종단 간 트랜스포머 인코더 가속기의 FPGA 구현

노수민, 정서호, 안주혁, 허지현, 정기석*

한양대학교

{smrho, iona97, juhuyk123, jhheo, kchung}@hanyang.ac.kr

An FPGA Implementation of End-to-End Transformer Encoder Accelerator

Soo-Min Rho, Seo-Ho Chung, Ju-Hyuk Ahn, Ji-Hyeon Heo, Ki-Seok Chung*

Hanyang University, Seoul, Korea

요 약

본 논문은 트랜스포머 모델에서 사용하는 행렬곱셈 및 모든 Non-lilnear function들을 모두 하드웨어로 구현하여 소프트웨어의 추가 처리 없이 트랜스포머 인코더를 종단 간 (End-to-End) 처리하는 가속기 구조를 제안한다. 또한 Off-chip 및 On-chip 메모리 접근 횟수를 줄이기 위한 Always-on-chip token 기법 및 Non-linear function fusing 기법을 제안한다. 제안하는 가속기는 Verilog HDL로 구현되어 Vivado 및 Vitis 툴을 통해 ALVEO U200 FPGA에서 구현되었으며, PyTorch 프레임워크 및 Huggingface 패키지를 통하여 BERT-Base 모델의 처리 시간, 처리량을 평가하였다. 제안하는 가속기는 intel사의 i9-7900X CPU 및 Nvidia사의 Jetson TX2 GPU를 비교군으로 사용하였으며, 비교군 대비 최대 2.24배 높은 처리량을 보였다.

Ⅰ. 서론

2017년 처음 등장한 트랜스포머 (Transformer)는 토큰 간 유사도를 파악하는 attention 매커니즘을 이용하여 텍스트 분류, 기계 번역, 문장 생성 등 자연어 처리 분야에서 강력한 성능을 보인다 [1]. 뿐만 아니라, 이미지 처리 혹은 이미지 생성 등 다양한 도메인에서도 높은 활용률을 보인다. 그러나 기존의 트랜스포머 가속기 연구는 트랜스포머의 Attention 등 특정 기능 구현에만 초점을 맞추어 진행되었기 때문에, 트랜스포머 인코더의 추론을 위해서 프로세서에서 실행하는 소프트웨어의 처리가 필수적이며, 이는 Off-chip으로의 데이터 오프로딩을 유발한다.

본 논문에서는 트랜스포머 인코더의 종단 간 (End-to-End) 가속을 위해 트랜스포머의 모든 연산을 하드웨어로 구현한 FPGA 기반 가속기를 제안한다. 제안하는 가속기는 1) 트랜스포머 구조의 모든 연산을 하드웨어로 구현하였으며 2) Always-on-chip token 기법 및 3) Non-linear function fusing 기법을 통해 메모리 접근 횟수를 줄여 전력 소모 대비 높은 추론 성능을 보인다. 제안하는 가속기는 ALVEO U200 FPGA 장치 구현 하였으며, intel사의 i9-7900x CPU 및 Nvidia사의 Jetson TX2 GPU와 비교하여 대비 최대 2.24배 높은 처리량을 달성하였다.

Ⅱ. 본론

1. 트랜스포머 모델 구조

트랜스포머 모델은 크게 1) MHA (Multi-Head-Attention) 및 2) FFN (Feed-Forward-Network), 3) Add & Layernorm으로 구성된다. MHA는 Q (Query), K (Key), V (Value) 행렬의 self-attention 연산을 통해 입력 토큰 간 유사도를 계산한다. MHA는 수식 (1)처럼 입력 Token에 Linear layer를 적용하여 Q, K, V를 생성하고, 수식 (2)처럼 Q, K, V 행렬을 여러 head로 분할하여 self-attention을 수행하는 과정을 수행하며, 최종적으로 수식 (3)과 같이 head 별 attention context를 병합하고 Linear layer를 통해 최종 attention 출력을 생성한다. FFN의 경우 수식 (4)와 같이 2개의 Linear layer 사이에 GELU를 적용한 형태이며, Add & Layernorm의 경우 두 Token 행렬을 더한 후 수식 (5)와 같이 Layernorm을 수행한다.

따라서 트랜스포머 모델의 종단 간 추론을 지원하는 가속기는 MHA 및 FFN를 위한 1) Linear layer, 2) 행렬곱셈, 3) Softmax, 4) GELU 연산을 지원해야 하며, 뿐만 아니라 5) Add & Layernorm 연산도 지원하여야 한다.

$$Q = Q\_Linear(x), K = K\_Linear(x), V = V\_Linear(x) \quad (1)$$

$$head_i = self\_atten(Q_i, K_i, V_i),\ self\_atten = softmax(\frac{QK^T}{\sqrt{d_K}})V \quad (2)$$

$$MHA = O\_Linear(concat(head_1, \dots, head_h)) \quad (3)$$

$$FFN = Linear(GELU(Linear(x))) \quad (4)$$

$$LN = (\frac{x - mean(x))}{\sqrt{var(x) + \epsilon}})\omega + \beta \quad (5)$$

2. 가속기 구조

제안하는 가속기의 구조는 그림 1과 같다. 가속기는 Host CPU로부터 AXI4-Lite 인터페이스를 통해 하드웨어 구성 정보 및 추론 시작 신호를 전달 받는다. 추론 시작 신호 발생 시, IP Controller는 Command memory로부터 모델의 추론을 위한 명령어를 fetch & decode하며 가속기의 동작을 제어할 수 있다. 또한 가속기는 내부 DMA를 통해 외부 메모리에 존재하는 모델 파라미터 및 초기 Token을 불러올 수 있다. 불러온 Weight, Bias 및 생성되는 Token의 저장을 위한 On-chip 메모리가 존재하며 이는 FPGA의 URAM 및 Dual port BRAM 자원을 통해 구현되었다. 사용하는 모든 On-chip memory는 512bit 넓이의 Data port를 사용한다.

제안하는 가속기는 32 x 32 PE array를 통해서 행렬곱셈 연산을 지원하며, PE array 출력을 Vector Unit에 전달하여 Bias를 더하는 것으로 Linear layer 연산을 지원할 수 있다. Non-linear Function Unit은 Softmax, GELU, Layernorm의 3가지 Non-linear 연산을 지원한다. Softmax Unit은 Softmax 함수 내부의 자연 상수의 지수 연산 (e^x)이 아닌 정수 2의 지수 연산 (2^x)으로 변경하는 Approximated Base-2 Softmax 기법을 사용하며, 후보정 값을 추가하여 근사화에 의한 에러를

줄이는 방식을 사용한다. GELU Unit은 선형보간법 기반 근사화를 수행하는 NN-LUT 기법 [2]을 통해 구현 되었으며, 학습을 통해 최적의 선형 함수의 파라미터 값을 찾아 모델의 정확도 열화를 완화한다. Layernorm Unit은 mean(x). var(x)를 연산하는 기능을 수행하며, $1/\sqrt{x}$ 연산은 NN-LUT 기법을 적용해 구현되었다. Add & Layernorm의 Add, Weight, Bias 연산은 Vector Unit에서 수행된다.

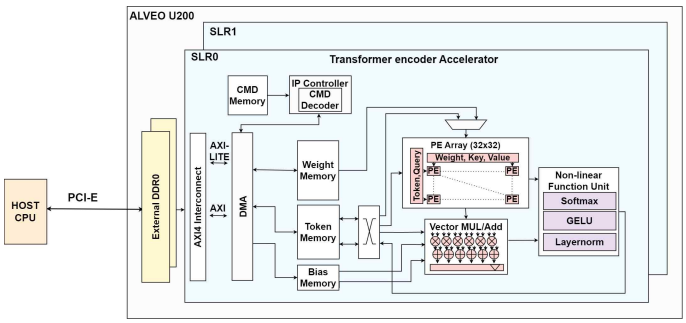

그림 1. 제안하는 가속기의 구조

3.Always-on-chip token기법

제안하는 가속기는 트랜스포머에서 사용하는 모든 연산을 지원하며, Host CPU 등 다른 프로세서의 도움 없이 종단 간 추론이 가능하다. 이는 PCI-E 혹은 AXI4 버스 등을 통한 가속기와 다른 프로세서 사이의 메모리 오프로딩 과정을 제거하여 추론에 소요되는 시간을 줄일 수 있다. 그러나 On-chip 메모리 영역이 부족하거나 효율적으로 매핑하여 사용하지 못하여, 완전히 사용되지 않은 데이터를 On-chip 메모리에서 Off-chip 메모리로 오프로딩을 해야 하는 문제는 여전히 존재한다. 본 논문에서는 Token memory로 사용되는 BRAM 자원을 6개의 영역으로 나누고, 이를 효과적으로 On-chip 메모리에 매핑하여 Off-chip 메모리로의 오프로딩 과정을 제거하는 Always-on-chip token 기법을 적용한다. 해당 기법은 메모리 상의 데이터를 다 사용하기 전까지 해당 메모리 영역의 덮어쓰기를 방지하도록 메모리 매핑을 미리 지정하는 방법이다. 그림 2는 Always-on-chip token을 위한 메모리 매핑 방법을 보여주며, 그림에서 나타난 순서와 메모리 매핑을 따라 연산을 수행하고 결과를 저장한다.

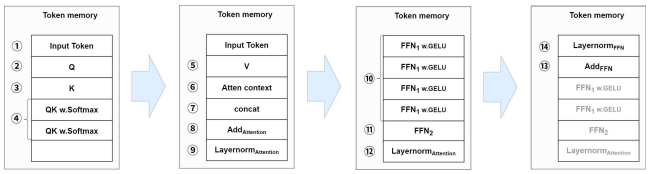

그림 2 Always-on-chip token을 위한 메모리 매핑

4.Non-linear function fusing기법

입력을 여러번 재사용할 필요가 있는 행렬곱셈과 달리, Softmax 혹은 GELU는 벡터 혹은 스칼라를 기본 연산의 단위로 사용하기 때문에 입력 데이터가 한번만 사용되는 특징이 있다. 이 점에 착안해, 제안하는 가속기는 PE array 혹은 Vector Unit의 출력을 물리적인 연결을 통해 On-chip 메모리가 아닌 Non-linear function Unit으로 우회하여, Non-linear 연산까지 동시에 처리한 뒤 On-chip 메모리에 저장하는 operation fusing 기법을 적용한다. 이 기법은 On-chip 메모리 접근 수를 줄여 추론 성능을 향상시킨다.

Ⅲ. 실험 및 결과

제안하는 가속기는 Verilog HDL을 이용하여 RTL-level에서 설계되었으며 16-bit fixed-point 연산을 사용한다. 하드웨어 개발 및 FPGA 검증은 Xilinx사의 Vivado 및 Vitis를 통해 수행하였다. 또한 제안하는 가속기는 Xilinx사의 ALVEO U200 FPGA를 사용하여 180MHz 주파수로 구현되었으며, U200 장치 내 두 개의 SLR (Super Logic Region)에 각각 하나씩, 총 2개의 가속기 인스턴스를 생성하였다. 또한 추론 성능을 평가하기 위하여 Pytorch 프레임워크와 함께 HuggingFace에서 제공하는 API를 이용하였으며, 비교군으로는 동일 컴퓨팅 시스템에 장착된 intel사의 i9-7900x CPU 및 Nvidia사의 Jetson TX2를 사용하였다. 표1은 U200 장치의 단일 SLR에서 구현한 가속기의 자원 사용량 및 활용률을 보여준다. 이는 Always-on-chip token의 적용을 위해, DSP 및 CLB 대비 많은 URAM 및 BRAM 자원이 On-chip 메모리로 사용되었음을 알 수 있다.

표 1. FPGA 자원 사용량 및 활용률 (단일 SLR)

자원	사용량 (활용률)
URAM	256 (80%)
BRAM	472 (65.63%)
DSP	1084 (47.54%)
CLB	22479 (45.63%)

표 2는 트랜스포머 인코더 모델인 BERT-Base [3]를 대상으로 각각 다른 연산 장치에서 처리 시간, 처리량 등을 비교를 나타낸다. 제안하는 가속기는 처리 시간은 i9-7900X와 비슷한 성능을 보였지만, 2개의 인스턴스를 사용하여 처리량은 비교군 대비 각각 1.76배, 2.24배 높은 성능을 보였다.

표 2. 장치 별 Bert-Base 모델의 처리량 비교

장치	처리 시간 (ms)	처리량 (추론 수/sec)
i9-7900X	0.08	12.5
Jetson TX2	0.102	9.8
U200	0.091	21.97

Ⅳ. 결론

본 논문은 트랜스포머 모델에서 사용하는 모든 연산을 지원하여 소프트웨어의 추가 처리 없이 트랜스포머 인코더를 종단 간 처리하는 가속기 구조를 제안한다. 또한 Off-chip 및 On-chip 메모리 접근 횟수를 줄이기 위한 Always-on-chip token 기법 및 Non-linear function fusing 기법을 추가로 적용하여 메모리 접근 횟수를 감소시켰다. 제안하는 가속기는 BERT-Base 모델을 대상으로 처리 시간, 처리량을 평가하였으며, 비교군 대비 최대 2.24배 높은 처리량을 보였다.

ACKNOWLEDGMENT

이 논문은 2020년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임(No. 2020-0-01304, 모바일 자가 학습 가능 재귀 뉴럴 네트워크 프로세서 기술 개발).

참 고 문 헌

[1] Vaswani, A. "Attention is all you need." Advances in Neural Information Processing Systems (2017).

[2] Yu, Joonsang, et al. "Nn-lut: neural approximation of non-linear operations for efficient transformer inference." Proceedings of the 59th ACM/IEEE Design Automation Conference. 2022.

[3] Devlin, Jacob. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).