 졸업(공학석사)
- 2004년 8월 : 광운대학교 일반대학원 졸업(공학박사)
- 2005년 9월 ~ 2008년 2월 : 한성대학교 조교수
- 2008년 3월 ~ 현재 : 광운대학교 전자재료공학과 교수
- ORCID : https://orcid.org/0000-0003-1046-395X
- 주관심분야 : 실감미디어, 2D/3D 영상 신호처리, SoC 설계, 디지털 홀로그램