논문 2023-3-7 http://dx.doi.org/10.29056/jsav.2023.09.07

심층신경망의 스마트응용을 위한 온칩 시스템

이봉규*†

An On-a-Chip System for Smart Applications based on DNN (Deep Neural Network)

Bongkyu Lee*†

요 약

심층신경망 (Deep Neural Network, DNN)은 이미지 처리, 음성 인식과 같은 패턴인식 방면에서 뛰어난 성능을 보여주기 때문에 스마트응용을 효과적으로 구현할 수 있다. 그러나 부동소수점에 대한 MAC (Multiply and Accumulate) 연산을 위한 뛰어난 연산성능을 요구하는 단점이 있다. 본 논문에서는 심층신경망 연산기반 임베디드 스마트 애플리케이션을 위한 프로그래밍 가능한 온칩 (On-a-Chip) 시스템을 제안한다. 제안된 온칩 시스템의 구현 및 테스트를 위하여 FPGA 기반의 개발 플랫폼을 제작하여 사용한다. 프로토타입을 사용하여 메인 프로세서와 관련 하드웨어를 하나의 FPGA에서 구현할 수 있으므로, 모든 하드웨어 블록을 쉽게 통합하고 검증할 수 있다. 온칩 시스템의 코어 초기화, 동작 및 소프트웨어 개발 환경을 위한 운영프로그램을 구현한다. 제안한 온칩 시스템의 유효성을 실제 DNN 기반 응용시스템을 구현한 실험을 통해 보여준다.

Abstract

Although Deep Neural Network (DNN) shows excellent performance in pattern recognition such as image processing and voice recognition, it has the disadvantage of requiring intensive computing powers for real-time processing. This paper presents a programmable On-a-chip hardware for embedded smart applications based on Deep Neural Network (DNN) computations. To implement and test the proposed on-chip system, an FPGA-based development platform is produced and used. Since the main processor and related hardwares can be implemented in the same FPGA using the platform, all hardware blocks can be integrated and verified easily. A operation program is provided for On-a-chip system core initialization, operation, and software development environment. The effectiveness of the proposed On-a-chip system is shown through an experiment implementing an actual application system based on DNNs.

한글키워드 : 심층신경망, 스마트응용, 임베디드, 개발플랫폼, 오류역전파

keywords : DNN, Smart Applications, Embedded, Development platform, Error Backpropagation

1. 서 론

개인용 디바이스에 영상입력 장치가 보편화 되면서, 기존에 고성능 데스크톱이나 전용컴퓨터에서 구현되던 인공지능에 기반을 둔 '스마트' 응용을 개인용 디바이스에 구현하는 예가 증가하고 있다[1]. 심층신경망 (Deep Neural Networks, DNN)

* 제주대학교 데이터사이언스학과
† 교신저자: 이봉규(email: bklee@jejunu.ac.kr)
접수일자: 2023.09.05. 심사완료: 2023.09.11.
게재확정: 2023.09.20.

www.kci.go.kr

은 이미지 처리, 음성 인식과 같은 패턴인식 방면에서 뛰어난 성능을 보여주기 때문에 '스마트' 응용을 효과적으로 구현하는데 사용할 수 있다. 그러나 DNN은 부동소수점에 대한 MAC (Multiply and Accumulate) 연산을 해야 하며 이를 실시간으로 처리하기 위한 뛰어난 연산능력을 요구한다. 따라서 부동소수점 연산 기능이 없는 개인용 디바이스에 소프트웨어적으로 구현하는 것은 어렵다. 이런 문제점을 극복하기 위하여 DNN을 위한 가속화 H/W를 통해 해결하는 방법이 다양하게 연구되고 있다[2-4]. 그러나 현재 DNN을 위한 일반화된 가속화 H/W 구현은 2가지의 문제점을 해결하지 못하고 있다. 일반적으로 DNN 기반의 응용 시스템은 응용에 따라서 각기 다른 기능을 포함할 수 있으므로 S/W를 포함할 수 있는 구조가 필요하다. 또한, 응용에 따라 변하는 DNN의 구조에 무관한 일관된 H/W 작동이 필요하다.

본 논문에서는 계산자원이 한정된 개인용 디바이스에 DNN에 기반을 둔 '스마트' 응용 구현에 사용 가능한 온칩 시스템 (On-a-Chip System) 형태의 DNN_SoC를 Co-design[5] 방법으로 설계한다. 일반적으로 신경망을 이용한 응용은 Preprocessing 단계, 신경망 가중치 연산과 결과에 대한 Postprocessing 단계로 구성이 된다. 이들 중에서 Preprocessing과 Postprocessing은 복잡한 MAC 계산이 필요 없다. 그러나 신경망 연산의 경우는 처리방법은 일반화되어 있어 변화가 적으나 MAC 연산으로 인한 계산량이 많다. 따라서 신경망 연산 단계에 해당하는 가중치 계산은 H/W로 구성하고, 계산이 복잡하지 않은 나머지 부분은 소프트웨어로 구현할 수 있는 형태로 설계한다. 제안하는 온칩 시스템에 대한 구현을 위해 자체적인 개발플랫폼을 제작하여 사용한다. 구현되는 개발플랫폼의 FPGA에는 DNN_SoC와 함께 IP (Intellectual Property)를 활용한 처리장치 (Processor) 및 제어용 H/W들도 함께 구현되기 때문에 전체 시스템을 통합 검증할 수 있다. 구현된 온칩 시스템의 동작 설정과 임베디드 S/W 개발을 위하여 운영프로그램을 구축한다. 제안하는 온칩 시스템 DNN_SoC의 유효성을 보이기 위하여 DNN 기반의 숫자 인식시스템 전체를 SoC에 실제 구현하고 성능을 평가한다.

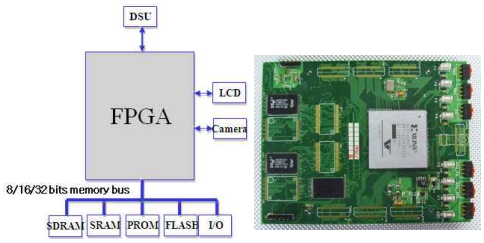
2. 개발플랫폼 구현 및 활용

FPGA에 기반을 둔 플랫폼은 H/W를 포함한 다양한 온칩 시스템을 보드 형태로 설계하고 레지스터 수준에서 설계하고 검증할 수 있는 환경이 필요하다. 이런 이유로 FPGA 기반의 개발플랫폼이 상용화되어 개발에 활용되고 있다. 그러나 이런 상용화된 개발플랫폼은 CPU를 비롯한 여러 컨트롤러가 FPGA 외부에 미리 구현되어 있으므로 독립적으로 운영되는 온칩 형태의 개발에는 사용이 어렵다. 또한, 외부에서 제공하는 다양한 IP 기반 H/W블록을 함께 사용할 수 있는 유연성이 필요하다. 본 연구팀은 오픈 CPU 기반의 개발플랫폼을 제작과 이를 통한 다양한 온칩 시스템 구현에 관한 연구를 진행하였다[6]. 이번 연구에서는 DNN과 같은 복잡한 신경망의 구현에 사용할 수 있도록 기존의 개발플랫폼을 개선하여 사용한다.

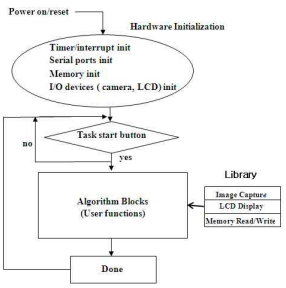
플랫폼에는 FPGA (XILINX)를 중심으로 512M 바이트와 128M바이트의 SDRAM 기반 및 플래시 메모리를 보유하고 있다. SDRAM 메모리는 외부 디바이스와 인터페이스 및 주기억장치 임무를 수행하기 위한 것이며, 플래시 메모리는 코드가 들어가는 부분으로 전원에 무관하게 코드를 유지할 수 있으므로 운영프로그램과 구현되는 온칩 시스템의 S/W 부분을 저장하는 기능을 수행한다. 이전의 개발플랫폼에 비하여 이들 메모리는 확장이 되었다. FPGA에 DNN 관련 부분 외에 구현될 기본적인 요소로는 CPU, 버스시스템, 외부 센서제

www.kci.go.kr

어용 컨트롤러 등이다. 그림 1에서는 실제로 구현한 개발플랫폼과 이 플랫폼을 구동하는 운영체제인 운영프로그램의 사양을 보여준다. 운영프로그램에 있는 'Algorithm Blocks'에는 구현되는 DNN 관련 S/W가 저장된다.



(a) The development platform



(b) Operating Program

그림 1. 개발플랫폼 및 운영프로그램

Fig. 1. Development platform and operating program

목표로 하는 온칩 시스템을 운영하는 중앙처리장치는 LEON 2를[7] 사용한다. 채택된 LEON2 CPU와 다른 H/W 요소 간의 데이터/제어의 흐름을 위해서 사용되는 내부 버스시스템은 AMBA (AHB/APB) 버스를 채용하고 구현한다. 외부데이터를 얻는데 사용하는 외부디바이스의 제어를 위한 인터페이스로는 카메라와 같은 영상장치를 대상으로 한다. 이러한 모든 요소는 실제 신경망 연산을 담당할 DNN H/W와 같은 FPGA에 모두 통합된 HDL (Hardware Description Language)로 구현이 되기 때문에 하나의 칩에서의 동작을 개발단계부터 검증이 가능한 장점이 있다.

영상처리/인식 응용에서 활발히 사용되고 있는 DNN은 2개 이상의 은닉층을 가지는 다단계 신경망으로써 서로 다른 범주의 입력 집합과 출력 집합을 대응시킬 수 있다. 이러한 매핑 능력은 심층 신경망을 구성하는 각 층의 노드들이 비선형적으로 입력을 처리하기 때문이다. DNN에서의 학습은 대상 문제에 대해서 적절한 가중치를 찾는 과정이다. 현재 가장 널리 사용되는 학습 알고리즘은 오류역전파방법 (Error Backpropagation)으로 단순하면서 일반화 기능으로 많은 분야에서 사용할 수 있다[8]. 그러나 많은 연산을 요구하는 단점이 있다.

DNN의 인식 과정을 위한 출력값은 다음과 같이 진행된다. 먼저 입력층으로 들어오는 인식대상에 대해서 출력층까지 순방향으로 차례대로 출력이 계산된다. 즉 입력층으로부터 받은 값을 이용하여 은닉층이 출력값을 계산한 후, 그 출력값을 상위의 층으로 계속 전달하는 과정을 출력층까지 진행한다. 수식 (1)은 노드 i가 n개의 자신보다 하위의 층에 있는 노드로부터 들어오는 값과 가중치를 결합한 값의 합 $receive_i$를 구하는 과정을 보여준다. w_{ij}는 노드 i 와 j사이의 가중치이며, out_j는 하위층 노드 j로부터 들어오는 값을 나타낸다. 노드 i의 최종 출력값은 (1)에서 얻어진 값을 활성함수 f로 처리한 수식 (2)로 얻을 수 있다.

$$receive_i = \sum_{j=1}^{n} w_{ij} out_j \quad (1)$$

$$out_i = f(receive_i) \quad (2)$$

수식 (1), (2)는 신경망 연산의 핵심이며, 반복적으로 수행된다. 따라서 이 연산을 빠르게 수행하는 것이 신경망의 처리속도를 높이는 핵심이다. 먼저 수식 (1)의 연산을 효율적으로 처리하기 위해 가중치를 고정소수점으로 변환시켜서 연산하는 형태로 H/W를 설계한다. 수식 (2)의 활성화 함수를 빠르게 계산하려는 방법으로 원하는 활성함수의 입력에 따른 출력값을 미리 계산하여 메모리에 저장하는 방법을 통하여 복잡한 함수연산을

www.kci.go.kr

메모리 검색으로 변환하는 방법으로 변환한다. 이런 기법을 통하여 연산시간은 줄이면서 다양한 함수를 사용할 수 있는 유연성을 확보한다.

3. DNN_SoC

LEON 2를 중심으로 설계한 심층신경망 기반의 상'스마트응용'을 구현할 수 있는 온칩 시스템(DNN_SoC)의 내부 구조를 그림 2에서 보여준다.

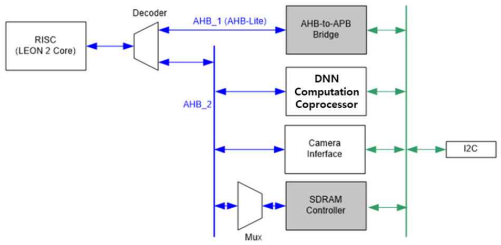


그림 2. 실현된 내부 구조도
Fig. 2. The architecture of internal blocks

주 CPU인 LEON2는 공개된 IP를[7] 이용하여 FPGA에 구현한다. LEON2와 함께 FPGA에는 AHB_APB 버스 제어기, 외부장치 제어기와 메모리 제어기 (Memory controller)를 구현한다. AHB/APB 시스템은 CPU와 다른 제어기들의 인터페이스를 위해 사용된다. AHB 시스템은 메인 버스 (AHB_1)와 보조버스 (AHB_2)로 분리하여 설계하고 디코더를 이용하여 통합 운영한다. APB 버스는 컨트롤러와 외부장치를 연결하는 것으로 I^2C 인터페이스는 영상장치 연결에 사용된다.

온칩 시스템에서 핵심적인 부분은 실제 DNN에서 신경망 연산을 담당하는 'DNN Computation Coprocessor' (DNN_CC)이며, DNN_CC의 구조가 그림 3에 나타나 있다. DNN_CC는 메모리 및 버스시스템에 대한 접근과 제어를 위한 'Host interface' 부분과 심층신경망의 각 층을 구성하는 노드들의 출력값 연산을 담당하는 'DNN' 블록과 2개의 데이터 버퍼 (A, B)로 구성된다. 데이터 버퍼를 2개로 구성한 것은 하나의 버퍼에 있는 입력을 처리하면서 동시에 새로운 데이터를 나머지 버퍼로 가져오는 방식으로 성능을 높이기 위함이다.

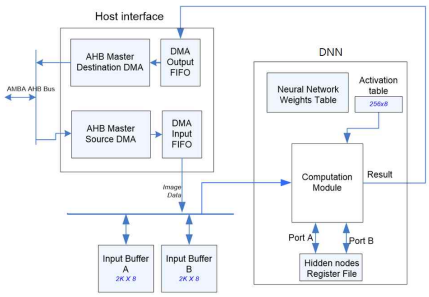


그림 3. DNN 기반연산을 위한 H/W 구조
Fig. 3. Architectural overview of computations

'Host interface' 블록은 DNN_CC와 그림 2의 다른 블록과의 인터페이스를 담당한다. Direct Memory Access (DMA)를 지원하기 때문에 DNN_CC가 CPU의 도움이 없이도 SRAM에 읽기/쓰기가 가능하다. 'Source DMA'는 CPU 외부에 메모리에 있는 입력을 DNN_CC에 있는 입력 레지스터로 넘겨준다. 입력을 연산하여 얻어진 결과는 'Destination DMA'가 SDRAM에 저장된다.

DNN 블록은 실제 연산을 담당하는 곳으로 저장공간과 계산 모듈인 'Compute_Module'로 구성되어 있다. 'Compute_Module'은 버퍼에 저장된 입력을 입력노드로 가져와서 최종적인 출력 노드의 값이 계산될 때까지 전방향 (Foreward)으로 각 층의 노드 값을 순차적으로 계산한다. 계산된 출력 노드의 값은 'Host Interface' 블록에 전달되어 SDRAM에 저장된다. 그림 4에서 'Compute_Module'의 레지스터 전달 레벨 (Register Transfer Level, RTL) 에서의 블록도와 계산에 사용되는 자료의 H/W 표현을 보여준다.

www.kci.go.kr

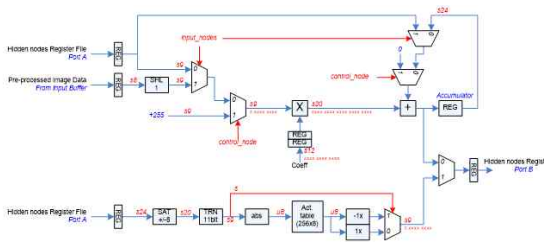


그림 4. 'Compute_Module'의 RTL 블록도
Fig. 4. The RTL block of the Compute_Module

DNN 블록에 있는 저장공간은 가중치 테이블, 활성함수 테이블과 레지스터 파일이다. 가중치 테이블은 현재 계산 중인 층간의 가중치를 가지며 19 bits (12 bits 가중치, 7 bits 인덱스) 정밀도이다. 활성 함수테이블은 미리 계산된 활성함수의 값을 저장하는 곳으로, DNN블록의 구조 변경없이 활성함수를 변경할 수 있게 한다. 레지스터 파일은 은닉층과 출력층 간의 중간값을 임시 보관하는 24bits 정밀도의 레지시터 집합이다.

4. 실험 및 결과

구현되는 DNN_CC를 구성하는 모든 H/W들은 Verilog HDL(VHDL)로 코딩되고 XILINX ISE[9]로 개발플랫폼 FPGA에 구현한다. 온칩 시스템의 동작을 검증하기 위하여 영상 응용시스템을 구현된 온칩 시스템에 직접 구현하고 실제 실행 여부를 검증한다. 구현되는 응용시스템은 MNIST[10] 필기체 숫자를 인식하는 것이다. 응용시스템은 3개의 주요 단계로 구성이 된다. MNIST의 필기체 숫자 100개를 가지고 있는 640×720크기의 영상 2개를 외부로부터 SRAM에 입력받는다. 입력된 문서영상은 전처리 과정을 통하여 개별 숫자 단위로 분리한다. 분리된 숫자 이미지들은 입력 크기인 28X28로 정규화한다. 이런 과정을 거쳐 얻어진 정규화된 200개의 필기체 숫자 이미지는 하나씩 차례대로 학습을 완료한 가중치를 가지는 온칩 시스템의 DNN에 입력되어 인식된다. 인식된 결과는 후처리 루틴에서 처리되어 출력된다. 표 1은 구현된 DNN의 학습에 관련된 주요 변수명과 사용된 값을 보여준다. 실험결과 구현된 DNN은 문서로부터 분리된 200개의 (숫자 당 20개)의 이미지 중에서 191개를 정확히 분류하였다.

표 1. 제안 DNN의 주요 형태
Table 1. Configurations of the proposed DNN

학습 관련 변수명	사용값
입력/은닉1/은닉2/출력 노드 수	784/16/8/10
활성함수	ReLU
인식률	95.5%

인식률과 더불어 온칩 시스템에 구현된 인식시스템의 처리속도를 평가한다. 구현된 인식시스템을 구성하는 3개 단계 각각에 대한 실행시간은 표 2에 나타나 있다. 구현된 인식시스템은 4.1초가 소요되었으며 실제 전처리와 후처리를 제외한 실제 인식에 (DNN 처리시간) 소요된 시간은 3.01초 (초당 66개 처리)'로 측정되었다. H/W 구현으로 얻어지는 속도 향상을 확인하기 위하여 LEON 2상에 전처리나 후처리와 같이 S/W 적으로 구현하고 인식을 수행하였다. 수행 결과 CPU를 사용하는 S/W로 구현할 경우 DNN 수행에 668초가 소요되는 것으로 확인되었다. 결과적으로 온칩 시스템은 S/W보다 200배의 처리속도 향상을 실현하였다. 이런 결과를 통하여 구현한 온칩 시스템은 제한된 연산능력을 가지는 개인용 디바이스에서 기존의 S/W적 방법으로는 불가능한 심층신경망 기반의 복잡한 응용을 구현하는 것을 가능하게 한다.

표 2. 단계별 처리시간
Table 2. Processing times of processing steps

처리 단계	구현형태	요구시간(msec)
분리/정규화	S/W (LEON2)	953
신경망 연산	DNN H/W	3,061
후처리	S/W (LEON2)	92

www.kci.go.kr

5. 결론

본 연구에서는 개인용 디바이스에 '스마트' 응용의 효과적인 구현에 사용이 가능한 DNN_SoC를 제안하고 설계하였다. FPGA 기반의 개발플랫폼에 온칩 시스템을 실제 구현하고 테스트 및 작동을 확인하였다. 제안된 온칩 시스템은 Co-design 기법을 사용하여 S/W와 융합되는 구조를 가지기 때문에 변화되는 부분은 S/W로 처리할 수 있어서 다양한 구조의 DNN을 시스템에 구현하는 것이 가능하다. 개인용 디바이스에 적용이 가능한 응용 중 하나인 숫자 인식시스템 전체를 온칩 시스템에 구현하는 실험을 통하여 제안한 온칩 시스템이 개인용 디바이스에 '스마트' 응용을 구현할 때 효과적으로 쓸 수 있음을 보였다.

참 고 문 헌

[1] Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017). Understanding of a convolutional neural network. International Conference on Engineering and Technology (ICET), 1-6. https://ieeexplore.ieee.org/document/8308186

[2] Chen, Y., Xie, Y., Song, L., Chen, F., & Tang, T. (2020). A survey of accelerator architectures for deep neural net works. Engineering, 6(3), 264-274. https://www.sciencedirect.com/science/article/pii/S2095809919306356

[3] Lulin Chen. (2018). A CNN(Convolutional Neural Network) hardware implementation. https://github.com/lulinchen/cnn_open

[4] G. Kwon & T. Suh. (2022). A Study of Edge AI Hardware for Real-Time Inference of Convolutional Neural Network Model Using FPGA. The winter Conference of Korean association of Computer Education, 26(1), 273-276. https://www.dbpia.co.kr/pdf/pdfView.do?nodeId=NODE11032169&googleIPSandBox=false&mark=0&ipRange=false&accessgl=Y&language=ko_KR&hasTopBanner=true

[5] M. Shabiul, M. S. Beg, M. S. Bhuyan & M. Othman. (2006). Design and Implementation of Discrete Cosine Transform Chip for Digital Comsumer Products. IEEE Transaction on Consumer Electronics, 52(3), 998 - 1003. https://ieeexplore.ieee.org/document/1706499

[6] B. Lee. (2014). A programmable Soc for Various Image Applications Based on Mobile Devices. Journal of Korea Multimedia Society, 17(3), 324-332. http://dx.doi.org/10.9717/kmms.2014.17.3.324

[7] Gaisler Research.. LEON2 processor user's manua. http://www.gaisler.com

[8] Alex Krizhevsky, Ilya Sutskever & Geoffrey E. Hinton. (2011). ImageNet - Classification with Deep Convolutional Neural Networks. Proceedings of the 25th International Conference on NIPS, 1, 1097-1105. https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[9] P. G. D. Valle, D. Atienza, G. Paci & F. Poletti. (2007). Application of FPGA Emulation to SoC Floorplan and Packaging Exploration. XXII Conference on Design of Circuits and Integrated System, 236 - 240. https://www.researchgate.net/publication/37450691_Application_of_FPGA_Emulation_to_SoC_Floorplan_and_Packaging_Exploration

[10] LeCun, Y. (2013). Mnist handwritten digit database. http://yann.lecun.com/exdb/mnist/

저 자 소 개

이봉규 (Bongkyu Lee)

1995.2 서울대학교 컴퓨터공학과 박사
1996.3-현재 : 제주대학교 교수
<주관심분야> 인공지능, 패턴인식

www.kci.go.kr