MLP 기반의 문자 추출을 위한 하드웨어 구현

경동욱[1], 정기철[2]
숭실대학교, 정보과학대학, 미디어학과, HCI Lab.[1 2]
{kiki227[1],kcjung[2]}@ssu.ac.kr

Hardware Implementation for MLP Based Text Detection

Dongwuk Kyoung[1], Keechul Jung[2]
HCI Lab., College of Information Science, Soongsil University [1 2]

요 약

현재 많은 신경망의 하드웨어 구현은 부동 소수점 연산에 비해서 적은 면적과 빠른 수행시간을 가지는 고정소수점 연산을 많이 사용하지만, 소프트웨어에서는 일반적으로 높은 정확도를 가지는 부동소수점 연산을 사용한다. 신경망의 하드웨어 구현에서 많이 사용하는 고정소수점 연산은 부동소수점 연산에 비해서 빠른 처리속도와 적은 면적으로써 쉽게 하드웨어 구현에 용이하지만, 부동소수점 연산에 비해서 낮은 정확도와 기존의 부동소수점 연산을 사용하는 소프트웨어 신경망을 쉽게 적용할 수 없는 단점을 가진다. 본 논문에서는 부동소수점 연산을 사용하여 문자 추출 MLP의 데이터 변환 없이 적용할 수 있는 전체 파이프라이닝 설계 구조를 제안한다. 제안된 설계방법은 신경망의 전체 구조를 입력층과 은닉층을 링크 병렬화 방법과 은닉층과 출력층을 뉴런 병렬화 방법을 개선하여 쉽게 파이프라이닝 구조로 설계함으로써 신경망 처리는 은닉층 뉴런수와 동일한 주기로 처리되며, 기존의 문자추출 소프트웨어 신경망을 제안된 하드웨어 설계방법으로 구현하였을 때 11배의 빠른 성능을 나타낸다.

Keyword : Neural Network, MLP, Pipelining, Text Detection, FPGA, Floating-point arithmetic

1. 서론

영상 내의 문자 추출은 입력 영상 내의 문자 영역을 자동으로 추출하는 것을 의미한다. 본 논문에서는 입력 영상 내의 문자의 유무를 판별하기 위한 질감 분석기로써 신경망을 사용하였다. 신경망을 사용하여 문자 영역과 비-문자 영역으로 나누는 방법이다 [1-2]. 이러한 소프트웨어 신경망은 많은 시간을 소요함으로써, 실시간 처리가 가능한 하드웨어 구현으로 많이 사용한다.

일반적으로 하드웨어 구현에서는 적은 면적과 빠른 처리속도로 인해서 고정소수점 연산을 많이 사용하지만, 소프트웨어 신경망은 높은 정확도를 위해서 부동소수점 연산을 사용한다. 소프트웨어 신경망의 빠른 처리를 위해서 구현이 용이한 고정소수점 연산을 사용하는 하드웨어 구현에 적용하기 위해서는 데이터의 변환이 필요하다는 문제로 쉽게 적용하기가 어려웠다. 최근에는 고정소수점의 낮은 정확도와 낮은 유연성 문제로 하드웨어 구현에서도 부동소수점 연산을 사용하는 연구가 진행 중이다[3-4]. 32 비트의 부동소수점 연산 대신 10 비트의 부동소수점 연산을 사용하여 일반적인 32 비트 부동소수점 연산보다 빠른 처리를 수행하는 연구[3], 부동소수점 연산과 고정소수점 연산을 병행하여 빠른 처리를 수행하는 연구가 진행되었다 [4].

위의 연구들은 부동소수점 연산을 개선하여 빠른 처리속도를 가지는 신경망에 관한 연구이며, 다음은 신경망의 구조를 개선하여 효율성을 높이는 연구이다. 신경망은 기본적으로 병렬 또는 분

산처리의 구현이 용이하며, 각각의 뉴런은 단위적인 동작을 수행한다. 일반적으로 신경망의 병렬화 방법은 계층 병렬화(layer parallelism), 뉴런 병렬화(neuron parallelism) 그리고 링크 병렬화(link parallelism)의 3 가지로 정의한다[5]. Xiaobin 등은 입력층과 은닉층 사이에서의 링크 병렬화 기법과 은닉층과 출력층 사이에서의 뉴런 병렬화 기법을 혼합하여 적은 면적으로 빠른 처리속도를 가지는 8 비트 고정수소점 연산을 사용하는 신경망을 설계하였다 [6]. 그러나 고정소수점 연산은 입력과 동시에 결과를 출력하는 조합회로로써 뉴런 병렬화를 쉽게 구현할 수 있지만, 부동소수점 연산은 느린 처리속도를 보완하기 위해서 보통 파이프라인구조로 설계되므로 결과는 입력후 몇 단계의 처리과정을 수행하여 출력함으로써 뉴런 병렬화가 어렵다.

본 논문에서는 MLP기반의 문자 추출을 실시간 처리를 위해서 부동소수점 연산의 느린 처리속도를 해결하는 전체 파이프라이닝(Fully-Pipelining) 구조를 제안한다. 제안된 설계방법은 어떤 2계층 MLP (multi-layer perceptron)라도 파이프라이닝 기술을 사용하여 은닉층의 뉴런수와 같은 주기로써 수행이 완료되는 설계방법이다. 제안된 설계방법의 전체구조는 입력층과 은닉층에서는 링크병렬화 기법을 사용하며, 은닉층과 출력층에서는 개선된 뉴런병렬화 방법을 제안하여 어떤 2계층 MLP를 쉽게 전체 파이프라이닝 구조로 설계할 수 있는 방법이다. 제안된 설계방법을 이용하여 2계층 MLP를 적용하였을 때, 기존의 CPU에서 보다 11배 정도 빠른 성능을 보인다.

2. 하드웨어 구현

본 2 장에서는 소프트웨어 신경망을 쉽게 하드웨어 환경에 적응하여 사용할 수 있는 설계방법을 기술한다. 2.1 절은 전체 시스템 구조에 대해서 설명하며, 2.2 절은 입력층과 은닉층 사이에서의 설계방법 및 2.3 절은 은닉층과 출력층 사이에서의 설계방법을 기술한다.

2.1 전체 시스템 구조

전체 파이프라인 구조에 앞서 일반적인 2계층 신경망의 구조 및 처리과정에 대해서 기술한다. 본 논문에서는 문자 분석기로써 2계층의 MLP를 사용하며, 41개의 입력값과 33개의 은닉층 뉴런 그리고 2개의 출력층 뉴런으로 구성된다 (그림 1).

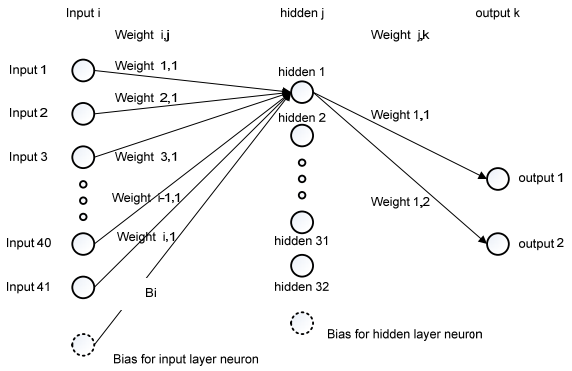

그림 1. MLP 의 처리과정

전체 신경망 구조는 링크 병렬화와 수정된 뉴런 병렬화 방법을 사용한 파이프라이닝 구조이다 (그림 2). 기본 연산은 부동소수점의 곱셈과 덧셈연산을 사용하며, 부동소수점 곱셈과 덧셈 연산은 느린 처리속도로 인해서 일반적으로 몇 단계로 나누어 처리되는 파이프라인 기법을 사용한다.

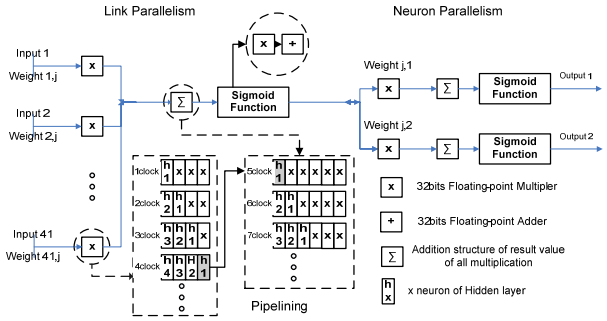

그림 2. 전체 MLP 의 하드웨어 구조

부동 소수점은 IEEE 표준[7]으로써 부동소수점 수를 하나의 부호 비트, 바이어스 된 지수(biased exponent)의 8 비트, 그리고 정규화된 가수(mantissa) 또는 유효수(significand)의 23 비트로 표현한다. 부동소수점 수를 사용하는 덧셈은 6 단계,

곱셈은 5 단계로 구성된 파이프라인 구조이다 [8]. 활성화 함수 $a(x)$ 는 가중치와 입력값을 가지고 덧셈연산과 곱셈연산의 결과값을 수행하여 결과값을 출력한다. 여기서 활성화 함수로 다음과 같은 시그모이드 함수를 사용한다.

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

시그모이드 함수는 지수연산과 덧셈연산 그리고 곱셈연산이 사용되는데 이 연산들은 많은 수행시간과 면적을 소요한다. 이 문제점을 해결하기 위해서 Piece Wise Linear(PWL)방식으로 구현하였다 [9]. PWL 방식은 비선형으로 이루어진 함수를 여러 개의 선형 함수로 처리하는 방식으로써 덧셈연산과 곱셈연산만으로 비선형 함수를 처리할 수 있다. 본 논문에서는 간단한 구현을 위해서 3 segment PWL $(y_3(x)=1$, $y_2(x)=(1/2+x/4)$, $y_1(x)=0)$ 방식으로 시그모이드 함수를 구현하였으며, 파이프라인 구조의 덧셈연산과 곱셈연산을 사용함으로써 시그모이드 함수 역시 파이프라인구조로 수행한다.

2.2 입력층과 은닉층사이의 링크병렬화 설계

입력층과 은닉층 사이에서는 파이프라이닝 구조를 위해서 링크 병렬화 방법을 사용하며, 파이프라이닝 구조로써 1 클럭에 하나의 은닉층 뉴런이 처리된다. 링크 병렬화 구조는 입력수와 같은 41개의 곱셈연산 및 덧셈연산 그리고 하나의 시그모이드 연산, 그리고 가중치 값을 저장하고 있는 롬(ROM)으로 구성된다 (그림 3). 여기서 덧셈 연산의 구조는 입력 층 뉴런의 수에 따라 결정되는데, 입력 층의 뉴런 수(바이어스 포함)를 i 라고 할 때 $i=2^n$ 이면 덧셈 연산의 구조(즉, 덧셈 연산의 층)는 n의 층으로 이루어 진다. 또한 $2^{n-1}<i<2^n$ 일 때 덧셈 연산의 구조는 n 층으로 이루어진다. 그림 3은 입력 수 42개일 때의 구조이며 $2^5<42<2^6$ 으로써 덧셈연산은 6층으로 구성된다. 이때 전체 파이프라이닝 구조를 위해서 $i=2^n$ 이면 버퍼가 필요 없지만, $2^{n-1}<i<2^n$ 이면 버퍼가 필요하다. 전체 파이프라이닝 처리를 위해서 처리되지 않은 연산자의 결과값을 다음 단계에서 처리하기 위해서 저장이 필요하기 때문이다.

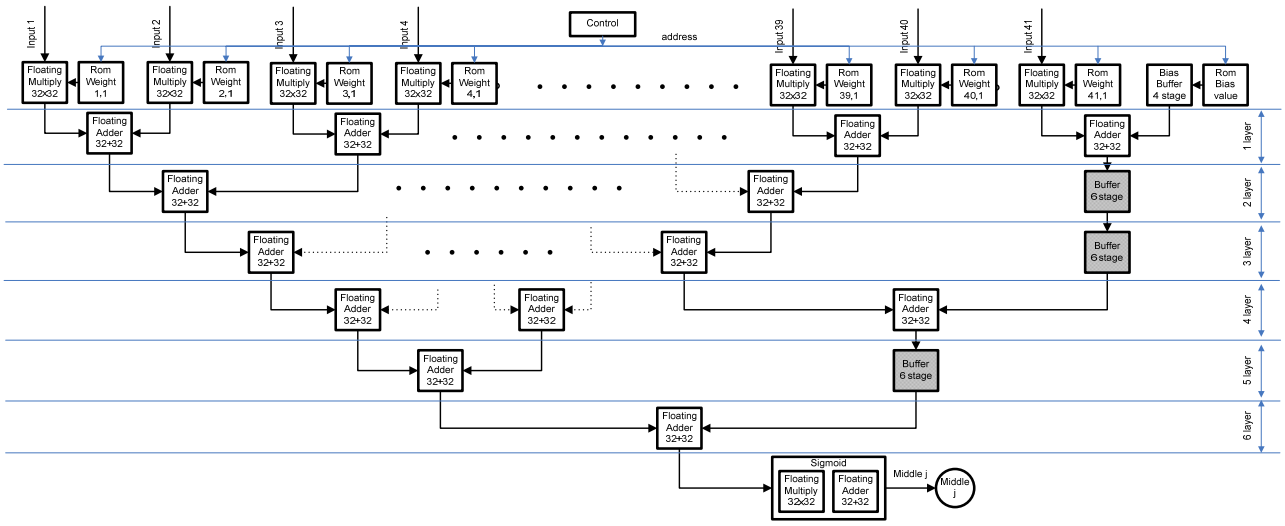

그림 3. 41 개의 입력값의 하드웨어 구조

2.3 출력층의 수정된 뉴런 병렬화 설계

부동소수점 연산을 사용하면서도 전체 파이프라이닝 설계방법을 제공하기 위해서 기존의 뉴런 병렬화 방법을 개선하여 사용한다. 그림 4는 수정된 뉴런 병렬화 구조와 링크 병렬화 구조를 비교한

그림이다. 링크 병렬화 구조에서 곱셈 연산을 수행하기 위해서는 입력수 만큼의 곱셈 연산이 사용되지만, 뉴런 병렬화 구조는 입력되는 은닉층 뉴런의 결과값을 하나의 곱셈연산으로 처리하며, 링크 병렬화 방법에서 덧셈연산의 각층은 하나의 덧셈 연산을 이용하여 수행한다. 각 덧셈연산은 곱셈연산 및 덧셈연산에서 출력되는 결과값이 2개가 될 때까지 저장할 버퍼가 필요하다. 이러한 과정으로 개선된 뉴런 병렬화 방법은 수행되며, 최종 결과값은 은닉층 뉴런의 수만큼의 주기로 최종 결과를 출력한다. 그림 4와 같이 하나의 출력층 뉴런 구조를 나타내며, 출력층 뉴런들은 은닉층 뉴런수 만큼 반복 수행되었을 때 완료된다.

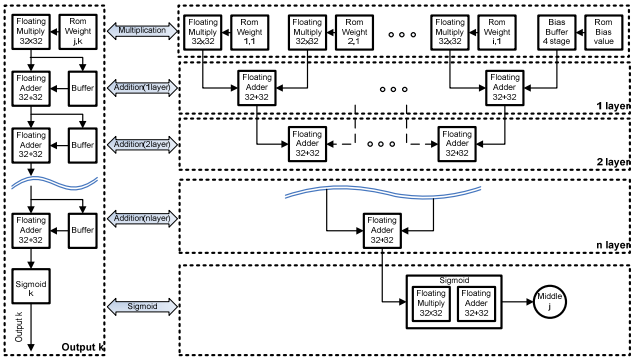

그림 4. 수정된 뉴런 병렬화의 구조

그림 5는 구현된 수정된 뉴런 병렬화 구조를 나타낸다. 출력층에서 필요한 연산자의 수를 살펴보면 곱셈 연산과 시그모이드 연산이 출력층 뉴런 수만큼 필요하며, 덧셈연산은 은닉층의 뉴런의 수와 바이어스를 포함한 총수를 $j+1$ 이라고할 때 $n=\lceil \log_2^{j+1} \rceil$ 이면 덧셈연산의 수는 $n \times k$ 개이다. 이때 k 는 출력층의 뉴런수이다.

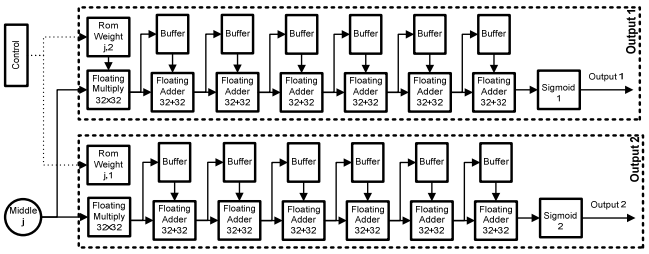

그림 5. 출력층의 수정된 뉴런 병렬화 구조.

3 실험결과

본 논문에서는 자이링스(Xilinx)사의 XC2V8000 의 FPGA Chip 상에서 VHDL 의 합성을 위해서 Xilinx ISE 7.1 를 사용하여 전체 파이프라이닝 구조로 설계하였다. 표 1 은 전체 파이프라이닝 신경망을 구현할 때 사용된 기본 연산의 성능을 나타낸다.

[표 1] 연산자의 성능분석

연산자	면적 (gate)	동작속도 (MHz)	클럭수 (clk)	수행시간(ns)
Adder	9,239	90.917	6	65.994
Multiplier	20,655	81.274	5	61.520
Sigmoid	29,941	80.658	11	136.378

본 시스템에서 제안하는 설계방법을 테스트 하기 위해서 소프트웨어 에서 사용된 2계층 MLP를 적용하여 수행하였다. 2계층 MLP는 41개의 입력과 33개의 은닉층 뉴런 그리고 2개의 출력층 뉴런을 가진다. 표 2는 MLP의 첫 수행을 분석한 결과이다.

[표 2] 신경망의 성능분석

MLP	면적 (gate)	동작속도 (MHz)	1st 클럭수 (clk)	수행시간 (ns)
MLP	1,426,312	54.259	111	2045.73

표 2 는 MLP 의 첫 수행을 나타내며, 첫 수행시 파이프라이닝 기법이 적용되지 않기 때문에 수행시간이 많이 소요된다. 하지만 표 3 에서는 파이프라이닝 기법에 적용 되었을 때의 성능분석이다.

[표 3] 신경망의 순차 처리분석

MLP	2nd 수행 (clk)	3rd 수행 (clk)	…	1722120th 수행 (clk)	수행 간격 (ns)
MLP	33	33	33	33	589.76

본 논문에서 제안된 전체 파이프라이닝 설계방법으로 설계하였을 때 은닉층 뉴런수와 같은 클럭주기로 처리됨을 볼 수 있다.

(a) (b) (c)

그림 6. 실험 결과: (a) 테스트 이미지, (b) CPU 처리결과, 그리고 (c) FPGA 처리결과

다음은 소프트웨어 신경망을 CPU 에서 처리한 결과와 하드웨어에 적용하여 처리한 결과를 비교 분석하였다 [표 4]. 이때 사용된 컴퓨터는 512 Mbyte 메모리와 2.4GHz 펜티엄 4 CPU 를 사용하였다. 이때 사용된 이미지는 1152×1546 크기로써 1722129 번의 번복 처리로써 신경망을 이용하여 문자 추출을 수행한다. 이때 CPU 와 FPGA 에서의 처리된 결과는 그림 6 과 같다.

[표 4] CPU 와 FPGA 의 성능 분석

FPGA	CPU
1.04(s)	11.73(s)

본 논문에서 제안된 설계방법은 CPU 를 사용한 소프트웨어 MLP 의 수행 성능보다 약 11 배 정도 빠른 성능을 보인다.

4. 결론

소프트웨어에서 학습된 가중치를 고정 소수점 연산을 사용하는 하드웨어에 쉽게 적용하기 어려운 점과 소프트웨어에서의 학습된 가중치를 쉽게 적용하기 위해서는 느린 처리속도를 가지는 부동소수점 연산을 사용해야 하는데 이때 어떻게 느린 처리 속도를 해결할 것인가가 문제이다. 본 논문에서는 입력층과 은닉층에서의 링크 병렬화 방법과 은닉층과 출력층에서는 개선된 뉴런 병렬화 방법을 제안하여 전체 MLP의 파이프라이닝으로 쉽게 설계하는 방법이다. 제안된 설계방법은 은닉층 뉴런 수만큼의 클럭 소요로 한번의 신경망이 처리한다. 여기서 문자 추출을 위해 사용된 신경망을 적용하였을 때 느린 처리속도를 가지는 부동소수점 연산을 사용하면서도 11배나 빠른 수행을 가짐을 알 수 있다. 따라서 영상처리 및 패턴인식과 같은 많은 반복으로 처리되는 분야에서 높은 정확도와 높은 처리속도로써 해결할 수 있는 설계방법이다.

참고문헌

[1] 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.

[2] Keechul Jung, Kwang In Kim, and Anil K. Jain, “Text Information Extraction in Image and Video: A Survey,” International Journal of Pattern Recognition, In Press.

[3]Watkins S.S, Chau P.M, “Reduced-complexity Circuit for Neural Networks,” Electronics Letters Volume 31, Issue 19, pp. 1644-1646, 1995.

[4]Wust H, Kasper K, Reininger H, “Hybrid number representation for the FPGA-realization of a versatile neuro-processor,” Euromicro Conference, 1998. Proceedings. 24th , Vol. 2, pp. 694-701, 1998.

[5] Henritte Ossoinig, Erwin Reisinger, Christian Steger, Reinhold Weiss, “Design and FPGA-Implementation of a Neural Network,” Proc. 7th Int. Conf. on. Signal Processing Applications & Technology, pp. 939-943, Boston,. USA, October 1996.

[6] 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.

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

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

[9] K. Basterretxea, J.M. Tarela and I. del Campo, “Approximation of sigmoid function and the derivative for hardware implementation of artificial neurons”, IEE proc., Circuits Syst., Vol. 151, No. 1, February 2004.