경량 CNN 가속기를 위한 Radix-2 Booth 기반 가변 정밀도 곱셈기

금덕현 [1], 전승진 [1], 최재영 [1], 김지혁 [1], 김선희 [1]
[1] 상명대학교 시스템반도체공학과
201721093@smu.ac.kr, 201721129@smu.ac.kr, 201721132@smu.ac.kr, 201721103@smu.ac.kr,
happyshkim@smu.ac.kr

Radix-2 Booth-based Variable Precision Multiplier for Lightweight CNN Accelerators

Duck-Hyun Guem[1], Seung-Jin Jeon[1], Jae-Young Choi[1], Ji-Hyeok Kim[1], Sunhee Kim[1]
[1]Department of System Semiconductor Engineering, Sangmyung University

요 약

엣지 디바이스에서 딥러닝을 활용하기 위하여 CNN 경량화 연구들이 진행되고 있다. 경량 CNN 은 대부분 고정 소수점을 사용하며, 계층에 따라 정밀도는 달라진다. 본 논문에서는 경량 CNN 을 지원하기 위하여, 사용 계층에 따라 정밀도를 선택할 수 있는 가변 정밀도 곱셈기를 제안한다. 제안하는 가변 정밀도 곱셈기는 낮은 정밀도 곱셈기를 병합하는 구조로, 정밀도가 낮을 때는 병렬 처리를 통해 효율을 높인다. 제안하는 곱셈기를 Verilog HDL 로 설계하고 ModelSim 에서 동작을 확인하였다. 설계된 곱셈기는 계층별로 정밀도가 다른 CNN 가속기에서 효율적으로 적용될 것으로 기대된다.

1. 서론

딥러닝이 의료, 시설 보안, 자율주행 자동차 등 다양한 분야에서 활용되면서, 최근에는 엣지 디바이스에서도 딥러닝을 활용하려는 연구가 진행되고 있다 [1,2]. 일반적으로 엣지 디바이스는 대용량 데이터를 처리해야 하는 훈련 과정은 처리하지 않고, 경량화된 딥러닝 네트워크를 사용하여 추론(inference)만 진행한다. 그리고 CNN 을 바탕으로 하는 네트워크의 대다수는 64/32 비트 부동 소수점(floating point)을 사용하지 않고, 16/8 비트 고정 소수점(fixed point)으로도 충분히 정확한 성능을 나타내고 있다[3,4]. 따라서, 본 연구는 계층 별로 다른 정밀도를 요구하는 CNN 네트워크를 위하여 비트 정밀도를 선택할 수 있는 곱셈기 구조를 제안한다.

제안된 곱셈기는 곱셈 연산의 효율을 높이기 위하여 Radix-2 Booth 알고리즘[5]을 기반으로 하였으며, 16 비트와 8 비트, 두 가지 정밀도를 선택할 수 있다. 8 비트로 동작할 때는 동시에 2 개의 곱셈 연산이 가능하여 처리 시간을 단축시킬 수 있다.

본 논문의 순서는 다음과 같다. 2 장에서는 제안하는 곱셈기의 알고리즘과 구조를 설명한 뒤 Verilog HDL 로 설계하고, 시뮬레이션을 통해 검증한다. 마지막으로 3 장에서 마무리한다.

2. 정밀도 가변 곱셈기

정밀도 가변 곱셈기에서 곱셈 연산은 Radix-2 Booth 알고리즘을 기반으로 한다. Radix-2 Booth 알고리즘은 승수(multiplier) Q 에 대하여 $\{Q_n,Q_{n-1}\}$ 두 비트를 판단하여 덧셈/뺄셈을 하는 방법으로, add-shift 곱셈 방식보다 빠르며, 부호가 있는 수(signed number)에 대해 적용이 가능한 알고리즘이다.

제안하는 곱셈기에서는 16 비트와 8 비트, 두 가지 정밀도를 지원한다. 입력 신호 sel_mode 에 따라 비트 수가 결정된다. 첫 번째 방식은 sel_mode 가 0 일 때, 16 비트 입력 데이터에 대한 곱셈기로 동작한다. 16 비트 피승수(multiplicand)와 승수(multiplier)를 각각 상위 8 비트와 하위 8 비트로 구분하면, 두 수의 곱셈은 다음 식과 같이 표현할 수 있다.

$$\{A,B\} \times \{C,D\} = (Ax2^8 + B) \times (Cx2^8 + D)$$
$$= AxCx2^{16} + (AxD + BxC) x2^8 + BxD \quad (1)$$

2 의 거듭제곱 수의 곱셈은 비트 이동 연산(shift)으로 간단하게 표현할 수 있으므로, 곱셈 결과는 표 1 에 정리한 것과 같이 8 비트 곱셈 연산 4 번, 8 비트 비트 이동 연산 2 번, 그리고 덧셈 3 번으로 처리된다. 이때, 8 비트 곱셈의 결과는 16 비트이다.

<표 1> 8 비트 곱셈을 이용하여 16 비트 곱셈을 하는 방법

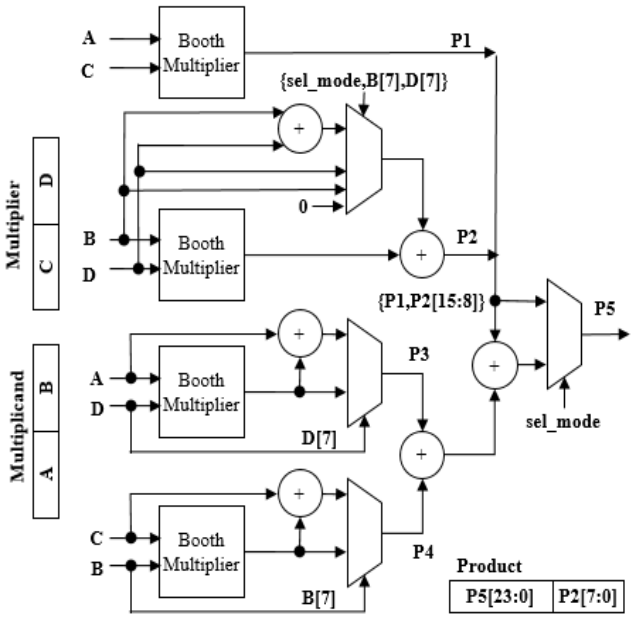
	MSB 8 bits	LSB 8 bits
Multiplicand	A	B
Multiplier	C	D
Product = {AC, BD} + (AD<<8) + (BC<<8)		

비트 이동 연산은 하위 8 비트에 0 을 채워주는 것이고, 0 은 덧셈의 항등원이다. 따라서 하드웨어 구현 시 비트 이동 연산은 필요하지 않다. 또한 3 번의 덧셈에 있어서, 32 비트 덧셈기 2 개가 필요하지 않다. A x D 와 B x C 는 16 비트 덧셈기를 이용하여 덧셈을 한다. 그리고, 그 결과를 {A x C, B x D}의 상위 24 비트와 덧셈을 한다. 즉 24 비트 덧셈기 하나가 필요하다. 따라서 8 비트 레지스터 2 개가 필요하지 않으며, 덧셈기의 비트 수는 약 60%로 줄일 수 있다.

Radix-2 Booth 곱셈기는 부호가 있는 수에 대하여 곱셈을 한다. 따라서 16 비트 곱셈을 8 비트 곱셈으로 나누어 처리하는 경우 부호 처리를 해야 한다. 우선 두 입력 데이터의 상위 비트 간의 곱셈인 A x C 는 Booth 곱셈기를 그대로 사용하면 된다. 하지만 부호가 없는 수(unsigned number)인 B, D 가 포함된 계산 AxD, BxC, BxD 의 경우에는 각 8 비트의 MSB 에 따라 동작을 구분해야 한다.

AxD 와 BxC 의 경우처럼 두 개의 입력 중 하나만 부호가 없는 수인 경우에는 부호가 없는 수 D 와 B 의 MSB 를 확인해야 한다. 만약 D 의 MSB 가 1 이면, AxD Booth 곱셈기의 결과는 A 와 D 의 2 의 보수의 절대값을 곱한 뒤 부호를 바꿔주는 결과를 만든다. 따라서 이 경우에는 A 에 2^8을 곱하여, Booth 곱셈 결과에 더해야 본래의 AxD 결과를 얻을 수 있다. 앞서와 마찬가지로 2^8 의 곱셈은 LSB 쪽에 0 을 8 번 채우는 것이며, 0 은 덧셈의 항등원이므로, Booth 곱셈 결과의 상위 8 비트와 A 를 더하는 것으로 하드웨어를 설계하면 된다.

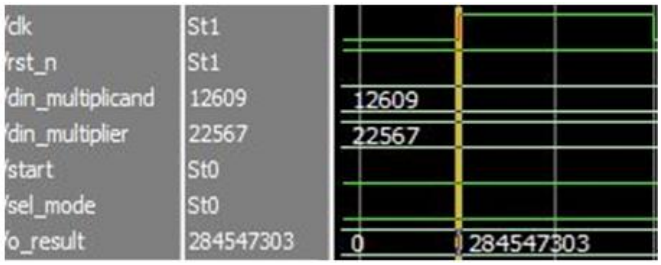
BxD 의 경우는 B 와 D 각각의 MSB 를 확인하여야 한다. 둘 중 하나만 MSB 가 1 일 경우에는 위의 계산 과정과 같다. 이 때는 MSB 가 0 인 입력을 Booth 곱셈의 상위 8 비트와 더해주면 된다. MSB 가 둘 다 1 일 경우 (B+D)를 Booth 곱셈의 상위 8 비트와 더해주면 된다. 그림 1 은 제안하는 곱셈기의 블록도이다.



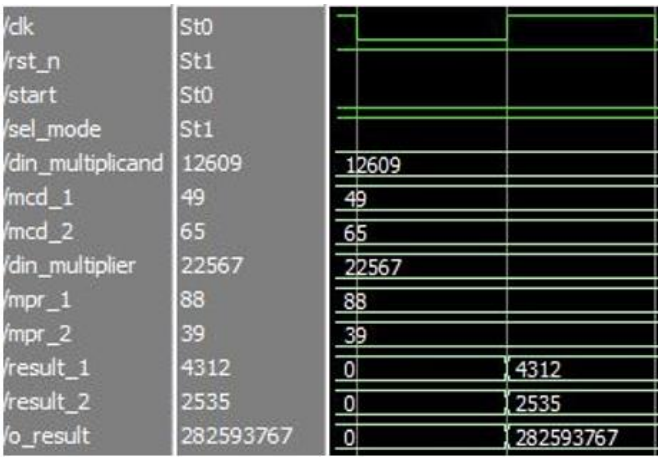
(그림 1) 제안하는 가변 정밀도 곱셈기 블록도

입력 신호 sel_mode 가 1 일 때는 2 개의 8 비트 곱셈기로 동작한다. 입력은 16 비트 모드일 때와 동일하게 {A,B}와 {C,D}로 주어진다. 이는 8 비트 입력 데이터 2 개가 16 비트 입력을 통하여 전달된 것으로, AxC 와 BxD 두 개의 곱셈 연산을 하면 된다. 즉 그림 1 의 블록도에서 4 개의 Booth 곱셈기 중 위에서부터 2 개의 곱셈기만 사용하며, 이때는 A, B, C, D 모두 부호가 있는 수이므로 MSB 확인 절차는 필요 없다.

그림 2 와 그림 3 은 제안하는 정밀도 가변 곱셈기를 Verilog HDL 로 설계한 후 ModelSim 에서 동작을 확인한 결과이다. sel_mode 가 0 일 때는 16 비트 모드로, 그림 2 에서 보는 바와 같이 입력 피승수 12,609 와 승수 22,567 를 곱하여 결과 284,547,303 를 얻었다. 그림 3 은 sel_mode 가 1 인 8 비트 모드의 시뮬레이션 결과이다. 입력은 16 비트 모드일 때와 동일하지만, 8 비트 모드에서는 상위 8 비트와 하위 8 비트가 독립된 데이터이다. 즉, 피승수 12,609(= 0x3141)는 49(= 0x31)와 65(= 0x41)로 구분해야 하며, 승수 22,567(= 0x5827)는 88(= 0x58)와 39(= 0x27)로 구분해야 한다. 따라서 결과는 4,312(= 49x88)와 2,535(= 65x39)이고, 두 결과를 32 비트로 합쳐서 표현하면 282,593,767(= 4,312x2^{16}+2,535)이 된다. 따라서, 제안하는 정밀도 가변 곱셈기가 정상적으로 동작함을 확인하였다.



(그림 2) 16 비트 모드일 때의 시뮬레이션 결과



(그림 3) 8 비트 모드일 때의 시뮬레이션 결과

3. 결론

본 논문에서는 Radix-2 Booth 알고리즘을 기반으로 한 가변 정밀도 곱셈기를 제안하였다. 8 비트 곱셈기 4 개를 병합하여 16 비트 곱셈기를 구성함으로써, 16 비트 곱셈이 가능하며, 8 비트 모드로 처리시에는 동시에 2 개의 곱셈 연산이 가능하다. Verilog HDL 로 설

계한 뒤 ModelSim 에서 동작을 검증하였다. 본 논문에서는 16 비트와 8 비트, 두 가지 정밀도만 제시하였지만, 낮은 정밀도 곱셈기를 병합하여 높은 정밀도 곱셈기를 만드는 것은 제안하는 구조로 모두 적용이 가능하므로, 4 비트/8 비트/16 비트/32 비트 등 다양한 정밀도를 제공하는 곱셈기에서 활용할 수 있다. 따라서, 계층별로 정밀도가 다른 CNN 가속기에서 효율적으로 적용될 것으로 기대된다.

감사의 글

본 연구는 과학기술정보통신부와 연구개발특구진흥재단이 지원하는 과학벨트 지원사업으로 수행된 연구결과입니다.

참고문헌

[1] 국경완, “인공지능 기술 및 산업 분야별 적용 사례,” 정보통신기획평가원 주간기술동향, vol. 20, pp. 15-27, 2019.

[2] 박현문, 황태호, “엣지컴퓨팅기술의 변화와 동향,” 한국통신학회지 정보와통신, vol. 32, no. 2, pp. 41-47, 2019.

[3] M. V. Valueva, N. N. Nagornov, P. A. Lyakhoc, G. V., Valuev, and N. I. Chervyakov, “Application of the residue number system to reduce hardware costs of the convolutional neural network implementation,” Mathematics and Computers in Simulation, vol. 177, pp. 232-243, 2020.

[4] C. Y. Lo, F. C. M. Lau and C. Sham, "Fixed-Point Implementation of Convolutional Neural Networks for Image Classification," 2018 International Conference on Advanced Technologies for Communications (ATC), Ho Chi Minh City, Vietnam, 2018, pp. 105-109.

[5] Booth, Andrew Donald, "A Signed Binary Multiplication Technique", Oxford University Press, pp. 100–104, 2018.