FPGA 를 이용한 신경망의 파이프라인 설계

경동욱, 정기철
숭실대학교 미디어학과
e-mail : {kiki227, kcjung}@ssu.ac.kr

Pipelined Design of a Neural Network Using FPGA

Dongwuk Kyoung, Keechul Jung
School of Media, Soongsil University

요 약

본 논문에서는 부동소수점 연산을 사용하면서도 빠른 처리속도를 가지는 신경망의 파이프라인 설계를 제안한다. 부동소수점 연산은 고정소수점 연산보다 느린 처리속도와 많은 면적으로 일반적인 하드웨어 구현에서 잘 사용되지 않지만, 제안된 구조에서는 고정소수점 연산보다 더 정확한 값을 계산할 수 있는 부동소수점 연산을 사용하며 부동소수점의 느린 처리 속도를 보완할 수 있도록 파이프라인 구조를 사용한다. 파이프라인 구조의 성능을 검증하기 위해 2 가지의 서로 다른 구조의 신경망을 사용한다. 실험 환경으로는 Xilinx XC2V8000 칩과 Xilinx ISE 6.2 의 합성 도구를 사용한다. 실험 결과는 파이프라인 구조일 때의 신경망은 각각 7 클럭, 8 클럭이 소요되고, 파이프라인 구조가 아닐 때 각각의 신경망은 77 클럭, 84 클럭으로써 파이프라인 구조일 때 약 10 배의 빠른 처리를 가진다.

1. 서론

신경망은 기본적으로 고등동물의 두뇌 구조와 그 기능을 본떠 기존의 컴퓨터 시스템에서 해결하기 어려운 문제, 즉 비선형성을 가지는 문제나 최적화 문제 등에 이용하고자 개발되었다. 그러나 실제의 응용에 있어서 신경망은 단위 시간당 막대한 연산을 수행하여야 하는 문제가 발생하므로 실시간 처리가 요구되는 응용에 있어서는 하드웨어 구현이 사용되고 있다.

신경망의 하드웨어 구현에서 사용되는 연산은 부동소수점 연산 보다 빠른 처리속도와 적은 면적을 가지는 고정소수점 연산을 많이 사용한다 [1-3]. 하지만 신경망의 소프트웨어 구현에서는 고정소수점 연산보다 부동소수점 연산을 사용한다. 그 이유는 부동소수점 연산이 표현할 수 있는 수의 범위가 더 넓기 때문에 정확한 계산을 할 수 있기 때문이다.

또한, 신경망을 사용하는 분야 즉, 영상처리 및 패턴인식과 같은 분야에서는 한번의 수행으로써 결과가 처리되어 지기 보다는 반복된 수행으로써 최종 결과를 보여준다. 이때 반복된 수행에서 반복 연산의 주기가 짧을수록 최종 결과를 빠르게 처리할 수 있다[4, 5].

본 논문에서는 부동소수점을 사용하여 보다 정확한 값을 계산하며, 빠른 처리를 위해서 신경망의 파이프라인 구조를 제안한다. 신경망이 파이프라인 구조로 설계되었을 때, 첫 번째 처리는 많은 수행시간을 가지지만 두 번째 신경망의 처리는 은닉 층(hidden layer)의 뉴런 수만큼의 짧은 주기의 수행으로 처리 함으로써 영상처리 및 패턴인식과 같은 분야에서 빠르게 처리할 수 있다.

2. 하드웨어 구현

부동소수점 연산을 사용하는 신경망의 파이프라인 설계에서 입력 층과 은닉 층의 구조와, 은닉 층과 출력 층의 구조 그리고 내부 연산에서 사용되는 연산(덧셈, 곱셈, 시그모이드 함수)의 구조는 다음과 같다.

2.1 전체 시스템 구조

전체 파이프라인 구조에 앞서 2계층 신경망의 구

조에 대해서 살펴보자. 2계층 신경망은 입력 층(input layer), 은닉 층(hidden layer), 출력 층(output layer)으로 그림 1과 같이 구성된다.

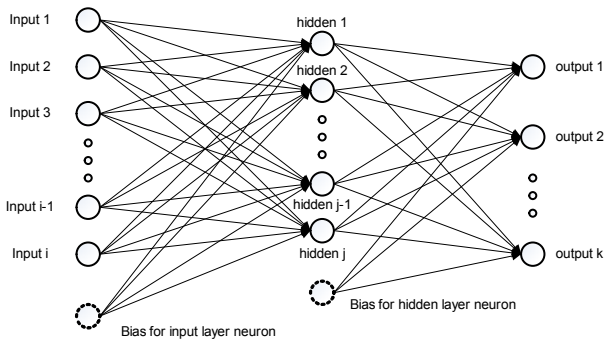

그림 1. 2 계층 신경망 구조.

본 논문에서 제안된 수행은 그림 2와 같이 모든 입력 값과 가중치($weight(\mathrm{i,j})$)을 가지고 곱셈, 덧셈 그리고 활성화 함수인 시그모이드 함수(sigmoid function)를 수행한다. 여기서 입력 층의 입력 값 수는 i, 은닉 층의 뉴런 수 j, 그리고 출력 층의 뉴런 수 k 라고 할 때 다음과 같은 식을 따른다.

$$hidden\ \mathrm{j} = a\left(\sum_{n=1}^{i}(weight(\mathrm{i,j}) \bullet input\ \mathrm{i}) + \mathrm{B}\right) \quad (1)$$

$$output\ \mathrm{k} += (weight\_m(\mathrm{j,k}) \bullet hidden\ \mathrm{j}) \quad (2)$$

입력 값($input\ \mathrm{i}$), 가중치($weight(\mathrm{i,j})$), 바이어스 값(B), 그리고 활성화 함수($a(x)$)을 가지고 hidden j를 처리하는 식 1이다. hidden j는 한번의 수행으로 하나의 뉴런이 처리된다. 식 2는 은닉 층과 출력 층을 계산하는 식으로 출력 층의 모든 뉴런은 은닉 층 뉴런이 처리되는 순으로 동시에 처리되며 은닉 층의 모든 뉴런이 처리될 때 출력 층 뉴런의 처리도 완료된다.

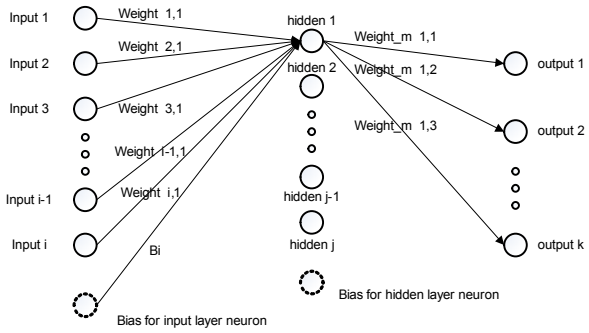

그림 2. 신경망의 수행도.

위와 같은 구조로 처리되는 이유는 전체 신경망의 파이프라인 구조를 이루기 위해서이다. 은닉 층 뉴런과 출력 층 뉴런이 어떻게 파이프라인 구조로 이루어져있는지 살펴보기 전에 부동소수점에 대해서 간략하게 살펴보고 덧셈연산과 곱셈연산 그리고 활성화 함수에 대해서 살펴보자. 부동 소수점은 IEEE 표준[6]으로써 부동소수점 수를 하나의 부호 비트, 하나의 바이어스 된 지수(biased exponent), 하나의 정규화된 가수(mantissa) 또는 유효수(significand)로 표현한다. 단정도(single precision) 32 비트 부동소수점 수(C 의 float)는 그림 3과 같은 구조를 가지며 본 논문에서 제안한 파이프라인 구조에서도 이와 같은 데이터 구조를 가진다.

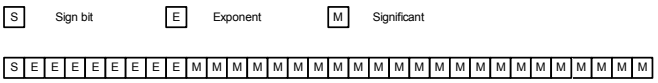

그림 3. 부동소수점의 구조.

부동소수점 수를 사용하는 덧셈과 곱셈에 대해서 살펴보면, 덧셈은 6단계, 곱셈은 4단계로 구성된 파이프라인 구조이다[7].

활성화 함수 $a(x)$ 는 가중치와 입력 값을 가지고 덧셈연산과 곱셈연산의 결과값을 가지고 뉴런의 결과값을 출력한다. 여기서 활성화 함수로 다음과 같은 시그모이드 함수를 사용한다.

$$a(x) = \frac{1}{1+e^{-x}} \quad (3)$$

시그모이드 함수는 지수연산과 덧셈연산 그리고 곱셈연산이 사용되는데 이 연산들은 많은 수행시간과 면적을 소요한다. 이 문제점을 해결하기 위해서 Piece Wise Linear(PWL)방식으로 구현하였다[8]. PWL 방식은 그림 4와 같이 비선형으로 이루어진 함수를 여러 개의 선형 함수로 처리하는 방식으로써 덧셈연산과 곱셈연산만으로 비선형 함수를 처리할 수 있다. 본 논문에서는 간단한 구현을 위해서 그림 4에서처럼 3 segment PWL 방식으로 시그모이드 함수를 구현하였다. 시그모이드함수는 파이프라인 구조의 덧셈연산과 곱셈연산의 사용으로 시그모이드 함수 또한 파이프라인 구조로 수행한다.

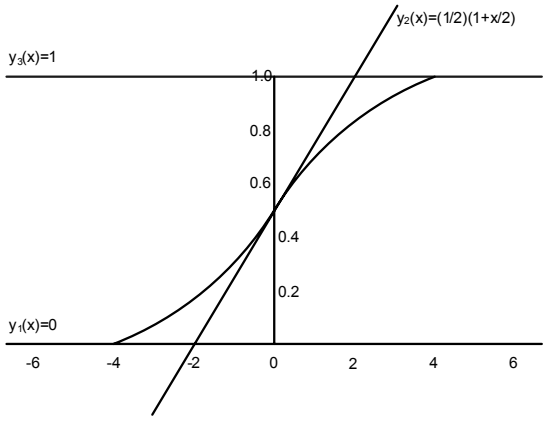

그림 4. 3 segment PWL 의 시그모이드 함수.

2.2 입력 층과 은닉 층의 연산 구조

입력 층과 은닉 층의 연산구조는 그림 5와 같으며 은닉 층 뉴런을 계산하기 위한 첫 수행은 입력 층의

뉴런 수에 따라 많은 클럭이 소요되지만, 은닉 층 다음 뉴런을 계산하기 위해서는 1클럭으로 수행이 처리된다. 입력 층과 은닉 층의 연산에서 사용되는 연산은 곱셈연산, 덧셈연산, 시그모이드 함수, 그리고 가중치 값을 저장하고 있는 롬(ROM)으로 구성된다. 곱셈연산은 입력 값과 가중치를 처리하기 위해서 입력 수만큼 필요하다. 또한 덧셈연산 수 역시 입력 수만큼 필요하며 시그모이드 함수는 하나 필요하다.

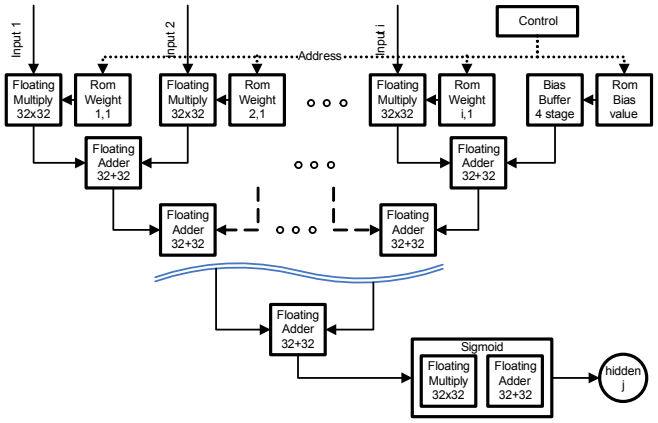

그림 5. 입력 층과 은닉 층의 하드웨어 구조.

여기서 덧셈연산의 구조는 입력 층 뉴런의 수에 따라 결정되는데, 입력 층의 뉴런 수(바이어스 포함)를 $i+1$라고 할 때 $i+1=2^n$ 이면 덧셈연산의 구조(즉, 덧셈연산의 층)는 n 의 층으로 이루어 진다. 또한 $2^n < i+1 < 2^{n+1}$ 일 때 덧셈연산의 구조는 $n+1$ 층으로 이루어진다. 그럼 입력 수에 따라 어떻게 달라지는지 예를 들어 살펴보자. 그림 6는 입력 수 15개일 때의 구조이며 그림 7은 입력 수 19개 일 때 덧셈 구조이다.

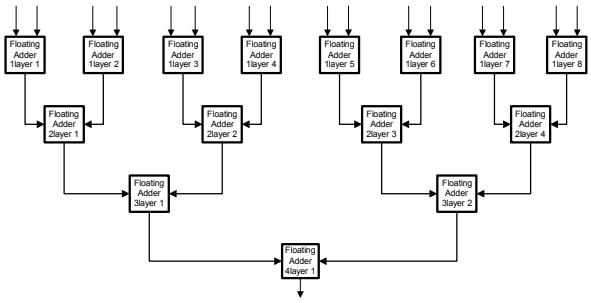

그림 6. 입력 수 15 개 일 때의 덧셈 구조.

입력 수가 15개 그리고 바이어스의 값을 포함하여 총 16개로써 $16=2^4=2^n$ 으로 되면 n개의 층으로 덧셈 구조는 설계된다. 하지만 입력 수가 19개와 바이어스의 값을 포함하여 총 20개로써 $20 \neq 2^n$ 으로 표현되지 않고 $2^4 < 20 < 2^5$ 으로써 5개의 층으로써 이루어진다. 여기서 총 입력수가 2^n 이면 버퍼(buffer)가 필요 없지만 그렇지 않으면 그림 7과 같이 덧셈연산을 하지 않은 값을 각 층마다 저장되어야 하기 때문에 버퍼가 필요하다.

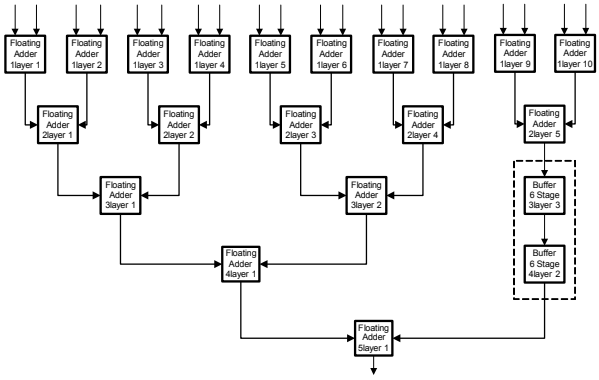

그림 7. 입력 수 19개 일 때의 덧셈 구조.

즉, 그림 7에서 2층5(2layer 5)의 덧셈연산의 결과 값을 3층 덧셈연산이 이루어지는 동안 유지하기 위해서 3층3(3layer 3)의 버퍼가 필요하며 또한 4층의 덧셈연산이 이루어지는 동안 4층2(4layer 2)의 버퍼에서 유지된다. 이렇게 입력 수에 따라 덧셈연산이 이루어지지 않은 값들은 버퍼로써 유지시켜 주며 이 버퍼는 덧셈연산과 같은 6단계를 가지는 버퍼 구조이다. 따라서 입력 층의 연산은 파이프라인 구조로써 바이어스를 포함한 입력 수에 따라 덧셈 구조는 달라지지만 은닉 층의 두 번째 뉴런부터는 1클럭 마다 계산된다.

2.3 출력 층의 연산 구조

은닉 층 뉴런의 결과는 1클럭에 하나의 뉴런이 처리된다. 이렇게 1클럭에 하나씩 처리되는 값을 가지고 파이프라인 구조를 이루기 위해서는 그림 8과 같은 구조가 필요하다.

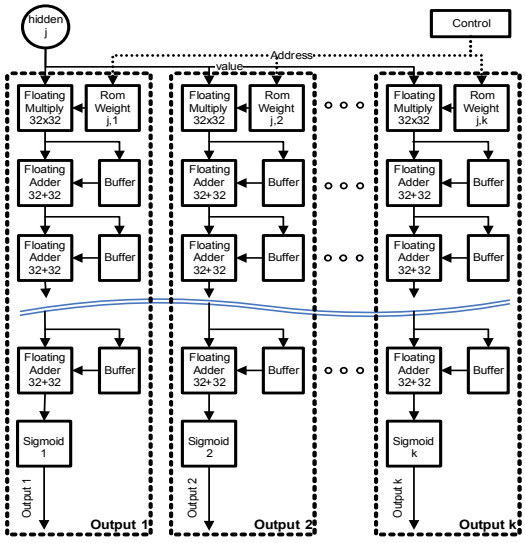

그림 8. 출력 층의 하드웨어 구조.

출력 층의 모든 뉴런들은 동시에 수행되며 은닉 층의 뉴런들이 수행이 완료된 후 출력 층의 뉴런들도 수행이 완료된다. 물론 출력 층의 구조 역시 파이프라인 구조로 구성되어 있기 때문에 초기 수행 시간이 필요하다.

출력 층에서 필요한 연산자의 수를 살펴보면 곱셈 연산과 시그모이드 연산이 출력 층 뉴런 수만큼 필요하며 덧셈연산은 은닉 층의 뉴런의 수와 바이어스를 포함한 총수를 $j+1$ 이라고 할 때 $j+1=2^n$ 이면 덧셈 연산의 수는 $n \times k$ 개이다. 또한 $j+1$ 이 $2^n < j+1 < 2^{n+1}$ 이면 덧셈연산의 수는 $(n+1) \times k$ 개이다. 이와 같이 은닉 층의 뉴런 수와 출력 층의 뉴런 수에 따라 출력 층의 하드웨어 구조가 달라진다.

3. 실험 결과

본 논문에서 제안된 방법을 실험하기 위한 실험환경은 Xilinx XC2V8000 칩을 타겟으로 Xilinx ISE 6.2를 사용하여 합성하였다. 기본적인 연산의 성능분석은 표 1과 같다.

표 1. 연산자의 성능분석

연산자	면적 (gates)	동작속도 (MHz)	클럭 수 (clk)	수행시간 (ns)
Adder	9,520	65.210	6	92.010
Multiplier	22,323	43.471	4	92.015
Sigmoid	30,041	42.658	10	234.423

위와 같은 연산을 이용하여 두 가지 신경망을 테스트하였다. 첫 번째 신경망(a)은 입력 층 뉴런은 15개, 은닉 층 뉴런 7개, 출력 층 뉴런 4개이고 두 번째 신경망(b)은 입력 층 뉴런은 19개, 은닉 층 뉴런 8개, 출력 층 뉴런 4개를 가지는 두 신경망의 성능을 분석하였을 때 결과는 표 2와 같다.

표 2. 2 개 신경망의 성능분석

신경망	면적 (gates)	동작속도 (MHz)	클럭수 (clk)	수행시간 (ns)
신경망(a)	812,378	39.034	77	1972.639
신경망(b)	939,689	38.858	84	2161.717

표 2는 한번의 신경망 수행을 의미하며 본 논문에서 제안된 파이프라인 구조의 성능을 분석하기 위해서는 한번의 수행이 완료된 후 다음 수행이 완료까지의 수행시간으로 확인할 수 있다. 표 3은 파이프라인 구조의 성능 분석으로써 첫 수행은 많은 수행시간이 걸리지만 두 번째부터의 수행은 은닉 층의 뉴런 수만큼의 클럭소요로 수행이 완료된다. 즉, 전체 신경망의 구조를 파이프라인 하였을 때 은닉 층 뉴런 수만큼의 짧은 주기를 보여준다.

표 3. 신경망의 순차 처리분석

신경망	첫 번째 수행 (clk)	두 번째 수행 (clk)	세 번째 수행 (clk)	수행 간격 (ns)
신경망(a)	77	7	7	179.331
신경망(b)	84	8	8	205.878

4. 결론

본 논문에서는 일반적으로 상용하는 고정소수점 연산보다 정확한 값을 계산하는 부동소수점 연산을 사용하여 신경망을 하드웨어로 설계하였으며, 느린 처리 속도를 해결하기 위해서 신경망을 파이프라인 구조로 설계하였다.

우리는 두 가지의 신경망을 가지고 실험하였다. 실험 결과에서 첫 번째 신경망은 파이프라인 구조로 설계 되었을 때 11배 빠른 처리 속도를 보여준다. 또한 두 번째 신경망은 10배 빠른 처리 속도를 보여주고 있다. 이러한 파이프라인 구조는 은닉 층 뉴런의 수만큼의 수행으로 한번의 수행이 이루어 짐을 알 수 있다. 따라서 영상처리 및 패턴인식과 같이 한번의 수행이 반복 처리되는 분야에서 짧은 주기를 가지는 파이프라인 구조를 이용하면 빠르게 수행할 수 있다.

참고문헌

[1] S. Coric, I. Latinovic and A. Pavasovic, “A Neural Network FPGA Implementation,” Neural Network Applications in Electrical Engineering. Proceedings of the 5th Seminar on NEUREL 2000. 2000:117-120.

[2] Ma Xiaobin, Jin Lianwen, Shen Dongsheng and Yin Junxun, “A Mixed Parallel Neural Networks Computing Unit Implemented In FPGA,” IEEE Int. Conf. Neural Networks & Signal Processing, 14-17,December 2003.

[3] Nan Bu, Taiji Hamamoto, Toshio Tsuji and Osamu Fukuda, “FPGA Implementation of a Probabilistic Neural Network for a Bioeletctric Human Interface,” Circuits and Systems, MWSCAS '04, Vol 3, July 2004.

[4] Keechul Jung, “Neural Network-based Text Localization in Color Images,” Pattern Recognition Letters, Vol. 22, No. 14, pp. 1503-1515, 2001.

[5] Keechul Jung and JungHyun Han, “Hybrid Approach to Efficient Text Extraction in Complex Color Images,” Pattern Recognition Letters, Vol. 25, Issue 6, pp. 679-699, Apr. 2004.

[6] “IEEE Standard for Binary Floating-Point Arithmetic,” ANSI/IEEE std 754-1985, New York, The Inst. Of Electrical and Electronics Engineers, Inc, Aug, 1985.

[7] Attila Hidvegi, “Implementation of Neural Networks in FPGA,” http://www.sysf.physto.se/~attila/ANN.pdf, 2002.

[8] K. Basterretxea, J.M. Tarela and I. del Campo, “Approximation of Sigmoid Function and the Derivative for Hardware Implementation of Artificial Neurons,” IEE Proceedings, Circuits Syst., Vol. 151, No. 1, February 2004.