RISC-V 플랫폼 기반 CNN 모듈의 버퍼링 분석

김진영*, 임승호*
*한국외국어대학교 컴퓨터공학부
kestrel960408@gmail.com, slim@hufs.ac.kr

Buffering analysis of CNN module based on RISC-V platform

Jin-Young Kim*, Seung-Ho Lim*
*Divition of Computer Engineering, Hankuk University of Foreign Studies

요 약

최근 임베디드 엣지 컴퓨팅 디바이스에서 AI와 같은 인공지은 연산을 수행하여 AI 추론 연산의 가속화 및 분산화가 많이 이루어지고 있다. 엣지 디바이스는 임베디드 프로세서를 기반으로 AI의 가속 연산을 위해서 내부에 딥러닝 가속기를 포함하여 가속화시키는 시스템 구성을 하고 있다. 딥러닝 가속기는 복잡한 Neural Network 연산을 위한 데이터 이동이 많으며 외부 메모리와 내부 딥러닝 가속기 간의 효율적인 데이터 이동 및 버퍼링이 필요하다. 본 연구에서는 엣지 디바이스 딥러닝 가속기 내부의 버퍼 구조를 모델링하고, 버퍼의 크기에 따른 버퍼링 효과를 분석해 보았다. 딥러닝 가속기 버퍼 구조는 RISC-V 프로세서 기반 가상 플랫폼에 구현되었다. 이를 통해서 딥러닝 모델에 따른 딥러닝 가속기 버퍼의 사용성을 분석할 수 있다.

1. 서론

최근 임베디드 엣지 컴퓨팅 시스템에서는 데이터나 사물의 학습과 추론을 위해서 AI가 적용되고 있는데, 엣지 컴퓨팅 시스템의 자원의 효율적 활용을 위해서 주로 학습은 서버단에서 수행하고, 추론은 엣지 디바이스 자체에서 직접 수행하는 방식으로 진화하고 있다. 엣지 디바이스에서 추론을 가속화하기 위해서 최근 임베디드 프로세서는 딥러닝 최적화를 위한 딥러닝 가속기를 내부 구성 요소로 포함하여 설계가 되는 연구가 많이 진행되었다. 그러나 여전히 엣지 디바이스와 같은 임베디드 디바이스는 고성능 서버 시스템에 비해서 컴퓨팅 자원의 한계로 인해서 딥러닝 가속기를 포함한 딥러닝 아키텍처의 최적화가 필요하다.

딥러닝 가속기를 포함한 임베디드 시스템 플랫폼은 프로세서 내부에 코어와 전용 딥러닝 가속기로 구성되며 딥러닝 모델의 네트워크 처리에 따른 순서대로 프로세서와 딥러닝 가속기 간의 인터렉션을 통해서 딥러닝 연산을 수행한다. 이 때 딥러닝 가속기 내의 내부 버퍼의 크기와 데이터 타입에 따른 버퍼링 등 효율적인 버퍼 사용성이 필요하다.

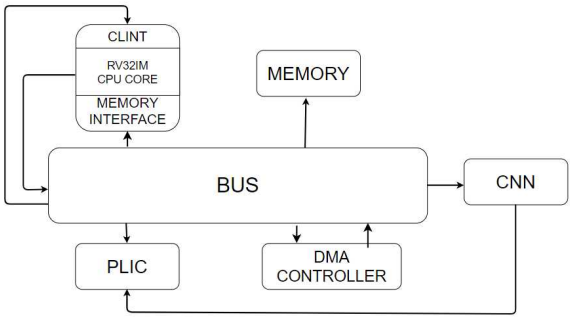
본 논문에서는 임베디드 프로세서 기반 딥러닝 가속기와 딥러닝 가속기의 버퍼를 모델링 및 구현하고하고, 딥러닝 네트워크 모델에 따른 임베디드 엣지 프로세서의 딥러닝 가속기의 버퍼링 효과에 대해서 분석을 해보았다. 딥러닝 가속기 버퍼는 RISC-V[1] 플랫폼 기반으로 SystemC[2]로 구현하였으며, 실행한 딥러닝 모델은 Yolo Tiny3[3]이다. 이를 통해서 딥러닝 모델이 딥러닝 가속기 기반 엣지 디바이스에서 실행될 때 내부 버퍼의 사용성을 분석할 수 있다.

2. 딥러닝 가속기 및 버퍼 모델링

임베디드 프로세서 기반의 딥러닝 가속기 플랫폼은 최근 오픈소스 임베디드 프로세서 아키텍처로 각광받고 있는 RISC-V 기반의 가상 플랫폼[4,5]을 기반으로 딥러닝 가속기 시스템을 구축하였다. 기존 RISC-V 기반 가상 플랫폼은 SystemC로 구현되어 있으며, RISC-V 프로세서와 메인 메모리 및 몇가지 컨트롤러로 구성되어 있다. 각 모듈은 TLM(Transaction Level Model) 2.0기반 System Bus에 연결된다. 전체적인 구조는 그림 1에 나타난 바와 같으며, 각 모듈의 구성은 다음과 같다.

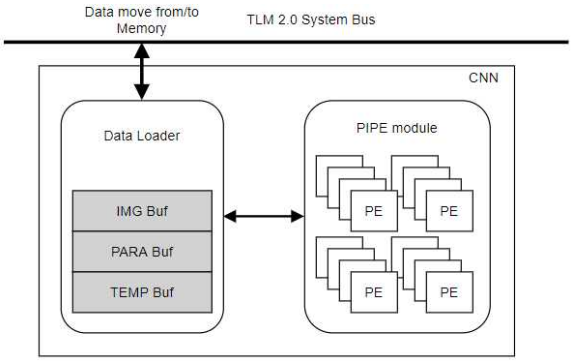
RV32IM CPU core 모듈은 RISCV-VP 32bit CPU core를 구현한 것이고, CLINT(Core Local Interrupt Controller)와 Memory Interface를 포함한다. PLIC(Platform Level Interrupt Controller)는 core이외의 모든 interrupt를 처리해주는 모듈이다. Memory 모듈은 DRAM과 같은 메모리를 모델링한 모듈이다. TLM 2.0 기반의 Bus에 연결되며, CPU core의 Initiator Socket을 통해서 지정된 주소 영역에 해당되는 Target Socket에 연결된 모듈과 통신을 수행한다.

Bus에 연결된 CNN 모듈은 딥러닝 연산 가속을 위해서 추가된 모듈이다. CNN 모듈은 기본적으로 Convolution 연산과 Activation/Pooling에 대한 하드웨어 연산을 설정에 따라서 수행하도록 설계 및 구현되었다.



(그림 1) RISC-V 가상 플랫폼 기반 딥러닝 가속기 구조도

그림 2는 CNN 모듈 내부 구조를 도식화 한 것이다. CNN 모듈은 크게 Data Loader 모듈과 PIPE 모듈 두 개로 구성되어 있다. Data Loader 모듈은 직접 TLM 2.0 System Bus와 연결되며 RISC-V CPU 및 Memory와 인터렉션을 하여 딥러닝 대상이 되는 데이터를 주고 받는다. Data Loader 모듈 내부에는 각 데이터 타입 별로 버퍼를 구성하여, 데이터를 각각 버퍼링 할 수 있는 구조로 구성되어 있다. 일반적으로 CNN 연산을 수행하기 위해서는 연산의 대상이 되는 이미지 데이터, Parameter 데이터, Bias 데이터 등으로 구성된다. 또한, 내부 데이터 연산의 결과를 임시로 저장하는 Temporary 버퍼 영역도 필요하다. 그림에서 보는바와 같이 각 버퍼 영역은 데이터 타입별로 별도로 구성되며, 데이터를 이동하거나 연산의 결과를 저장할 때 독립적인 데이터 이동 및 관리가 가능하다. PIPE 모듈은 Data Loader 모듈과의 인터랙션을 통해서 연산의 대상이 되는 데이터를 받아서 실제 연산을 수행한다.



(그림 2) CNN 모듈 내부와 Data Loader 서브 모듈에 존재하는 데이터 타입별 버퍼 구조

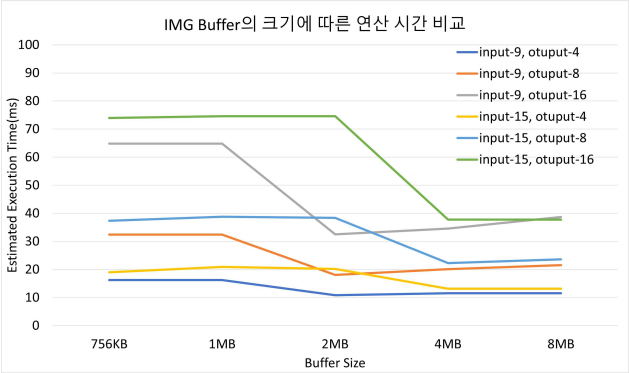
CNN 모듈에서 수행하는 연산은 Convolution, Activation, Pooling 연산을 수행할 수 있으며, 데이터 이동 시 함께 설정된 각 연산의 설정에 따른 연산을 수행한다. 각 연산의 수행은 내부에 구성된 연산 조직인 PE(Processing Element)에 부분 연산을 분산시켜서 연산을 수행하며, 연산의 결과는 다시 Data Loader 모듈의 Temp Buffer에 버퍼링 된다. 위의 CNN 모듈 및 내부 모듈과 버퍼 구조는 RISC-V 가상 플랫폼에 SystemC로 구현하였으며, 각 모듈 및 버퍼는 실행 시간 및 버퍼 크기를 내부 파라미터로 설정하는 것이 가능하므로, 다양한 딥러닝 모델의 구성에 대한 버퍼 크기별 연산에 대한 시간 분석이 가능하다.

3. 실험 결과

구현된 RISC-V 가상 플랫폼 기반 CNN 모듈의 버퍼 사용성을 분석하기 위해서 가상 플랫폼 위에서 실행할 수 있는 Convolution 연산 응용 프로그램을 작성하였다. 작성된 Convolution 연산 프로그램은 Convolution의 다양한 설정이 가능하도록 되어 있는 RISC-V 기반 C 프로그램이다. 해당 프로그램은 RISC-V 크로스 컴파일러로 컴파일하여 ELF 포맷으로 생성되며, 생성된 ELF 실행 파일은 RISC-V 가상 플랫폼 위에서 실행 가능하다.

우리는 다양한 형태의 Convolution 설정에 따른 버퍼 성능을 분석하기 위하여 다음과 같은 설정을 사용하였다. Convolution 연산의 설정은 Yolo Tiny3의 layer 0을 모델링하여 생성하였으며 이미지 크기 416x416에 filter 크기는 3x3 필터를 사용하였다. 우리는 다양한 input 및 output channel 수에 따른 이미지 버퍼의 크기와 이미지 버퍼 대비 데이터 이동량 및 연산 시간 분석이 가능한가를 알아

보기 위한 실험을 진행하였다.



(그림 3) Image Buffer의 크기에 따른 각 딥러닝 모델 설정 별 연산 시간

다양한 버퍼 중에서 이미지 버퍼에 대한 분석을 진행하였으며, 이미지 버퍼의 크기를 756KB에서부터 8MB까지 변경시켜가며 Convolution 연산을 수행하였다. 그림 3은 이미지 버퍼의 크기에 따른 각 Convolution 연산의 Input-output 설정에 대한 Convolution 연산 시간을 도식화한 것이다. 연산 시간은 시뮬레이터에서 실행되는 각 모듈의 상대 시간이다. 그림에서 보는 바와 같이 각 Convolution 연산의 설정에 대해서 이미지 버퍼의 크기에 따른 각 연산의 연산 시간의 변화를 관찰할 수 있다. 이를 이용해서 실시간으로 변화하는 딥러닝 연산의 변화에 따른 적절한 버퍼 크기를 결정하여 시스템에 적용하는 적응이 가능하다.

4. 결론

최근 임베디드 엣지 디바이스에서 인공지능 연산 및 추론을 수행하는 IoT시스템의 적용이 늘고 있다. 엣지 디바이스는 여전히 컴퓨팅 자원의 제한적 사용이 존재하는 시스템이므로 엣지 디바이스에서 딥러닝 연산을 최적화하기 위한 작업이 필요하다. 본 논문에서는 RISC-V 가상 플랫폼 기반 딥러닝 연산기 구조와 내부 버퍼 구조를 설계하고 시뮬레이션 하였다. 특히, 딥러닝 가속기 버퍼는 다양한 딥러닝 설정에 따른 사용성 분석이 가능하므로 엣지디바이스의 리소스 사용량을 분석할 수 있다.

Acknowledgments

본 연구는 과학기술정보통신부 및 정보통신기획평가원의 SW중심대학지원사업의 연구결과로 수행되었음 (2019-0-01816)

참고문헌

[1] A. Waterman, Y.S. Lee, and R. Avizienis, D. Patterson, and K. Asanovi´c ”The RISC-V Instruction Set Manual“, Volume II: Privileged Architecture, July, 2016, https://people.eecs.berkeley.edu/~krste/papers/riscv-privileged-v1.9.pdf

[2] D.C. Black, J. Donovan, B. Bunton, and A. Keist, “SystemC : From The Ground up”, Eklectic Ally, Inc.

[3] Darknet, http://pjreddie.com/Darknet/

[4] Github, RISC-V Virtual Prototype, https://github.com/agra-uni-bremen/riscv-vp

[5] V. Herdt, D. Großе, H. M. Le and R. Drechsler, "Extensible and Configurable RISC-V Based Virtual Prototype," 2018 Forum on Specification & Design Languages (FDL), Garching, pp. 5-16, 2018.

[6] NVIDA, NVIDIA Deep Learning Accelerator, [Online]. Available: https://nvdla.org