

RISC-V 기반의 저전력 GEMM 연산 가속기 하드웨어 구현

박수빈, 노수민, 정기석*

한양대학교

subinpark@hanyang.ac.kr, smrho@hanyang.ac.kr, *kchung@hanyang.ac.kr

Hardware Implementation of RISC-V-based Low Power GEMM Operation Accelerator

Su bin Park, Soo Min Rho, Ki-Seok Chung*

Hanyang Univ., Seoul, Korea

요약

최근, 다양한 딥러닝 기술이 임베디드 디바이스에서 활용되고 있다. 그러나 딥러닝 모델의 핵심 연산인 GEMM을 지원하는 기존의 MAC(Multiply-Accumulate) 기반의 systolic array 구조는 높은 전력 소모를 유발하므로 저전력 동작이 필수인 임베디드 환경에서 연산하기에 한계를 갖는다. 본 논문은 RISC-V 프로세서에 연결된 GEMM 코-프로세서를 제안하며 로그 산술 변환에 기반한 저전력 systolic array를 통해 낮은 전력 소모를 달성하였다. 더불어 코-프로세서를 동작시키기 위해 RISC-V 기반의 네 가지 사용자 정의 명령어를 제안하였고 전체 연산 시간을 줄였다. 제안하는 저전력 기반의 GEMM 코-프로세서는 기본 RISC-V 프로세서와 비교하여 연산 시간은 최대 1693배 빨라졌고, 일반적인 systolic array 코-프로세서 구현과 비교하여 회로 면적은 5.62% 적었고, 전력 소모량은 17.91% 절감됨을 보였다.

I. 서론

최근, 딥러닝을 포함한 다양한 AI 기술이 모바일 및 임베디드 디바이스에서 활발히 활용되고 있다. CNN, 트랜스포머와 같은 딥러닝 모델에서는 메모리 접근량 및 연산량이 높은 GEMM (General Matrix Multiplication)이 핵심 연산에 포함된다. Systolic array는 PE (Processing Element) 연결을 통해 데이터 재사용을 극대화하여 GEMM 연산을 효율적으로 수행하는 PE array 구조이다. 그러나 이는 다수의 PE가 상호 연결되는 구조이므로 높은 전력 소모를 유발하고, 이는 배터리로 전력을 공급하는 임베디드 환경에 적합하지 않다.

RISC-V[1]은 임베디드 환경에서 많이 사용되는 무료 개방형 ISA (Instruction Set Architecture)이다. 특히, RISC-V는 사용자 정의 명령어를 지원하므로 특정 어플리케이션의 가속을 가능하게 한다. 본 논문은 RISC-V ISA를 기반하는 프로세서에 로그 산술 변환이 적용된 저전력 PE 기반 systolic array 구조의 코-프로세서 (co-processor)를 추가하여 고성능의 GEMM 연산을 지원하면서도 낮은 전력 소모를 달성한다.

본 논문의 기여는 다음과 같다. 1) GEMM 연산 가속을 위해 RISC-V에 기반한 네 가지 사용자 정의 명령어를 제안한다. 2) 제안하는 명령어 처리를 위해 두 가지 코-프로세서를 설계한다. 첫 번째는 기존의 MAC (Multiply-Accumulate) PE systolic array를 적용하였으며, 두 번째는 Mitchell's Straight-Line Approximation[2] 기법에 기반하여, 저전력 환경에 적합한 로그 산술 PE systolic array를 적용하였다. 3) RISC-V 프로세서에 코-프로세서를 선택적으로 적용하여 연산 시간, 전력 소모 및 면적 사용량을 파악하고 비교하였다.

II. 본론

2.1 RISC-V 베이스라인 프로세서 및 제안하는 코-프로세서 구조

본 논문은 RISC-V 베이스라인 프로세서로 RV32IMC ISA를 지원하는 4단계 파이프라인 기반의 RI5CY[3]를 사용하였다. RI5CY 프로세서, 메모리, GEMM 코-프로세서를 포함한 전체 구조는 그림 1과 같다. GEMM은 16x16 8-bit 기반의 systolic array를 통해 연산되며 output stationary

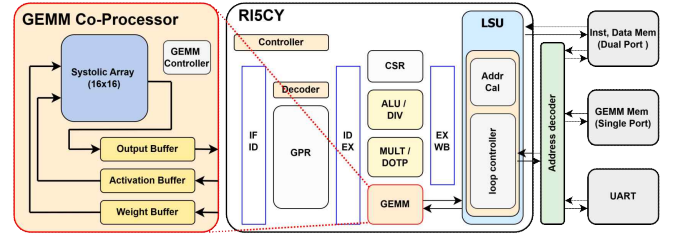


그림 1 제안하는 RI5CY[3] 기반의 GEMM 코-프로세서 및 전체 구조

dataflow를 따른다. 코-프로세서 내부에서 연산 전후의 데이터 저장에 위한 activation, weight, output 버퍼는 모두 1KB의 크기를 사용하였다.

2.2 제안하는 RISC-V 사용자 정의 명령어

제안하는 사용자 정의 명령어는 R-type 형식을 따르며 opcode는 사용자 정의 명령어를 위한 여러 opcode 중 0x0B를 사용하였다. 제안하는 네 가지 명령어의 구체적인 형식은 표 1에서 확인할 수 있으며 각 명령어의 설명은 다음과 같다.

1) LAC (Load activation matrix) : lac 명령어가 실행되면 행-우선 방식으로 저장되어 있는 메인 메모리로부터 데이터가 읽히며 행-우선 방식으로 activation 버퍼에 값이 저장된다. 행렬의 크기에 따라 메인 메모리의 접근 주소가 변화하므로 Rs1 레지스터의 상위 16-bit는 행의 크기, 하위 16-bit는 열의 크기가 저장되고 Rs2 레지스터에는 시작 주소가 저장된다.

2) LWET (Load weight matrix transpose) : lwet는 lac와 유사한 명령어로 메인 메모리로부터 가중치 값을 읽어오는 명령어이다. 이때, 가중치 행렬도 메인 메모리에 행-우선 방식으로 저장되어 있다. 그러나, GEMM

표 1. 제안하는 사용자 정의 명령어

Inst.	Funct7	Rs2	Rs1 (16b, 16b)	Funct3	Rd
lac	0	base_addr	a_row, a_col	0	/
lwet	1	base_addr	b_row, b_col	0	/
gemm	2	/	/	0	/
sout	3	base_addr	a_row, b_col	0	/

연산을 위해 두 번째 행렬의 데이터는 열-우선 방식으로 저장되어야 한다. 이를 위해 lwet 명령어가 실행되면 행 단위로 읽은 데이터가 전치된 후 내부 버퍼에 저장된다.

3) GEMM : gemm 명령어는 코-프로세서의 systolic array를 이용하여 activation과 weight 버퍼의 값을 GEMM 연산한 후 output 버퍼의 저장을 수행하는 명령어이다.

4) SOUT (Store output matrix) : sout 명령어는 연산 결과를 다시 메인 메모리에 저장하기 위한 명령어이다. Rs1, Rs2 레지스터를 통해 저장을 위한 메인 메모리의 시작 주소를 결정할 수 있다.

2.3 제안하는 저전력 GEMM 코-프로세서

본 논문은 저전력 GEMM 연산을 지원하기 위해 로그 산술 PE systolic array가 적용된 코-프로세서를 제안한다. 로그 산술 PE는 기존의 MAC 연산에 존재하는 곱셈 연산 ($a \times b$)을 1) [2]에 기반한 로그 변환 ($\log_2(a), \log_2(b)$), 2) 변환된 두 입력의 덧셈 ($\log_2(a) + \log_2(b) = \log_2(ab)$), 3) 로그 역변환 과정 ($2^{\log_2(ab)} = ab$)의 세 가지 단계로 대체한다. 그림 2의 (a)는 곱셈기와 덧셈기로 이루어진 기존의 MAC PE systolic array에 해당하며, 그림 2의 (b)는 로그 변환기 (log con), 로그 역 변환기(log deconverter), 덧셈기를 이용한 로그 변환 PE systolic array의 구조이다. 그림 2 (b)와 같이, 로그 변환 PE systolic array에서는 로그 변환기를 통해 입력 값의 로그 변환 전처리가 수행된다. 로그 변환기는 leading-1 detector와 leading-1 encoder를 통해 leading-1의 인덱스를 추출하여 로그 변환 값의 정수부를 결정하며, leading-1의 인덱스만큼 입력값을 shift 하여 로그 변환 값의 소수부(mantissa)를 결정한다. PE 내부에서는 덧셈기를 통해 두 입력의 로그 변환 값들을 더하고, 역 변환기를 이용해 최종 두 입력의 곱셈 결과를 얻는다.

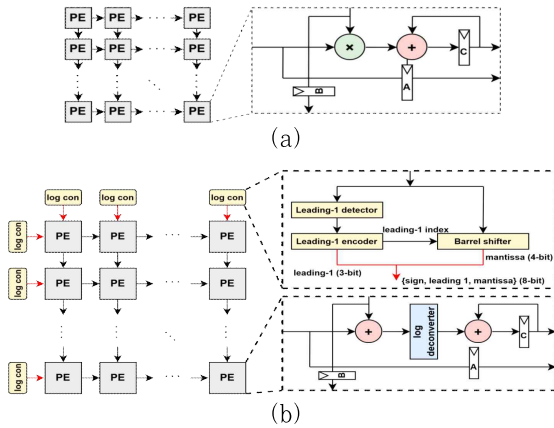


그림 2. (a) 기존 MAC PE 기반 Systolic Array 구조 (b) 로그 산술 PE Systolic Array 구조

2.4 성능 평가

제안하는 코-프로세서는 SystemVerilog를 통해 구현되었고 연산의 결과값을 확인하기 위해 UART(Universal asynchronous receiver and transmitter)를 활용하였다. 그림 3 및 표 2의 RI5CY는 베이스라인 프로세스인 [3]을 의미하며 RV-SA는 16x16 MAC PE systolic array로 구성된 코-프로세서, RV-LP는 16x16 로그 산술 PE systolic array로 구성된 코-프로세서를 의미한다. RV-SA와 RV-LP는 제안하는 명령어에 의해 동작한다. 그림 3은 ViT-Small 모델 구조의 몇몇 layer를 대상으로, RI5CY, RV-SA, RV-LP를 통한 다양한 크기 행렬의 GEMM 연산 시간을 비교한 결과이다. RV-SA와 RV-LP는 로그 변환기의 유무와 PE 구조

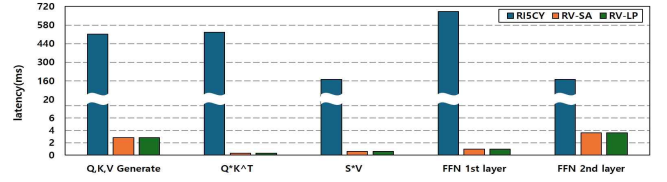


그림 3. ViT-Small 파라미터에 기반한 GEMM 연산 실행 시간 비교

표 2. 제안하는 코 프로세서 하드웨어 성능 비교

Processor	RI5CY	RV-SA	RV-LP
Area [μm^2]	8,311.7	106,018.01	100,062.95
Power [mW]	787.4595	1,773.6	1,455.9
Freq. [GHz]	2.2	2.2	2.2

의 차이만 있으므로 두 구조는 동일한 연산 시간을 갖는다. RI5CY에 대비해 RV-LP는 최대 1693배의 연산 시간 향상을 보였다. RI5CY는 단일 데이터 기반의 연산 및 저장 구조를 가지는 반면, RV-SA와 RV-LP는 사용자 정의 단일 명령어를 통해 여러 데이터를 순차적으로 처리하므로 높은 연산 속도를 갖는다. 특히, ViT-Small 파라미터의 $Q \times K^T$ 연산에서 행렬 데이터를 타일링 없이 한 번에 처리함으로써 activation의 데이터 재사용이 가능하므로 가장 높은 성능을 갖는다. 추가로, RV-LP는 RV-SA 대비 $2.7e-4$ 의 MSE(Mean Squared Error)를 보였다.

표 2는 Synopsys의 Design Compiler를 사용하여 15nm NangateOpenCell 라이브러리를 기반으로 합성한 결과를 나타낸다. RI5CY, RV-SA, RV-LP는 2.2GHz의 주파수로 동작하며 분석 결과, RV-LP는 RV-SA에 대비하여 면적은 5.62% 적었고, 전력 소모량은 17.91% 절감됨을 보였다. RV-SA는 256개의 곱셈기가 필요하지만 제안하는 RV-LP는 32개의 로그 변환기, 256개의 덧셈기와 로그 역변환기를 활용하므로 더 적은 면적 및 줄어든 전력 소모량을 갖는다.

III. 결론

본 논문에서는 전력 소모의 제약이 있는 임베디드 환경에서 GEMM 연산을 지원하기 위한 로그 산술 PE Systolic array가 적용된 저전력 코-프로세서를 제안하였다. 더불어, 코-프로세서를 제어하기 위해 RISC-V에 기반한 네 가지 사용자 정의 명령어를 제안하였다. 결과적으로, RV-LP는 RI5CY에 대비하여 최대 1693배 빠른 실행 시간을 가지며 RV-SA에 대비하여 면적 및 전력 사용량에서 각각 5.62%, 17.91%의 향상을 갖는다.

ACKNOWLEDGMENT

이 논문은 2024년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No. RS-2024-00409492)

참 고 문 헌

- [1] A. Waterman and K. Asanovi "The RISC-V instruction set manual, Volume I: Unprivileged ISA, Document Version 20190608-Base- Ratified", SiFive Inc, CS Division, EECS Department, University of California, Berkeley, Technocal Report, June 2019
- [2] J. N. Mitchell, "Computer multiplication and division using binary logarithms," IRE Trans. Electron. Comput., vols. EC - 11, no. 4, pp. 512 - 517, Aug. 1962
- [3] Gautschi, Michael, et al. "Near-threshold RISC-V core with DSP extensions for scalable IoT endpoint devices." IEEE transactions on very large scale integration (VLSI) systems 25.10 (2017):2700-2713.