Systolic Array를 이용한 3x3 Convolution 연산기 설계

*김형순 *이준희 *서영호
*광운대학교
*gudtns1541@kw.ac.kr *joonhi96@kw.ac.kr *yhseo@kw.ac.kr

Design 3x3 Convolution Calculator with Systolic Array

*Kim, Hyeong-Sun *Lee, Jun-Hee *Seo, Young-Ho
*KwangWoon University

요약

본 연구는 Convolution Neural Network에서 사용되는 Convolution 연산기를 Systolic Array를 이용하여 구현한다. 두 개의 층으로 나뉜 연산기에 고정 소수점 값을 가지는 커널 값과 연속적인 입력을 넣고 정확한 출력이 나오는지 확인한다. 연산기 구현은 Verilog HDL로 하였으며 대조 연산은 Python에서 진행하였다.

1. 서론

현대의 딥러닝 기법 중에서 이미지 인식에 많이 이용되고 있는 Convolution Neural Network(CNN) 의 경우 그 연산에 콘볼루션(Convolution) 연산이 사용되게 된다. Convolution 연산은 행렬의 곱셈과 덧셈으로 이루어진 연산이므로 순차 연산을 할 경우에 그 속도가 상당히 느려지게 되므로 병렬 구조를 사용하는 것이 비교적 좋다. 본 논문에서는 Systolic Array라는 병렬 구조를 이용한 Convolution 연산기를 Verilog HDL을 이용해 설계하고 그 결과를 Python을 이용해 대조해본다.

2. 콘볼루션 연산

콘볼루션 연산이란 다음과 같은 식으로 정의한다.

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau) g(t-\tau)\, d\tau \quad (1)$$

하지만 이는 연속적인 함수에서 적용되는 연산식이고 CNN에서 사용할 행렬에서의 Convolution 연산은 조금 다르다. 두 개의 행렬이 있을 때 두 행렬을 곱하고 나온 행렬의 모든 원소들을 더해주는 것이 행렬에서의 Convolution 연산이다. 행렬의 곱셈은 앞 행렬의 행 원소들과 뒤 행렬의 열 원소가 곱해지는 형태이고 순차적인 연산기에서는 연산 속도가 느리므로 병렬 구조에서의 연산이 필수적이다.

3. 제안한 하드웨어의 구조

연산기를 구성하는 PE(Process Element)를 설계하고 이를 Systolic Array[1] 구조에 맞추어서 연결한다. 입력 값과 커널 값을 Python에서 임의로 설정해서 가져온다. 이때 입력 값들은 unsigned data이고 커널 값들은 signed data이다. 입력 값은 연속적으로 연산기에 들어가고 커널 값들은 각 PE에 저장되어 Convolution 연산을 실행한다. 이때 모든 값들은 고정소수점 값을 가지게 되므로 연산이 지속될수록 그 값이 커지게 되는데 정수부와 소수부의 비트 수를 일정 비트 이상으로 커지지 않게 제한함으로써 병목 현상을 방지하여 연산 속도의 저하를 막는다. 고정소수점 값이 계속해서 곱해지고 더해지는 연산기의 구조 상 비트를 제한하는 것은 필수적이다. 하지만 비트를 제한하게 되면 데이터의 손실이 일어나게 되므로 데이터의 손실을 최소화하면서도 비트가 너무 커지지 않게 하는 것이 중요하다.

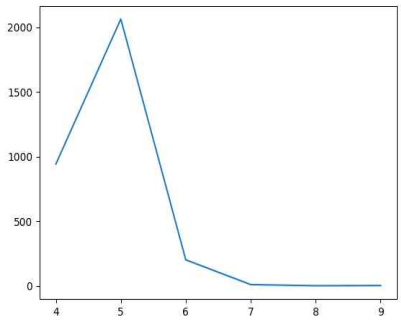

그림 1. 제한하는 Bit 와 그에 따른 데이터 오차에 대한 그래프
Figure 1. Limited Bit - Error Late graph

비트 수를 제한해가며 시뮬레이션을 돌렸을 때 7비트부터 오차가 거의 없어지는 것을 볼 수 있다. 따라서 소수부의 비트는 7비트로 제한하고 연산기를 설계했다. 정수부는 부호 비트를 제외하고 13비트까지 사용하였다.

Control unit과 data path를 따로 제작하여 연산자 구현을 진행하였다. 1층 연산자에서 외부 입력을 받고 그 입력값이 PE에 한 행씩 순차적으로 datapath에 대입이 된다. 대입 신호는 control unit에서 나와 조절한다. 첫 번째 입력신호의 계산이 완료되면 PE는 다음 입력신호를 받아들이고 PE의 출력값은 덧셈기에 의해 더해진다. 예를 들어 3x3 filter를 가진 Convolution 연산이라면 PE에는 한 행씩 3번 입력이 들

어가게 되고 PE의 출력값은 한 입력에 3개의 출력이 나온다. 세 값을 덧셈기를 통해 더하여 1층 연산자의 출력이 결정된다.

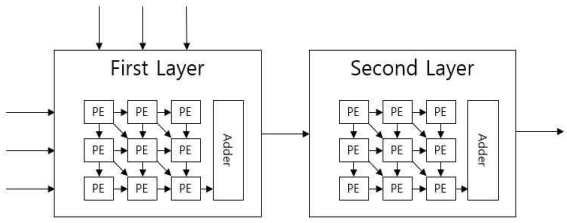

그림 2. 연산기의 블록 구조도
Figure 2. Calculator Block Diagram

2층 연산자는 1층 연산자의 출력을 입력으로 받는다. 2층 연산자 또한 Convolution 연산이기에 입력값이 연산을 하기 알맞은 상태가 될 때까지 값을 유지해야한다. 연산을 할 수 있는 상태가 되면 PE에 대입하여 한 행씩 계산을 진행하고 이후 출력값을 더해 최종 결과값이 출력된다. 연속적인 값을 받고있기 때문에 2층 연산자는 다음 행의 값을 새롭게 받고 기존에 있던 연산 입력값을 한 행씩 올려서 다음 연산 입력값을 만들어 PE에 대입을 진행한다. 계산이 다음 열로 넘어가면 앞선 열에서 첫 번째로 계산했던 입력값을 한 열씩 왼쪽으로 옮겨서 맨 오른쪽 열이 새로운 값으로 채워지도록 설계했다. 동일한 입력값을 피하기 위해 이런 식으로 제작하였다. 입력값이 다 주어지고, 계산도 마무리가 되면 usignal로 연산이 마무리 되었다는 신호가 주어진다. 모든 연산은 datapath에서 진행이 되며 datapath를 조절하는 것은 입력값을 제외하면 control unit이다.

4. 구현결과

임의로 설정한 커널 값은 다음과 같다.

b1=13'b1111111100011; //-0.21875 1's complement
b2=13'b0000000011000; //0.1875
b3=13'b1111111000000; //-0.4921875 1's complement
b4=13'b1111111101011; //-0.15625 1's complement
b5=13'b0000000110010; //0.390625
b6=13'b0000000010111; //0.1796875
b7=13'b1111111000000; //-0.4921875 1's complement
b8=13'b1111111001110; //-0.3828125 1's complement
b9=13'b0000000001011; //0.0859375

입력값은 2358개의 임의의 값을 Python에서 받아와 연속적인 입력을 넣었다. 입력 값들은 전부 8비트 값을 가지게 하였고 그에 따른 출력은 21비트 값을 가지게 하였다. 다음의 표는 입력과 그의 따른 출력을 Verilog HDL과 Python으로 각각 계산한 결과이다.

표 1. 고정소수점 형식의 콘볼루션 연산 과정

Input 1	8'b11101001; //233.0
Input 2	8'b01010001; //81.0
Input 3	8'b10110111; //183.0
Input 4	8'b01110001; //113.0
Input 5	8'b01010001; //81.0
Input 6	8'b01100001; //97.0
Input 7	8'b00011100; //28.0
Input 8	8'b00110111; //55.0
Input 9	8'b00110010; //50.0
Verilog HDL Output	000000001001001000110; //36.546875
Python Output	000000001001001000110; //36.546875

Input 1	8'b10100010; //162.0
Input 2	8'b11111101; //253.0
Input 3	8'b01100111; //103.0
Input 4	8'b11000001; //193.0
Input 5	8'b10001001; //137.0
Input 6	8'b11110100; //244.0
Input 7	8'b00011000; //24.0
Input 8	8'b00000111; //7.0
Input 9	8'b11111111; //255.0
Verilog HDL Output	000000110110101001100; //218.59375
Python Output	000000110110101001100; //218.59375

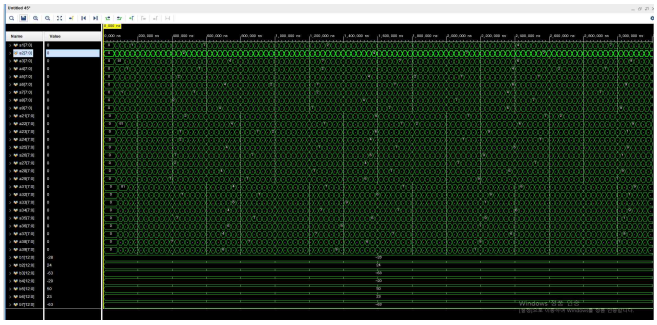

그림 3. 연속적인 입력에 대한 콘볼루션 연산의 시뮬레이션 결과
Figure 3. Simulation result of the convolution calculation for continuos input

5. 결론

Convolution Neural Network에서의 연산을 위한 연산기를 Verilog HDL을 이용해 제작하였고 임의로 지정한 커널 값을 이용하여 연속적인 입력을 넣었을 때 그 결과가 Python에서 연산한 결과와 일치하는 것을 확인하였다. 제안한 하드웨어의 구조를 실제로 구현 했을 때 정확하게 작동하였고 이를 통해서 CNN 내부에서 필요한 연산을 정확하게 수행할 수 있을 것으로 기대된다.

감사의 글

이 논문은 2021년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(2018R1D1A1B07043220).

참고문헌

[1] H.T Kung, “Systolic architecture, which permit multiple computations for each memory access, can speed execution of compute-bound problems without increasing I/O requirement”, Institute of Electrical and Electronics Engineers(IEEE), 1982, pp. 37-45