타일링과 스케줄링: 딥러닝 가속 하드웨어의 실행코드 최적화

권용인 ○, 김영주, 유미선, 이제민, 김태호
한국전자통신연구원
{yongin.kwon, kr.yjkim, msyu, leejaymin, taehokim}@etri.re.kr

Tiling and Scheduling: Machine code optimization for deep learning accelerators

Yongin Kwon ○, Young-Joo Kim, Misun Yu, Jemin Lee, Taeho Kim
Electronics and Telecommunications Research Institute

요 약

딥러닝은 인공지능 영역에서 객체인식, 자연어 처리 등의 영역에서 널리 사용되고 있다. 모바일 기기 상에서도 딥러닝 추론을 성공적으로 수행하기 위해 여러 종류의 딥러닝 가속 하드웨어가 개발되고 있고, 이러한 하드웨어 상에서 효율적인 인공 신경망 연산을 수행하기 위해서는 효율적인 실행코드의 생성이 필요하다. 본 논문에서는 딥러닝 가속 하드웨어 개발의 한 방법인 FPGA로 개발된 VTA 딥러닝 가속 하드웨어를 대상으로 최적화 코드 생성 기법인 타일링과 스케줄을 적용해 본다. 그 결과 더 많은 메모리 연산이 필요했음에도 불구하고 인공 신경망 연산의 수행속도가 30% 향상됨을 보여준다.

1. 서 론

막대한 연산 처리량을 요구하는 딥러닝(Deep Learning) 기술은 그래픽 프로세서 등의 컴퓨팅 능력의 발달과 더불어 자연어 처리나 컴퓨터비전 같은 인공지능 영역에서 높은 성능을 보여주었다. 최근 휴대전화나 IoT 기기 등 모바일 기기에서도 객체 인식 등의 딥러닝 연산 수행이 요구됨에 따라 딥러닝 관련 알고리즘, 신경망 모델 최적화, 딥러닝 프레임워크 개발, 컴파일러 최적화, 딥러닝 가속 하드웨어의 개발 등 여러 방면에서의 연구가 활발하게 진행 중이다.

모바일 환경에서 딥러닝 가속 하드웨어의 성능을 높이기 위해서는 인공 신경망 연산에 따른 최적의 실행코드 생성이 필요하다. 특히, 딥러닝 가속 하드웨어의 내부의 버퍼는 하나의 인공 신경망 연산 계층의 입출력 데이터를 한 번에 담지 못하는 경우가 많다. 이로 인해 입출력 데이터를 여러 조각으로 나누고 버퍼의 할당과 스케줄링을 거쳐 순차적으로 실행하도록 하는 최적의 실행코드 생성이 필요하다.

본 논문에서는 오픈소스 딥러닝 가속 하드웨어인 VTA를 대상으로 하나의 인공 신경망 계층 연산을 위해 타일링과 스케줄링 기법을 적용하여 생성된 실행코드를 비교한다.

2. 딥러닝 가속 하드웨어

딥러닝 가속 하드웨어는 크게 범용 하드웨어와 딥러닝 전용 하드웨어로 나뉜다. 범용 하드웨어에는 CPU와 GPU가 있으며, 특히 GPU는 높은 병렬성을 갖기 때문에 딥러닝 연산에 적합하여 학습과 추론에 널리 쓰이고 있다. 딥러닝 전용 하드웨어는 인공 신경망 연산 속도를 가속화하고 전력소모를 줄이는 등의 목적으로 만들어진 하드웨어이다. Google사의 TPU, Alibaba사의 Hanguang 등과 같이 기업과 학계에서 딥러닝 학습 및 추론을 가속화하는 다양한 형태의 하드웨어를 개발하고 있다.

FPGA(Field Programmable Gate Arrays)는 재프로그래밍 가능한 집적회로로, 저전력 고성능의 특성 때문에 ASIC(Application Specific Integrated Circuit)의 프로토타이핑 용도 뿐만 아니라 통신, 의료 등 다양한 분야에서 사용 가능하다. 딥러닝 분야에서의 FPGA는 범용 하드웨어의 전력 비효율성과 ASIC의 낮은 유연성을 보완할 수 있는 특성을 가지고 있어, 인공 신경망 모델의 특성에 맞게 다양한 형태로 개발이 가능하다.

FPGA상에서 개발된 딥러닝 가속 하드웨어는 크기는 작지만 반응 속도가 빠른 SRAM으로 구성된 내부 버퍼, 크기는 크지만 반응 속도가 비교적 느린 DRAM, SRAM과 DRAM 사이의 메모리 연산을 전담하는 DMA(Direct Memory Access)유닛 그리고 딥러닝 연산을 병렬로 수행 가능한 다수의 PU(Processing Unit)로 구성된다. 그림 1은 보편적인 딥러닝 가속 하드웨어의 구조를 보여준다. PU는 딥러닝에서 많이 쓰이는 연산자인 Convolution이나 행렬곱을 가속화하기 위해 MAC(Multiply-Accumulate) 연산을 다양한 방법으로 병렬화 한다. PU의 빠른 연산능력을 뒷받침하기 위해서는 입력 데이터의 공급과 출력 데이터의 저장이 원활하게 이루어 져야 하고, 이를 위해

SRAM을 버퍼로 사용한다.

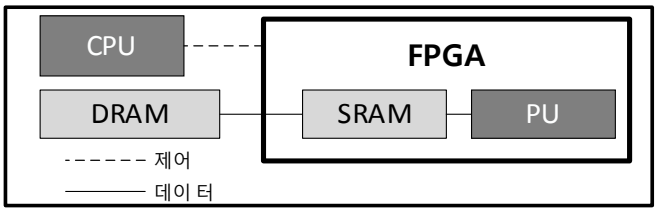

그림1 FPGA 상에서의 딥러닝 가속 하드웨어 구조

3. VTA

3.1 VTA 하드웨어 구조

VTA(Versatile Tensor Accelerator)는 오픈소스 가변형 딥러닝 가속 하드웨어로 FPGA 보드 상에서 구현이 가능하다. 그림 2와 같이 VTA는 여러 모듈로 구성되어 있고 각 모듈 사이에 데이터 흐름을 제어하기위한 큐(Queue)와 FPGA의 블록 RAM으로 구현한 버퍼들이 존재한다. 각 모듈의 기능은 다음과 같다. LOAD 모듈은 DRAM으로부터 입력 데이터와 가중치 데이터를 각각 INPUT 버퍼와 WEIGHT 버퍼로 로드한다. COMPUTE 모듈은 이 버퍼들로부터 데이터를 읽어 GEMM(General Matrix Multiply) 연산과 산술연산을 수행한 후 OUTPUT 버퍼에 저장한다. STORE 모듈은 OUTPUT 버퍼의 데이터를 DRAM으로 내보낸다. 인공 신경망 연산에 따라 버퍼할당, 연산순서 스케쥴링 등이 정해진 실행 코드가 하드웨어세서 수행되고, 이는 수행시간과 전력소모에 영향을 준다.

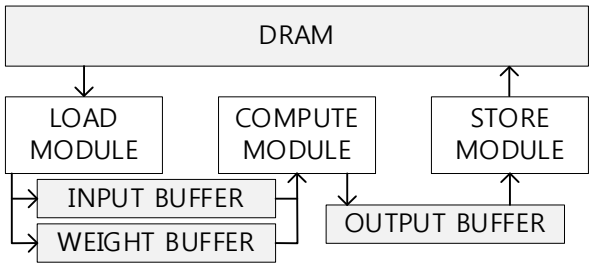

그림 2 VTA의 하드웨어 구조

3.2 타일링

SRAM은 DRAM에 비해 반응속도가 빨라 딥러닝 가속 하드웨어의 내부 버퍼로 사용하기 적당하지만 비교적 넓은 집적회로 면적을 필요로 한다. 특히 FPGA로 구현된 하드웨어의 경우, 한정된 자원과 지연시간의 최소화를 위해 내부 버퍼의 크기를 작게 유지하기 때문에 하나의 인공 신경망 연산 계층의 입출력 데이터를 다 담을 수 없는 경우가 발생한다. 이 경우 일부분만 내부 버퍼로 로드하고 연산결과를 부분적으로 생성하여 DRAM에 저장하는 방법을 사용하는 것이 보편적이며, 이를 타일링(Tiling) 이라 부른다.

인공 신경망 연산에서 타일링은 다차원 출력데이터의 각 축방향으로 이루어 진다. Convolution 연산의 출력이 3차원 데이터일 경우, H(Height)방향, W(Width)방향, C(Column)방향으로 타일링을 할 수 있다. 이 중, H와 W방향의 타일링은 로드 할 입력 데이터의 크기에, C방향의 타일링은 로드 할 가중치 데이터의 크기에 영향을 준다.

그림 3은 3차원 출력 데이터(a)의 각 축에 따른 타일링 방법으로, H방향 타일링(b), W방향 타일링(c), C방향 타일링(d), H와 W방향의 2중 타일링(e), HWC 방향의 3중 타일링(f)을 각각 나타낸다.

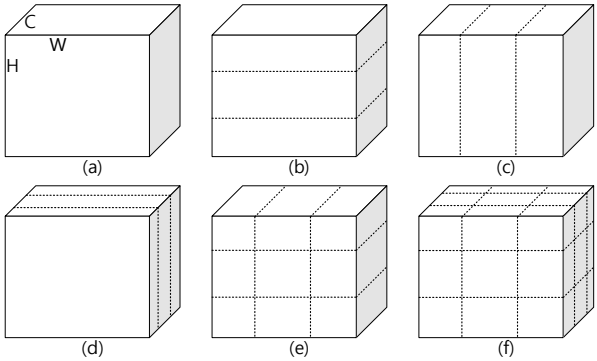

그림 3 출력데이터 타일링 방법

3.3 멀티 쓰레딩

VTA상에서의 신경망 연산은 LOAD -> COMPUTE-> STORE 모듈이 순차적으로 진행되어야 하며, 동일 타일에 대하여 이전 모듈의 수행이 종료되기 전에 다음 모듈이 동작해서는 안된다. 다수의 타일이 있는 경우 그림 4와 같이 각 타일을 순차적으로 연산할 수 있다.

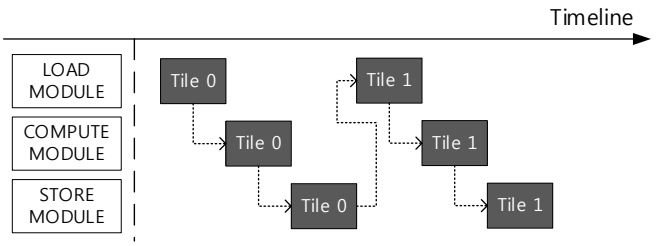

그림 4 Naive 타일링과 스케줄링

반면에 각 타일의 크기를 충분히 작게 하여 동시에 여러 타일이 버퍼 상에 존재할 수 있다면 그림 5와 같은 스케줄링이 가능하여 수행 시간을 줄일 수 있고, 이를 멀티 쓰레딩이라 한다.

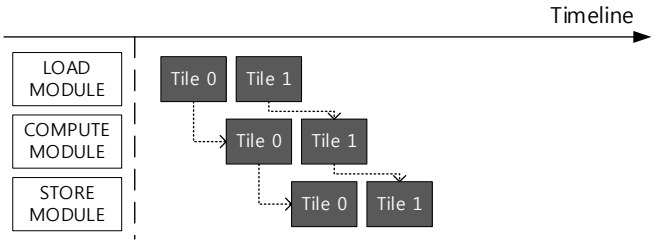

그림 5 최적화된 타일링과 스케줄링

4. 실험

4.1 실험 방법

표 1과 같은 임의의 Convolution with ReLU 계층을 VTA 환경에서 동작 시키고 총 메모리 연산과 수행시간을 측정한다.

표 1 실험 신경망 계층

입력 데이터	60(H) × 60(W) × 256(C)
Pad/Stride	1 / 0
가중치 데이터	256(N) × 3(H) × 3(W) × 256(C)
출력 데이터	60(H) × 60(W) × 256(C)

위 신경망 계층은 타겟 하드웨어의 내부 버퍼에 한번에 로드될 수 없을 정도로 크기 때문에 타일링이 필수적이다. 타일링 후 Naive 스케줄링의 성능과 멀티 쓰레딩 기법이 적용된 스케줄링의 성능을 비교한다.

4.2 실험 환경

- FPGA 보드: Digilent PYNQ-Z1
  - 650Mhz 듀얼코어 Cortex-A9
  - 로직 슬라이스: 13,300개
  - 블록 RAM: 630KB
  - DRAM: 512MB
- 구현된 VTA 딥러닝 가속 하드웨어
  - 클럭 주파수: 100 MHz
  - 입출력 데이터: 8비트 정수
  - 입/출력 데이터 버퍼 크기: 각 32KB
  - 가중치 데이터 버퍼 크기: 128KB
  - 16개의 산술연산 Initiation Interval: 2 cycle
  - 16X16 GEMM 연산 Initiation Interval: 1 cycle

4.3 실험 결과 및 분석

표 2는 각 타일링에 따라 Naive 스케줄링을 적용하여 LOAD 모듈이 로드 한 총 데이터 양과 수행시간을 나타낸다. STORE 모듈의 저장 데이터 양과 COMPUTE 모듈의 연산양은 해당 실험 신경망 계층에서 항상 일정하였다. 그 결과, 총 수행시간은 총 타일의 수가 아니라 로드데이터의 크기에 선형 증가함을 보여준다.

표 2 Naive 스케줄링 실험 결과

총타일수(H×W×C)	로드(MB)	수행시간(ms)
3×3×4 = **36**	10	150
4×4×2 = **32**	12	154
4×4×4 = **64**	14	164
6×6×1 = **36**	22	183
6×6×2 = **72**	24	189
6×6×4 = **144**	26	202
12×12×1 = **144**	87	363
12×12×2 = **288**	88	374
12×12×4 = **576**	92	399

표 3은 타일의 크기가 충분히 작아 멀티 쓰레딩 기법의 스케줄링을 적용가능한 타일링에 대한 수행시간 변화를 보여준다. 그 결과, 모든 경우 Naive 스케줄링보다 성능이 높아지고 Naïve 스케줄링의 최적코드 보다 데이터 로드가 1.6배 증가 했음에도 불구하고 수행시간이 30% 향상됨을 보여준다.

표 3 멀티 쓰레딩 스케줄링 실험 결과

타일(쓰레드 수)	로드(MB)	수행시간(ms)
6×6×4 = **144** (4)	26	202 -> 115
6×6×2 = **72** (2)	24	189 -> 116
12×12×2 = **288** (2)	88	374 -> 282
12×12×4 = **576** (4)	92	399 -> 298

5. 결론

본 논문에서는 FPGA 상에서 구현되는 딥러닝 가속 하드웨어의 구조를 설명하고 구조상의 한계점 때문에 타일링이 불가피함을 설명하였다. 또한 다양한 타일링 방법과 더불어 스케줄링을 통해 딥러닝 가속 하드웨어의 성능을 높일 수 있음을 보였다. 본 논문에서는 비교적 간단한 스케줄링의 결과만을 보였지만 딥러닝 가속 하드웨어용 컴파일러는 이러한 하드웨어의 특성을 바탕으로 고도화된 최적화 알고리즘을 적용하여 다양한 형태의 입력 데이터와 연산자를 대상으로 최적 코드를 생성해야 할 것이다.

참 고 문 헌

[1] VTA, https://tvm.apache.org/vta
[2] Chen, Tianqi, et al. "TVM: An automated end-to-end optimizing compiler for deep learning.", OSDI. 2018.
[3] Mingzhen Li, et al. The Deep Learning Compiler: A Comprehensive Survey, arXiv:2002.03794, 2020