행렬 곱셈을 위한 융합된 형태의 부동소수점 내적 연산 회로 구현

김찬훈, 박상수, 정기석*
한양대학교

kch1103@hanyang.ac.kr, po092000@hanyang.ac.kr, kchung@hanyang.ac.kr

Implementation of Floating-Point Fused Dot-Product Unit for Matrix Multiplication

Chan-Hoon Kim, Sang-Soo Park, Ki-Seok Chung*
Hanyang University, Seoul, Korea

요 약

본 논문은 곱셈과 덧셈 연산을 융합된 형태로 처리할 수 있는 부동소수점 벡터 내적 연산 회로를 제시한다. 벡터 내적 연산은 행렬 곱셈을 위한 기본 연산으로, 본 논문에서 제시하는 설계는 곱셈을 다 실행한 후, 부분 곱을 누적하는 형태가 아니라, 곱셈과 덧셈이 융합된 형태로 구현된다. 면적과 전력소모를 줄이기 위한 융합된 방식을 갖는 32 비트 부동소수점 벡터 내적 연산 회로는 오픈소스 공정인 Nangate 45nm cell library 을 사용하여 합성을 진행하였으며, 최적의 면적과 전력 소모를 갖는 회로는 그렇지 않은 경우에 비해 최대 1.21 배의 면적과 1.34 배의 전력 소모가 개선됨을 확인하였다.

Ⅰ. 서 론

행렬 곱셈은 통신과 이미지 프로세싱 등 매우 다양한 응용분야에서 가장 핵심적인 연산으로 사용된다. 최근에는 컴퓨터 비전, 음성인식 등 인공지능 기반의 어플리케이션에서 매우 자주 사용되고 있다. 특히, 인공지능의 학습 응용에서는 부동소수점 (floating-point) 수를 사용하는 경우가 많은데, 많은 양의 부동소수점 행렬 곱셈은 매우 복잡하기 때문에, 전력 소모와 성능의 최적화를 위해서 부동소수점 행렬 곱셈을 위한 하드웨어 가속기 설계 연구가 큰 주목을 받고 있다 [1].

행렬 곱셈의 알고리즘에는 내적과 외적이 존재한다. 내적 (Inner-product, Dot-product)은 벡터의 같은 위치 값끼리 곱하고 합하는 방식으로 연산이 이루어지게 되며 결과값이 벡터가 아닌 스칼라 값을 가지게 된다. 외적 (Outer-product)은 연산의 결과값이 벡터로 나오게 된다. 행렬 곱셈에서 내적 연산을 위한 하드웨어는 외적 연산을 위한 하드웨어 보다 구현이 간단하다는 특징을 가지고 있다 [2]. 이에 따라 본 논문에서는 내적을 이용한 행렬 곱셈을 위한 하드웨어 구현을 목표로 한다.

이러한 부동소수점의 내적을 위한 회로를 설계함에 있어서 곱셈을 먼저 다 실행하고, 이후에 부분 곱들을 누적하는 방식보다는 곱셈과 덧셈의 연산이 융합된 (fused) 방식으로 설계하는 것이 더 효과적이다. 부동소수점 덧셈, 뺄셈 곱셈 및 나눗셈 연산은 복잡한 연산 과정을 가지고 있으며, 융합된 방식이 아닌 분리된 곱셈기와 덧셈기를 결합하는 방법의 설계는 면적 및 전력 소모 측면에서 한계가 있을 수 있다 [2]. 이러한 부분을 개선하기 위해 최적의 융합된 설계를 통해 개선해야 한다.

본 논문에서는 8*8 행렬들의 곱셈 연산을 위한 8 개의 값을 입력으로 하는 내적 연산 하드웨어를 제시하며, Verilog HDL 을 사용하여 하드웨어를 구현하였고, 면적과 전력 소모가 개선이 가능함을 확인하였다.

Ⅱ. 본론

2.1 행렬 곱셈을 위한 내적 및 외적 알고리즘

행렬 곱셈 방법으로는 내적과 외적을 통한 방법이 사용되고 있으며, 그 중에서는 내적의 방법이 가장 일반적으로 사용되고 있다. 내적은 그림 1 과 같이 행렬 A 의 행 벡터 (Row i)와 행렬 B 의 열 벡터 (Column j)를 곱하고 더하여 결과 행렬의 각 값을 계산하는 방법으로, 각 행 벡터와 각 열 벡터의 같은 위치에 있는 값끼리 곱하고, 이들의 합을 계산하여 결과 곱 행렬의 하나의 값 (스칼라)을 결정한다. 외적은 행렬 A 의 열 벡터와 행렬 B 의 행 벡터를 연산에 사용하는 방법으로, 내적과는 달리 결과 값 행렬 C 의 결과 값을 여러 개 (벡터) 계산한다. 이러한 내적 및 외적 알고리즘을 하드웨어로 구현하는 경우, 결과 값 행렬 C 를 계산하는 과정에서 별도의 버퍼를 필요로 한다 [3].

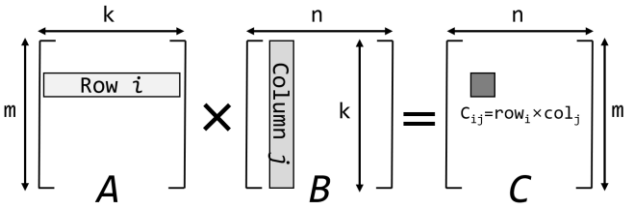
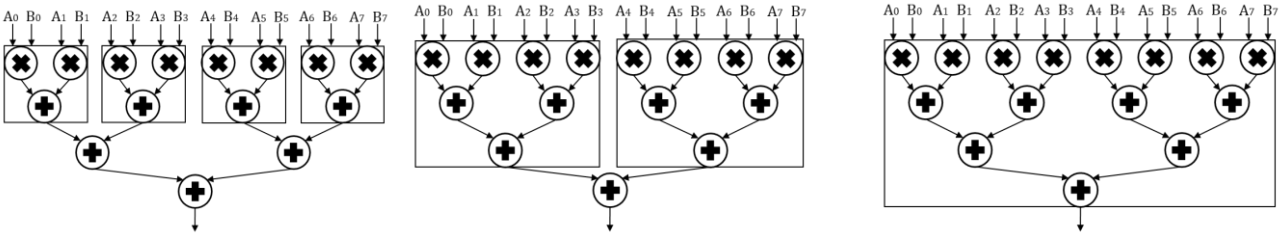


그림 1. 내적을 활용한 행렬 곱셈의 예

2.2 내적 연산을 수행하는 하드웨어

본 논문에서는 내적 연산을 통한 행렬 곱셈을 수행하는 것을 목표로 하며, 이를 위해 8 개의 벡터를 입력으로 하는 내적 연산을 수행하는 하드웨어를 설계하였다. 부동소수점을 이용하여 내적 연산을 하는 하드웨어는 융합 방법에 따라 연산 성능과 비용 (면적 및



(a) 융합 설계 버전 1 (b) 융합 설계 버전 2 (c) 융합 설계 버전 3

그림 2. 융합된 형태의 부동소수점 내적 연산기 구조

전력 소모)이 결정된다. 따라서, 곱셈과 덧셈이 어떻게 적절히 융합되어 설계되는가에 따라 최적의 연산 성능과 낮은 비용을 얻는 것이 가능하다. 8 개의 벡터를 입력으로 하는 부동소수점 내적 연산은 그림 2 (a)와 같이 총 4 단계의 연산이 필요하다(곱셈->덧셈->덧셈->덧셈). 본 논문에서는 부동소수점 곱셈기와 덧셈기로는 연산 성능 및 전력 소모 등에서 뛰어나다고 알려진, Wallace tree 곱셈기 [4]와 Kogge Stone 덧셈기 [5]를 각각 사용하였다. 이 부동소수점 곱셈기와 덧셈기를 활용하여 세 종류의 융합 설계로 회로를 구현하였으며 융합 설계를 하지 않고, 4 단계 파이프라인만이 적용된 회로와 비교하였다.

그림 2 (a)는 곱셈 2 번과 덧셈 1 번의 융합 설계가 적용된 구조를 보여주고 있다. 이러한 구조에서는 critical path 의 delay 가 다른 융합 설계 방법이 적용된 회로에 비해 짧기 때문에 높은 동작 주파수를 얻을 수 있게 된다.

그림 2 (b)는 곱셈 4 번과 덧셈 3 번의 융합 설계가 적용된 구조를 보여주고 있다. 융합 설계 방법이 적용되지 않은 구조에 비해 동작 주파수의 감소 폭을 최대한 줄이면서, 면적과 전력 소모 면에서 가장 이득을 볼 수 있는 것을 확인하였다.

그림 2 (c)는 곱셈 8 번과 덧셈 7 번의 융합 설계 방법이 적용된 구조이며, 많은 연산이 하나의 클럭 주기에 실행되어야 하기 때문에 어쩔 수 없이 가장 낮은 동작 주파수를 가지게 된다고 볼 수 있다.

2.3 실험 결과 및 분석

본 논문의 실험을 위해 8 개의 32 비트 IEEE-754 단정밀도 부동소수점 값을 입력으로 하여 내적 연산을 수행하는 하드웨어를 설계하였다. Verilog HDL 을 사용하여 4 종류의 내적 연산을 수행하는 하드웨어를 설계하였으며, Synopsys 社의 Design Compiler v2020.09-SP5-4 를 사용하여 오픈소스 셀 라이브러리 Nangate 45nm 를 셀 라이브러리로 합성을 진행하였다.

표 1 은 융합 설계 방식에 따른 내적 연산을 수행하는 하드웨어의 합성 결과를 나타내며, 융합 설계 방식에 따라 면적, 전력 소모가 영향을 받음을 알 수 있다. 융합 설계 방식을 하지 않은 4 단계 파이프라인 구조가 적용된 회로는 각 클럭 주기에 곱셈 또는 덧셈의 단일 연산만 수행하기 때문에 다른 융합 설계 방법이 적용된 회로에 비해 높은 동작 주파수를 얻을 수 있다는 장점이 있지만, 연산 완료까지 소요되는 클럭 사이클의 수가 4 로 가장 크고, 면적과 전력 소모 면에서 가장 높게 나온 것을 확인하였다. 곱셈 2 번과 덧셈 1 번의 융합 설계가 적용된 회로와 전체가 다 융합된 형태로 구현된 설계는 각각 1.09 배, 1.28 배의 전력 소모가 개선되지만 면적은 크게 개선되지 않는 것을 확인하였다. 곱셈 4 번과 덧셈 3 번의 융합 설계가 적용된 회로는 그렇지 않은 회로에 비해 1.21 배의 면적과 1.34 배의 전력 소모가 개선됨을 확인하였다.

Ⅲ. 결론

본 논문에서는 매우 많은 응용에서 핵심 연산이 되는 부동소수점 행렬 곱셈에서 기본 연산에 해당하는 벡터 간의 부동소수점 내적 연산기를 4 종류의 하드웨어로 설계하였다. 내적 연산에서는 곱셈 연산과 덧셈 연산이 어떻게 융합된 형태로 설계되는가에 따라 내적 연산기의 성능 및 전력 소모가 영향을 받으며, 각 구현된 설계의 성능, 면적, 전력 소모 등의 평가를 통해, 곱셈 4 번, 덧셈 3 번의 융합 설계된 구조가 융합 설계가 적용되지 않은 일반적인 구조 대비 최대 1.21 배의 면적 감소와 1.34 배의 전력 소모를 개선할 수 있는 것을 확인하였다.

표 1. 융합 설계 방식 별 합성 결과 비교

	No fused	Fused v1	Fused v2	Fused v3
Frequency (MHz)	250	227	222	192
Power (mW)	35.35	32.41	26.39	27.53
Area (um^2)	177,673	177,028	146,801	177,044

ACKNOWLEDGMENT

이 논문은 2020 년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임(No. 2020-0-01304, 모바일 자가 학습 가능 재귀 뉴럴 네트워크 프로세서 기술 개발).

참 고 문 헌

[1] Jongwook Sohn, Earl E.Swartzlander, Jr. “Improved Architectures for a Floating-Point Fused Dot Product Unit” IEEE 21st Symposium on Computer Arithmetic 2013.

[2] Hani H.Saleh, Earl E.Swartzlander, Jr. “A Floating-Point Fused Dot-Product Unit” IEEE 2008.

[3] Zhekai Zhang “SpArch: Efficient Architecture for Sparse Matrix Multiplication” The 26th IEEE HPCA 2020.

[4] Kokila Bhartu Jaiswal, “Low Power Wallace Tree Multiplier Using Modified Full Adder” ICSCN 2015.

[5] U Penchalaiah “Design of High-Speed and Energy-Efficient Parallel Prefix Kogge Stone Adder” IEEE ICSCA 2018.