ISSN : 1226-7244 (Print)
ISSN : 2288-243X (Online)
논문번호 20-01-30

j.inst.Korean.electr.electron.eng.Vol.24,No.1,225~231,March 2020
http://dx.doi.org/10.7471/ikeee.2020.24.1.225

한정된 자원을 갖는 FPGA에서의 이진가중치 신경망 가속처리 구조 설계 및 구현

Design and Implementation of Accelerator Architecture for Binary Weight Network on FPGA with Limited Resources

김 종 현*, 윤 상 균**

Jong-Hyun Kim* and SangKyun Yun**★

Abstract

In this paper, we propose a method to accelerate BWN based on FPGA with limited resources for embedded system. Because of the limited number of logic elements available, a single computing unit capable of handling Conv-layer, FC-layer of various sizes must be designed and reused. Also, if the input feature map can not be parallel processed at one time, the output must be calculated by reading the inputs several times. Since the number of available BRAM modules is limited, the number of data bits in the BWN accelerator must be minimized. The image classification processing time of the BWN accelerator is superior when compared with a embedded CPU and is faster than a desktop PC and 50% slower than a GPU system. Since the BWN accelerator uses a slow clock of 50MHz, it can be seen that the BWN accelerator is advantageous in performance versus power.

요 약

본 연구에서는 임베디드 시스템에 적용하기 위해 자원이 제한된 조건의 FPGA를 기반으로 BWN 가속처리를 하는 방법을 제시하였다. 사용할 수 있는 로직의 개수가 제한적이기 때문에 다양한 크기의 Conv-layer, FC-layer를 처리할 수 있는 하나의 연산장치를 설계해서 재활용하였다. Input feature map 데이터를 한번에 병렬처리를 할 수 없는 경우 데이터를 여러 번 읽어서 중간결과를계산하고 합산하여 최종 출력을 계산하였다. 사용할 수 있는 BRAM 모듈 개수가 제한적이기 때문에 BWN 가속기내의 데이터 bit수를 최소화한 구조를 사용하였다. 구현한 BWN가속기의 이미지 분류 처리 시간은 소형 시스템과 비교하였을 때 처리시간 측면에서 우수함을 보였고 고성능 시스템과 비교하였을 때는 데스크탑 PC보다는 빠르고 높은 클럭속도의 GPU시스템의 50%정도 느렸다. BWN가속기는 50MHz의 느린 clock을 사용하므로 성능대비 전력측면에서 유리함을 확인할 수 있었다.

Key words : Binary Weight Network, Low precision network, FPGA acceleration, SoC-FPGA, FPGA

* Telechips Inc.
** Department of Computer and Telecomm. Engineering, Yonsei University, Wonju
★ Corresponding author
E-mail : skyun@yonsei.ac.kr Tel : +82-33-760-2267
Manuscript received Mar. 6, 2020; revised Mar. 18, 2020; accepted Mar. 23, 2020.

Ⅰ. 서론

최근에 인공지능 분야에서 각광을 받고 있는 심층신경망은 컴퓨터 비전과 음성인식 등 다양한 분야에서 성능이 입증되고 있으며 다양한 심층신경망 중에서 CNN(Convolutional Neural Network)이 컴퓨터 비전 분야에서 가장 널리 사용되고 있다[1, 2].

CNN을 계산하기 위해서는 시간이 많이 소요되는 많은 양의 부동소수점 연산이 필요하기 때문에 가속처리에 대한 연구가 여러 연구자들에 의해서

진행되었다. GPU를 사용한 가속처리가 널리 사용되고 있지만, GPU는 전력을 많이 소모하는 단점이 있으며, GPU 기반의 그래픽 장치가 없는 소형 임베디드 시스템에서는 사용할 수 없다.

저전력 소모를 위한 대안으로 FPGA 또는 ASIC 기반으로 CNN 계산을 가속처리하는 연구도 진행되었다[3-5]. FPGA는 GPU보다 최대 클록 속도는 느리지만, 전력을 적게 소모하고 병렬성을 크게 할 수 있으며, ASIC과는 달리 신경망 구조에 따라서 재구성하여 사용할 수 있는 장점이 있다. 그리고 최근에 Xilinx와 Intel 등에서 CPU와 FPGA를 함께 내장한 SoC를 보급함에 따라서 소형 임베디드 시스템에서 FPGA를 손쉽게 사용할 수 있는 환경이 갖추어졌다.

CNN을 수행하는데 많은 수의 부동소숫점 연산뿐 만 아니라, 계산에 사용되는 weight 등의 파라메터와 계산된 데이터를 저장하기 위한 메모리 요구량이 매우 크다. 대표적인 CNN인 AlexNet[1]은 약 240MB의 부동소수점 파라메터가 필요하다. 그렇지만 이러한 CNN은 FPGA 내부의 블록 메모리(BRAM) 만을 사용하여 구현하기가 쉽지 않다.

CNN 계산에서의 메모리 요구량과 계산량을 줄이기 위해서 CNN에서 사용하는 파라메터와 데이터의 정밀도를 줄이는 저정밀도 신경망에 대한 연구가 진행되었다. 32비트 부동소수점 수를 적은 비트 수의 고정소수점 수로 양자화시키는 방법들이 주로 연구되었으며[6,7], 특히 가장 극단적인 형태로 weight를 +1과 -1의 값으로 이진화한 이진 신경망[8]이 제안되었다. weight뿐만 아니라 데이터도 이진화한 BNN(Binarized Neural Network)은 MNIST와 같은 작은 이미지에 대해서 정확도 손실없이 성능이 우수함을 보였으며[9, 10], weight만 이진화한 BWN(Binary Weight Network)은 비교적 큰 이미지인 ImageNet에 대해서도 높은 정확도를 보였다[11].

CNN 계산을 FPGA로 구현할 때에 메모리 요구량 감축과 성능 향상을 위해서 정확도의 손실을 크지 않은 범위에서 저정밀도 신경망을 채택하는 것이 일반적이며, 특히 FPGA 기반으로 BNN 계산을 가속시키는 연구[12-14]와 BWN 계산을 가속시키는 연구[15]가 최근에 수행되었다. FPGA로 구현한 기존 연구는 하드웨어 자원이 풍부한 최상위의 FPGA에서 설계되어서 소규모 FPGA 설계에 적용하기가 어려우며, BWN 계산을 가속시키는 YodaNN[15]은 ASIC으로 구현된 것으로 컨볼루션(Conv) 계층 연산만 처리하고 Fully-connected(FC) 계층 연산에 대한 기능은 없다.

본 연구에서는 SoC FPGA를 탑재한 소형 임베디드 시스템에서 weight만 이진화하는 BWN 구조를 사용하여 CNN 계산을 가속화하는 소형 FPGA에 적합한 구조를 설계하고 구현하였다. 구현 대상 신경망으로 BWN에 대한 기존 연구[11]에서 사용했던 CIFAR-10 데이터[16]를 처리하는 CNN을 사용했으며 Cyclone V SoC-FPGA를 탑재한 DE1-SoC 보드에서 구현하였으며, SoC의 프로세서에서 구현한 하드웨어를 사용하기 위한 호스트 인터페이스와 소프트웨어도 함께 구현하였다.

II. 관련 연구 및 연구 동기

2.1 컨볼류션 신경망(CNN)

전형적인 CNN 구조는 Convolution(Conv) 계층이라고 하는 몇 개의 기본 단계들로 구성되어 있으며, 각 기본 단계는 컨볼루션, 정규화, 비선형 활성화와 max-pooling 연산으로 구성되어 있으며 각 단계는 특징 추출기능을 수행한다. 그리고 Conv계층들 다음에 Fully-Connected (FC) 계층들이 연속하여 이어져서 분류 작업을 수행하여 결과를 제공한다. 이러한 심층 CNN 모델의 구조는 그림 1과 같이 나타낼 수 있다.

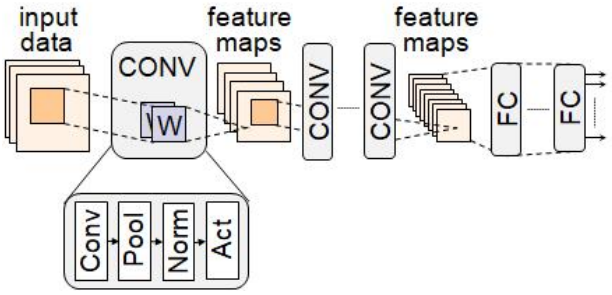


Fig. 1. Typical Deep CNN model structure.
그림 1. 전형적인 심층 CNN 모델 구조

2.2 BWN의 고정소수점 연산 정확도

CNN의 하드웨어 구현을 위해서는 정확도 손실이 크지 않다면 원래의 신경망 대신에 이진신경망과 같은 저정밀도 신경망을 채택하는 것이 하드웨어 복잡도가 줄어드는 장점이 있다. 신경망 중에서 weight만 이진화한 신경망을 BWN(Binary Weight

Network)이라고 하며 weight와 데이터 모두 이진화한 BNN에 비해서 더 좋은 정확도를 보인다. BWN을 하드웨어로 구현할 때에 부동소수점 연산이 복잡하기 때문에 데이터를 고정소수점으로 표현하여 고정소수점 연산을 한다면 하드웨어 복잡도를 더 감소시킬 수 있다.

본 연구의 사전 연구로 CIFAR-10을 처리하는 BWN인 표 1과 같은 구조를 갖는 ConvNet에 대해서 고정소수점 연산 기반으로 처리할 때의 연산 정확도를 분석하였다[17]. 이 표에서 Input feature map의 depth는 D, width는 W_{in}, output feature map의 depth는 N, width는 W_{out}로 나타내었다. 이 연구에서 부동소수점 연산을 사용한 BWN에서는 정확도가 85.7%이며, activation 데이터를 소수점이하 3비트인 8비트 고정소수점 수로 나타내고 중간 계산값은 16비트로 나타내었을 때에 고정소수점 연산을 사용한 정확도가 85.1%임을 보였다. 이처럼 activation 데이터를 적절한 형식의 고정소수점 수로 나타내고, 중간 계산 결과의 손실을 최소화할 수 있는 크기를 사용하고 오버플로를 적절히 처리한다면 고정소수점 연산을 적용하여도 부동소수점 연산과 비교하여 정확도 손실이 크지 않음을 보여주었다.

Table 1. ConvNet for BWN implementation on FPGA.
표 1. FPGA에서의 BWN 구현을 위한 ConvNet

Layer	D	W_{in}	W_{out}	N	input	weight
Conv1	3	32	32	128	3K	384×9
Conv2	128	32	32	128	128K	16K×9
Pool2	128	32	16	128	-	-
Conv3	128	16	16	256	64K	32K×9
Conv4	256	16	16	256	64K	64K×9
Pool4	256	16	8	256	-	-
Conv5	256	8	8	512	16K	128K×9
Conv6	512	8	8	512	32K	256K×9
Pool	512	8	4	512	-	-
FC1	8192	1	1	1024	8K	8M
FC2	1024	1	1	1024	1K	1M
FC3	1024	1	1	10	1K	10K

본 연구에서는 이진신경망 BWN인 ConvNet에 대해서 고정소수점 연산기반으로 처리하는 가속기 구조를 설계하고 FPGA로 구현하였다. 그리고 BWN에 대해서 Conv 계층 뿐 만 아니라 FC 계층도 함께 지원하는 하드웨어 구조를 제시하고, 소형 임베디드 시스템에서는 고성능 FPGA 대신에 자원 제약이 있는 FPGA를 사용하므로 이러한 FPGA에서 구현 가능한 구조를 제안하였다. BWN의 구현은 Cyclone V SoC-FPGA를 탑재한 DE1-SoC 보드를 사용하여 수행하였으며 BWN 하드웨어를 프로세서에서 사용하기 위한 호스트 인터페이스와 소프트웨어도 함께 구현하였다.

III. 이진 신경망 처리 가속기 구조

3.1 전체 구조

구현 대상 시스템인 DE1-SoC에서 사용하는 Cyclone V는 저용량 FPGA와 ARM Cortex A9기반의 SoC를 내장한 SoC-FPGA로서 397개의 1K×10 블록 메모리(BRAM)인 M10K와 768개의 32×2 메모리로 사용가능한 MLAB를 내장하여 총 4,450K 비트의 메모리를 제공한다.

DE1-SoC에 BWN 구현을 위한 설계를 할 때에 weight 크기로 1비트, feature map 데이터 크기로 8비트로 사용하면 1개의 BRAM은 한개의 32×32 feature map 또는 4개의 16×16 feature map을 저장할 수 있으며, 메모리의 각 주소에 3×3 weight (9-bit)를 저장한다면 1K개의 BWN weight 행렬을 저장할 수 있다.

BRAM들은 각 계층의 입력 데이터, 출력 데이터와 weight, parameter 저장에 사용된다. 두 그룹의 128개의 BRAM을 입력 데이터와 출력 데이터저장용으로, 128개의 BRAM을 weight 저장용으로 사용하면 총 384개로서 이 FPGA에서 구현 가능하며, 128개의 데이터 입력과 weight를 동시에 읽어서 병렬처리할 수 있다.

Ⅱ절의 표 1에 각 Conv계층과 FC 계층의 입력 데이터 개수와 weight 개수도 함께 적었다. 각 계층의 출력 데이터 개수는 다음 계층의 입력 데이터 개수와 같다. 입력 데이터 개수는 두 번째의 Conv2 계층이 128KB로 가장 많으며 다음 계층으로 갈수록 max pooling 연산으로 인하여 개수가 감소한다. weight 개수는 입력과 출력 개수가 증가함에 따라서 증가하여 FC1 계층에서 8MB로 최대가 된다.

최대 입력 데이터 용량인 128KB는 128개의 1KB

BRAM에 저장할 수 있으므로 ConvNet의 한 계층의 입력을 모두 저장할 수 있다. 데이터용 BRAM은 현재 계층의 출력이 다음 계층의 입력이 되므로 계층이 바뀔 때에 입력과 출력을 바꾸어 사용하면 두 그룹의 BRAM을 사용하여 모든 계층의 연산을 수행할 수 있다.

그리고 128개의 weight용 BRAM은 총용량이 128K×10b로서 표 1에서 Conv6, FC1, FC2 계층에 대한 weight를 모두 저장할 수 없다. 이에 따라서 weight는 FPGA 외부의 SDRAM에 모든 weight를 저장한 후에 각 단계마다 계산에 필요한 weight를 BRAM에 적재한 후에 사용하도록 하였다. 외부 SDRAM에서 128개의 weight용 BRAM에 적재하는 것은 순차 수행되므로 성능에 제약이 될 수 있다.

이렇게 설계한 BWN 가속기의 기본 구조는 그림 2와 같다. BWN 엔진은 핵심 처리장치로서 두 그룹 중 한 BRAM에서 입력받고, 다른 BRAM으로 출력을 저장하도록 되어 있다.

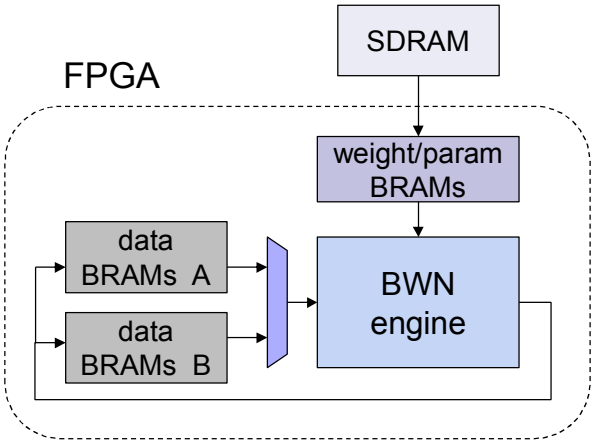


Fig. 2. Basic Architecture of BWN Accelerator.
그림 2. BWN 가속기의 기본 구조

BWN 엔진은 한 BRAM에서 입력을 병렬로 읽어서 컨볼루션, max pooling, batch normalization 및 activation 연산을 수행하고 출력을 다른 BRAM에 저장한다.

3.2 기본 연산 장치 PE

BWN 엔진에는 Conv계층의 컨볼루션 연산과 FC계층의 행렬곱 연산을 수행하는 기본 연산장치인 PE(Processing Element)가 포함되어 있다.

기본 연산장치 PE의 구조는 그림 3과 같은 systolic 구조로 되어 있다. PE는 Conv 계층에서 데이터와 3×3 weight와의 컨볼류션 연산을, FC 계층에서는 데이터와 weight와의 행렬곱 연산에 사용되는 벡터내적 연산을 수행한다.

Conv 계층에서 컨볼루션 연산을 할 때에 같은 BRAM에 저장되어 있는 입력 데이터가 연속적으로 공급되면 해당 데이터가 포함되는 컨볼류션들에 필요한 연산을 계산하여 저장하고, weight의 각 행의 마지막인 w[0,2]와 w[1,2]와 계산한 결과는 shift register에 저장되어 다음 행의 관련 데이터가 입력할 때까지 지연시킨 후 컨볼루션 계산을 누적하여 계속하도록 한다. 마지막 weight인 w[2,2]와 계산한 후에는 컨볼루션 계산 결과를 출력한다.

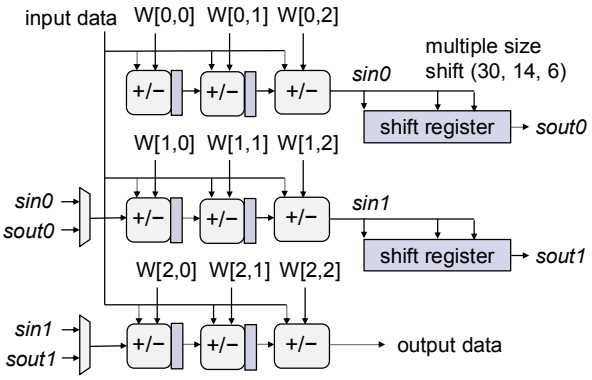


Fig. 3. Archtectire of PE.
그림 3. PE의 구조

Conv 계층들에서 feature map 크기는 32×32, 16×16, 8×8의 세 종류가 있으므로 30, 14, 6 크기를 지원하는 shift register를 구현하였다. 그리고 컨볼루션 계산을 할 때에 제어신호에 따라서 zero padding도 가능하도록 설계하였다.

32×32 feature map에 대하여 둘째 행의 셋째 열 데이터가 입력될 때에 첫째 컨볼루션 결과가 출력되기 시작하여 파이프라인 방식으로 매 클럭마다 컨볼루션 결과를 출력한다.

FC 계층에서는 행렬곱이 벡터 내적의 반복으로 이루어지는 데 PE는 같은 연산장치를 사용하여 벡터 내적을 구성하는 8개의 입력 데이터와 8개의 이진 weight와의 부분 벡터 내적 계산을 수행한다. 이때 결과 값은 컨볼루션과는 다르게 8개 의 입력 데이터를 받을 때마다, 즉 8 클럭마다 출력되며 PE에서 FC 계층에 대한 계산을 수행할 때 각 행의 계산 결과는 shift register로 입력되는 것 대신에 다음 행의 가감산기 입력으로 직접 연결된다.

3.3 BWN 엔진

BWN 엔진에는 동시에 입력되는 데이터를 동시에 병렬 처리할 수 있도록 그림 4와 같이 PE (Processing Element)가 병렬로 배치되어 있다. 각 PE의 출력은 입력 데이터에 대한 콘볼루션이나 벡터내적을 계산하기 위하여 adder tree를 거쳐 합산된다. BWN 엔진은 한 사이클에 최대 한 개의 출력을 계산하며, 입력 개수가 PE개수보다 많은 경우에는 입력을 여러 번 읽어서 계산을 반복하고 중간 결과는 data buffer RAM에 저장하여 다음 결과와 합산한다. 모든 입력을 처리하여 결과가 얻어지면 max pooling, batch normalization (BN)과 activation을 수행한 후에 최종 출력을 다른 그룹의 BRAM에 해당위치에 저장한다.

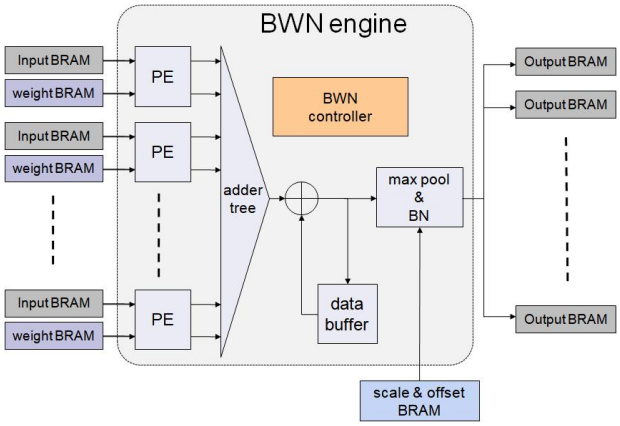


Fig. 4. Input Parallel BWN Engine Architecture.
그림 4. Input Parallel BWN 엔진 구조

DE1-SoC의 FPGA에서는 자원 제약으로 BWN을 구성하는 여러 계층들을 계산하는 장치를 동시에 배치할 수 없다. 여러 계층을 동시에 구현할 수 있는 경우에 파이프라인 설계를 한다고 해도 각 계층마다 계산량이 다르기 때문에 성능향상에 제약이 있다. 제안하는 설계에서는 BWN엔진은 한 계층의 연산을 가능한 한 최대로 병렬 수행하도록 설계하며, 하나의 BWN 엔진을 모든 계층의 연산에 반복하여 사용하는 구조를 채택하였다.

이를 위해 III-2절에서 소개한 바와 같이 그림 3의 PE는 모든 계층에서의 연산을 수행할 수 있도록 Conv-layer, FC-layer와 다양한 크기의 input feature map, output feature map에 대해서 계산을 수행할 수 있게 하였으며, 그림 2의 BWN 가속기 구조에서 한 계층의 연산이 끝날 때마다 BRAM A와 BRAM B가 BWN 엔진의 입력과 출력 역할을 교대하게 하여 이전 계층 출력이 자동적으로 다음 계층의 입력이 되게 했다.

그리고 그림 4의 BWN 제어기는 모든 동작에 필요한 제어신호를 적절한 타이밍에 공급하도록 FSM (Finite State Maching) 기반으로 설계되었다.

Ⅳ. 구현 및 성능 평가

4.1 개발환경과 구현

본 연구에서 사용한 FPGA 개발 보드는 Altera의 Cyclone V SoC FPGA를 사용한 DE1-SoC 보드이며 Cyclone V에는 ARM Cortex-A9 MPcore 프로세서가 내장되어 있다. 그리고 Quartus Prime 17.0 환경에서 Verilog를 사용해서 개발하였다.

본 연구에서 사용한 학습/테스트용 데이터는 CIFAR-10 data set이며 사전연구[17]를 통하여 BWN에 대해서 학습하여 이진 weight와 고정소수점 형식의 BN parameter를 얻어서 실제 구현에 사용하였다. SoC 내의 호스트에서 FPGA로 구현된 BWN가속기에 이미지를 전송하고, 분류 결과를 받도록 시스템을 구성하였다. 호스트에서는 BWN연산에 필요한 parameter들을 FPGA 외부의 SDRAM에 미리 적재하고, 분류하고자 하는 CIFAR-10 이미지의 전처리 과정을 수행한 후 BWN 가속기에 전처리 결과를 전송하고 처리가 종료되면 분류 결과를 받는다. 이런 방법으로 이미지를 한 개씩 전송하여 분류를 수행할 수 있다.

4.2 자원과 성능 평가

표 2는 병렬성 크기를 8부터 128까지 변화를 주면서 Cyclone V의 사용된 자원을 나타낸 것이다. 실험결과 주어진 자원 내에서 병렬성은 128까지 가능하였다.

Table 2. Resource Amount vs. Parallism Degree
표 2. 병렬성 크기에 따른 자원 사용량

Parallel Degree	ALM	register	BRAM
	total=32,070	total=128,300	total=3970(Kbits)
8	3,987(12%)	7,342(6%)	1,388(35%)
16	5,681(18%)	11,040(9%)	1,460(37%)
32	8,954(28%)	18,283(14%)	1,604(40%)
64	15,726(49%)	32,912(26%)	2,150(54%)
128	28,095(88%)	61,874(48%)	3,236(82%)

표 3은 BWN가속기와 다른 시스템과의 CIFAR-10 이미지 분류 시간을 비교한 것이다. 비교대상 시스템은 다음과 같다. CPU 시스템은 Intel의 i7-3700 CPU이고, ARM 시스템은 DE1-SoC내에 있는 임베디드 CPU이며, GPU 시스템은 NVIDIA의 GeForce GTX 760이다. CPU와 ARM 시스템에서는 32비트 부동소수점 연산을 사용해서 수행했다. BWN 가속기는 50MHz의 클록을 사용하여 980 MHz 클럭을 사용하고 1152 CUDA코어를 가진 GPU의 반 정도의 성능을 보였으며 GPU를 사용하지 않는 CPU 시스템보다는 10배 이상 빠르게 동작하였다.

Table 3. Performance Comparison vs CPU/GPU/FPGA.
표 3. 성능 비교

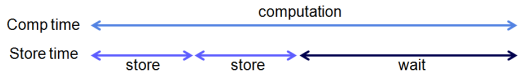
System	ARM	CPU	GPU	BWN accelerator
Feature	-O3 optimize	-O3 optimize	1152 CUDA core	128 parallel
Clock	800MHz	3.4GHz	980MHz	50MHz
Conv(sec)	4.8699	0.2314	-	0.0152
FC(sec)	0.1327	0.0307	-	0.0083
Total(sec)	5.0027	0.2621	0.0110	0.0235
Speedup	**1.0x**	**19.1x**	**454.8x**	**212.9x**
acc	85.7	85.7	85.7	85.1

표 4는 병렬성 크기에 따른 Conv 계층과, FC 계층의 처리시간 변화이다. Conv계층에서는 병렬성에 거의 비례하여 처리시간이 감소하는 것을 알 수 있다. 그렇지만 FC 계층에서는 병렬성이 늘어날수록 처리시간이 비례해서 감소하지 않는다. 특히 64병렬 부터는 처리시간 감소량이 크게 떨어진다. 비례해서 감소하지 않는 이유는 FC 계층에서는 weight 적재시간에 비해서 계산시간이 짧아서 실행시간이 weight 적재시간에 의해 영향을 받기 때문이다.

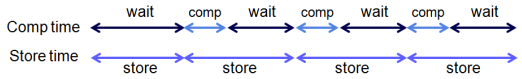
Table 4. Execution Time of BWN accelerator for parallelism degree (sec).
표 4. 병렬성에 따른 BWN가속기 처리시간 (초)

Parallel Degree	Conv	FC	Total
8	0.1949	0.0445	0.2384
16	0.0990	0.0238	0.1229
32	0.0513	0.0127	0.0641
64	0.0273	0.0089	0.0362
128	0.0152	0.0083	0.0235

현재 계층의 계산을 하는 동안 다음 계층을 위한 weight가 적재된다. 그림 5는 Conv 계층과 FC 계층에서 계산시간과 weight 적재시간의 관계를 나타낸다. Conv 계층에서는 계산시간이 많고 weight 적재시간이 적어서 계산시간이 실행시간이 된다. 이에 비해서 FC 계층은 weight 양이 많고 계산이 적어서 weight 적재시간이 계산시간보다 길어서 실행시간이 적재시간에 의해서 정해진다.



(a) Conv Layer : computation time > load time



(b) FC layer : computation time < load time

Fig 5. Computation time vs. Weight Load Time.
그림 5. 계산시간과 weight 적재시간

V. 결론

본 연구에서는 임베디드 시스템에 적용하기 위해 자원이 제한된 FPGA를 기반으로 BWN 가속처리를 하는 방법을 제시하고 설계, 구현하였다. 사용할 수 있는 로직의 개수가 제한적이기 때문에 다양한 크기의 Conv 계층과 FC 계층을 처리할 수 있는 하나의 연산장치를 설계해서 각 계층 연산에 재사용하였다. 또한 여러 개의 Input feature map 데이터를 동시에 입력받아서 병렬처리를 하여 행렬곱과 컨볼루션을 수행하였다.

BWN가속기의 CIFAR-10 이미지 분류 처리시간은 소형 시스템과 비교하였을 때 성능이 상당히 우수했다. 그리고 고성능 시스템과 비교하였을 때는 일반 데스크탑 시스템보다는 빨랐지만 GPU 시스템보다는 반 정도 느렸다. 그렇지만 BWN가속기는 비교 대상 시스템의 클럭 980MHz보다 훨씬 느린 50MHz 클럭을 사용하여 얻은 결과이므로, 성능 대비 전력소모 측면에서 훨씬 유리함을 확인하였다.

본 연구에서 제안한 구조는 자원이 충분한 FPGA에 대해서는 병렬성을 증가시키거나 여러 개의 출력을 동시에 계산하는 방법으로 개선하여 처리성능을 향상시킬 수 있다.

References

[1] C. Zhang and P. Li, "Optimizing Fpga-based accelerator design for deep convolutional neural networks," in *FPGA'15,* pp.161-170, 2015. DOI: 10.1145/2684746.2689060

[2] J. Qiu and J. Wang, "Going deeper with embedded FPGA platform for convolutional neural network," *in FPGA'16,* pp.26-35, 2016. DOI: 10.1145/2847263.2847265

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv:1409.1556, 2014.

[4] Y. H. Chen and T. Krishna, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *in 2016 IEEE Int. Solid-State Circuits Conf.(ISSCC)*, pp.262-263, 2016. DOI: 10.1109/JSSC.2016.2616357

[5] K. He and X. Zhang. "Deep residual learning for image recognition," arXiv:1512.0338, 2015.

[6] S. Gupta, A. Agrawal, et.al, "Deep learning with limited numerical precision," arXiv:1502.02551, 2015.

[7] D. Lin, S. Talathi, V. Annapureddy, "Fixed point quantization of deep convolutional networks," arXiv:1511.06393, 2016.

[8] M. Courbariaux, Y. Bengio, and J.-P. David, "BinaryConnect: Training deep neural networks with binary weights during propagations," *in Proc. Adv. Neural Inf. Process. Syst.,* pp.3123-3131, 2015.

[9] M. Courbariaux and Y. Bengio, "Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1,"CoRR. 2016.

[10] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," *in Proc. Adv. Neural Inf. Process. Syst.*, pp.4107-4115, 2016.

[11] M. Rastegari and V. Ordonez, "XNOR-Net: ImageNet classication using binary convolutional neural networks," *In Proc. the European Conf. Computer Vision(ECCV'16)*, pp.525-542, 2016.

[12] R. Zhao and W. Song, "Accelerating binarized convolutional neural networks with software-programmable FPGAs," *in FPGA'17,* pp.15-24, 2017. DOI: 10.1145/3020078.3021741

[13] Y. Umuroglu and N. J. Fraser, "FINN: a framework for fast, scalable binarized neural network inference," *in FPGA'17*, pp.65-74, 2017. DOI: 10.1145/3020078.3021744

[14] S. Liang and S. Yin, "FP-BNN: Binarized neural network on FPGA," *Neurocomputing,* vol. 275, pp.1072-1086, 2018. DOI: 10.1016/j.neucom.2017.09.046

[15] R. Andri and L. Cavigelli, "YodaNN: An ultra-low power convolutional neural network accelerator based on binary weights," *in ISVLSI '16*, pp.236-241, 2016. DOI: 10.1109/ISVLSI.2016.111

[16] CIFAR-10 and CIFAR-100 datasets, https://www.cs.toronto.edu/~kriz/cifar.html

[17] J. H. Kim and S. K. Yun, "Accuracy analysis of fixed point arithmetic for hardware implementation of binary weight network," *Journal of IKEEE*, Vol.22, No.3, 805-809, 2019. DOI: 10.7471/ikeee.2018.22.3.805

BIOGRAPHY

Jong-Hyun Kim (Member)

2016 : BS degree in Biomedical Eng.ineering, Yonsei University
2018 : MS degree in Computer Science, Yonsei University
2018.8~ : Researcher, R&D Center, Telechips Inc.

SangKyun Yun (Member)

1984 : BS degree in Electronics Eng.ineering, Seoul National University.
1986 : MS degree in Electrical Engineering, KAIST
1995 : PhD degree in Electrical Engineering, KAIST

1992~2001 : Professor, Department of Computer Science, Seowon University
2001~ : Professor, Department of Comptuer and Telecom. Engineering, Yonsei Univ. Wonju