Single Memory를 활용한 뉴럴 네트워크 프로세서용 효율적 Processing Element Array 구조 제안

이재학, 김병수, 송보배, 황태호
한국전자기술연구원
jhk507@keti.re.kr, bskim4k@keti.re.kr, 3232semi@keti.re.kr, taeo@keti.re.kr

Efficient Processing Element Array for Neural Network Processor using Single Memory

Jeahack Lee, Byung-Soo Kim, BoBae Song, Tae-ho Hwang
Korea Electronics Technology Institute

요 약

본 논문은 단일 메모리를 활용하여 효율적인 구조를 가지는 PEA(Processing Element Array)를 제안하였다. 제안하는 PEA는 복수의 메모리 대신 단일 메모리를 사용하여 SoC 설계 시 placement 측면에서 면적을 줄일 수 있을 것으로 기대된다. 또한, latency를 줄여 제안하는 PEA를 활용한 NNP (Neaueal Network Processor)의 idle 시간을 줄여 효율을 높일 수 있다. 다양한 크기의 filter size를 지원 할 수 있도록 재구성 가능하도록 PEA를 설계하다. 시뮬레이션을 통해 동작 검증을 수행하였으며 matlab과 연동하여 결과를 비교하였다.

Ⅰ. 서 론

ImageNet 프로젝트에서 AlexNet [1], VGG [2]등 CNN(Convolutional Neural Network) 기반의 딥러닝이 뛰어난 성능을 보이며 학계에 뉴럴 네트워크에 관한 관심도가 높아졌다. 이후, 알파고[3]의 등장으로 전세계적으로 딥러닝에 대한 뜨거운 관심이 쏟아졌고, 다양한 연구 분야에서 뉴럴 네트워크를 적용하여 기존 알고리즘과 비교하여 향상된 성능에 대한 연구 결과들이 나오기 시작하였다.

딥러닝의 뛰어난 성능은 높은 연산량을 요구하여 실시간 처리에 부적합한 문제가 있다. 이러한 문제를 해결하기 위해 뉴럴 네트워크 연산을 가속할 수 있는 CUDA (Compute Unified Device Architecture) [4] 등 다양한 GPU 기반의 프레임워크가 연구되었다. 하지만, GPU 기반의 딥러닝 시스템은 많은 전력소모를 필요로 하기 때문에 서버에 적합하고 엣지 디바이스에 적용이 불가능하다. 저전력 구현을 위해 NNP (Neural Network Processor)에 대한 다양한 연구[5-7]가 수행되고 있다. 기존의 연구들은 메모리를 연산단위별로 나누어 사용하기 때문에 placement 측면에서 큰 면적을 사용하게 된다. 본 연구는 단일 메모리를 사용하면서 PE(Processing Element)의 효율을 높이는 구조를 제안하였다.

본 논문은 2장에서 제안하는 PEA(Processing Element Array)를 설명한다. 3장은 간단한 예시와 함께 시뮬레이션 결과를 통해 동작 검증을 수행하였다. 4장에서 본 연구의 결론과 함께 추가 연구 방향에 대해 논의하였다.

Ⅱ. 제안하는 NNP의 PEA

본 논문에서는 연속적인 convolution layer와 fully-connected layer 연산에 최적화된 PEA 구조를 제안한다. Convolution layer 연산은 수식 (1)과 같다.

$$y = \sum_i w_i x_i + b = \sum_i p_i + b, \quad (1)$$

y는 convolution layer 연산의 결과를 의미하고, w_i는 i번째 가중치(weight), x_i는 i번째 입력(ifmap), b는 편향(bias), p_i는 i번째 부분합(partial sum)으로 가중치와 입력의 곱을 나타낸다. Convolution layer의 filter size는 1x1, 3x3, 7x7 등 다양한 크기를 가질 수 있어 PEA를 재구성 가능하도록 설계가 되었다. Fully-connected layer는 convolution layer에서 filter size가 1x1의 연산과 수식적으로 동일하여 1x1 filter를 사용하는 연산으로 대체 가능하다. 따라서, 본 논문에서는 convolution layer 연산에 대해서만 다루었다.

Convolution layer 연산을 하드웨어 관점에서 보면 입력(x)을 SRAM에 저장 후 PEA에 입력하여 연산 결과를 얻게 된다. 행렬 연산을 위해 동시에 입력을 위해서는 filter size 만큼의 register를 사용하기 때문에 하드웨어 구조적으로 효율적이지 않고, 순차적으로 입력을 하여 하드웨어 효율적인 구조의 경우 단순한 연산기 배치로 convolution layer 연산이 불가능하다. 본 논문은 그림 1과 같이 PE를 chain 형식으로 연결하여 SRAM의 순차적인 입력에 대해 효율적 처리가 가능한 PEA 구조를 제안한다.

PEA는 N개의 PE와 하나의 adder tree로 구성된다. PE는 filter size에 따라 구성이 되어 다양한 filter size에 대해 처리가 가능하다. 예를 들어 3x3 filter size의 경우 9개씩 PE가 묶음으로 convolution layer 연산이 수행되며, $\lfloor N/9 \rfloor$ 개의 filter 연산이 동시에 수행 가능하다. PE는 그림 2(a)와 같이 입력(x_i)과 해당 가중치(w_i)의 곱연산을 통해 가중치를 계산

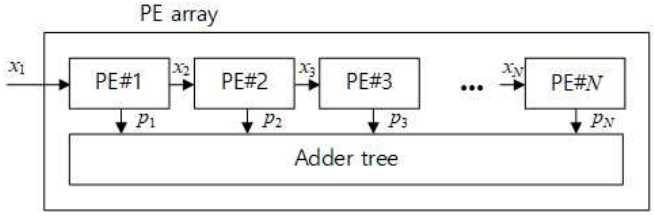

그림 1. 제안하는 PE array 구조.

하며, 다음 PE에 지연된 입력(x_{i-1})을 전달한다. Adder tree는 그림 2(b)와 같이 PE에서 연산한 부분합을 filter에 맞게 누적하는 연산을 수행한다. 새로운 filter 연산이 수행되는 경우 ps_i를 0으로 설정하여 누적 연산을 수행하고, 아닌 경우 이전에서 누적한 ps_i와 PE에서 연산한 p_i를 누적하여 다음 연산에 사용한다. 이와 같이 제안하는 PEA는 SRAM에서부터 출력되는 연속적인 입력에 대해 처리가 가능하여 latency를 줄일 수 있으며, 하드웨어 구현에도 적합한 구조이다.

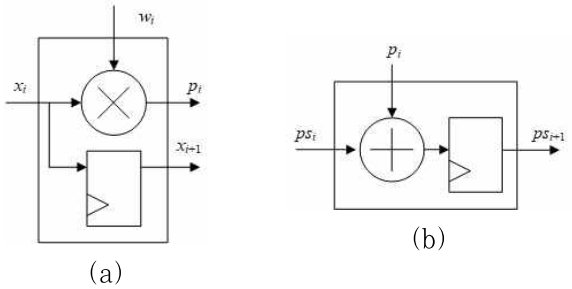

그림 2. (a) PE의 상세 구조. (b) Adder tree의 상세 구조.

Ⅲ. 시뮬레이션 결과

Verilog 언어를 사용하여 PEA를 설계하였으며 Cadence社의 irun을 사용하여 시뮬레이션을 수행하였다. 3x3 filter size에 대해 연산을 수행하였고, 가중치는 그림 3(a)와 같다. 입력은 그림 3(b)와 같으며 출력은 5bit fraction으로 shift 연산하여 그림 3(c)와 같다.

1	2	3
4	5	6
7	8	9

(a)

0	1	2	3	4
10	11	12	13	14
20	21	22	23	24
30	31	32	33	34
40	41	42	43	44

(b)

21	23	24
35	37	38
49	51	52

(c)

그림 3. 시뮬레이션 입력 (a) 가중치, (b) 입력, (c) 출력

설계한 PEA를 그림 3의 입력과 같이 수행한 결과 그림 4와 같이 동일한 출력 결과가 나오는 것을 확인 할 수 있다. 동일한 코드를 사용하여 1x1, 5x5, 7x7 filter size에 대해서도 시뮬레이션을 수행하였고 예측한 결과와 같은 출력을 얻어 동작을 확인하였다.

Ⅳ. 결론

본 논문에서는 하드웨어에 적합한 PEA 구조를 제안하였다. 제안하는 구조는 연속적인 입력에 대해 처리가 가능하며, 다양한 filter size를 지원하도록 재구성이 가능하도록 설계 되어 있다. 1x1, 3x3, 5x5, 7x7 filter size에 대해 시뮬레이션 검증을 수행하였으며 예측한 결과와 같은 출력을 얻을 수 있었다. 추가적인 연구를 통해 제안하는 PEA 구조를 사용하는 neural network processor 설계가 필요하며, FPGA 플랫폼 또는 SoC 설계를 통해 검증이 필요하다.

ACKNOWLEDGMENT

이 연구는 2020년도 산업통상자원부 및 산업기술평가관리원(KEIT) 연구비 지원에 의한 연구임('20009972').

참 고 문 헌

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in Proc. Adv. Neural Inf. Process. Syst., 2012, pp. 1097-1105.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, arXiv:1409.1556. (https://arxiv.org/abs/1409.1556).

[3] https://deepmind.com/

[4] D. K. Panda, K. Tomko, K. Schulz, and A. Majumdar, "The MVAPICH Project: Evolution and Sustainability of an Open Source Production Quality MPI library for HPC," in WSSPE, 2013..

[5] D. Shin, J. Lee, J. Lee, J. Lee and H. Yoo, "DNPU: An Energy-Efficient Deep-Learning Processor with Heterogeneous Multi-Core Architecture," in IEEE Micro, vol. 38, no. 5, pp. 85-93, Oct. 2018.

[6] D. Shin, J. Lee, J. Lee and H. Yoo, "14.2 DNPU: An 8.1TOPS/W reconfigurable CNN-RNN processor for general-purpose deep neural networks," 2017 IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, CA, 2017, pp. 240-241.

[7] J. Lee, C. Kim, S. Kang, D. Shin, S. Kim and H. Yoo, "UNPU: An Energy-Efficient Deep Neural Network Accelerator With Fully Variable Weight Bit Precision," in IEEE Journal of Solid-State Circuits.

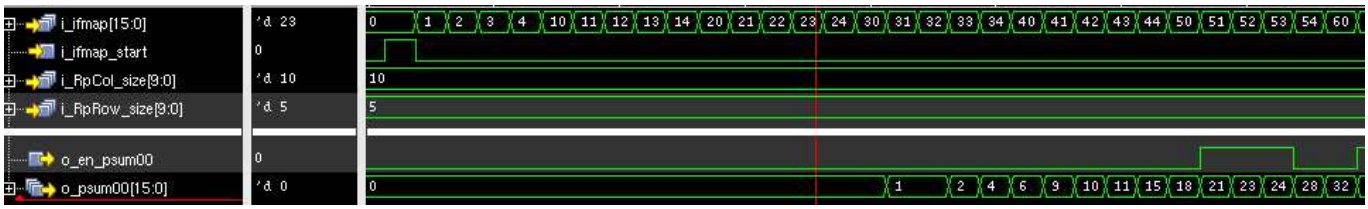

그림 4. 시뮬레이션 결과