투고용 논문 2021

효율적인 Capsule Network 추론 시스템의 High-Level Synthesis 설계 및 FPGA 구현

(High-Level Synthesis Design and Implementation of an Efficient Capsule Network Inference System in an FPGA)

양해찬*, 박상준*, 박관영*, 사재현*, 김태환**

(Hae-Chan Yang, Sang-Jun Park, Kwoan-Young Park, Jae-Hyun Sa and Tae-Hwan Kim ⓒ)

요 약

Capsule Network (CapsNet) 모델은 등가성을 가지기 때문에 이미지의 방향 및 크기 등의 변화에 강인한 추론이 가능하다고 알려져 있다. 하지만 추론 알고리즘의 높은 연산 복잡도로 인해 한정된 자원을 가진 FPGA에서의 효율적인 구현이 어렵다. 본 연구는 한정된 자원을 갖는 FPGA에서의 CapsNet 기반의 효율적인 추론 시스템을 제안한다. 제안하는 시스템을 위하여 추론 알고리즘을 고정 소수점 최적화하였고, 활성화 함수의 변형을 통해 초월 함수의 사용을 피했다. 또한, 추론 알고리즘의 일부 병목 단계들을 고속으로 수행하기 위한 전용의 Component들을 구현하여, 시스템에 집적하였다. 제안된 CapsNet 추론 시스템은 6348개의 ALM, 163K-bit BRAM, 3개의 DSP로 Intel Cyclone V FPGA에서 구현되었다. 순수 소프트웨어 구현 대비 193배 빠른 추론 속도를 보이며, 2.35 GOP/s/DSP를 달성한다.

Abstract

Capsule Network (CapsNet) models are known to be feasible for the robust image classifications, inherently having equivariance characteristics. However, its high computational workload involved in the inference algorithm hinders an efficient implementation. This study proposes an efficient CapsNet inference system in a resource-limited FPGA. The inference algorithm has been optimized for the fixed-point arithmetics. The complicated arithmetic operations such as the division and square-root have been eliminated effectively by modifying the activation functions. In addition, dedicated hardware components to perform time-critical steps have been designed and integrated. The proposed system has been implemented in Intel Cyclone V FPGA with 6348 ALMs, 163K-bit BRAM and 3 DSPs. The proposed system shows 193× faster inference speed, compared to the software implementation, achieving 2.35 GOP/s/DSP.

Keywords: Inference System, Capsule Network, FPGA, High-Level Synthesis, Artificial Intelligence.

Ⅰ. 서 론

Capsule Network(CapsNet)는 기존 인공 신경망 모델들과 달리 뉴런들을 하나의 벡터로 묶은 캡슐[1]을 단위로 추론 과정을 진행하는 모델이다[2]. 캡슐들은 추출한 특징의 위치, 크기, 속도, 질감 등 다양한 정보를 성분으로 가지고 있다[1]. CapsNet 모델을 기반으로 하는 추론 알고리즘에서는 동적 라우팅을 통해 하위 캡슐인

*학생회원, **평생회원, 한국항공대학교
(School of Electronics & Computer Engineering, Korea Aerospace University)
ⓒ Corresponding Author(E-mail: taehwan.kim@kau.ac.kr)
※ 본 연구는 경기도의 경기도 지역협력연구센터 사업의 일환으로 수행하였고 [GRRC항공2017-B06, 지능형 이동 보조 수단을 위한 센서 데이터 처리 시스템 연구] 2017년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 [2017-0-00528, 다중 모드 스마트 레이다용 지능형 반도체 개발 기초 연구실]. EDA 도구는 IDEC의 지원을 받았음.
접수일자: 2021년 8월 10일, 수정일자: 2021년 x월 xx일
게재확정: 2021년 9월 24일

PrimaryCaps에서 상위 캡슐인 DigitCaps로 연결되는데, 이 과정에서 캡슐 간 공간 정보가 조합된다[2]. 이를 통해, 이미지의 방향 및 크기 등의 변화에 강인한 추론이 가능하다[2]. 하지만 CapsNet 추론 알고리즘은 내부적으로 초월 함수 계산을 포함하고 있으며, 동적 라우팅이 반복적으로 수행되어야 하기 때문에 높은 워크로드를 갖게 되며, 이에 따라 한정된 자원을 갖는 FPGA에서의 효율적인 구현이 어렵다. 따라서 이를 위한 효율적인 CapsNet 추론 시스템 설계가 필요하다.

이미지 분류를 위해 Convolutional Neural Network(CNN) 모델 기반의 다양한 추론 알고리즘이 도입되고 있으며, 이를 가속하기 위한 다양한 연구가 진행되었다[3-5]. 참고문헌 [3]에서는 채널에 따라 입력 Feature를 분리해 각각 Convolution 연산을 수행하는 Depthwise Separable Convolution 기법을 통해 파라미터 수와 연산 복잡도를 크게 줄이고, Matrix Multiplication Engine을 통해 추론 속도를 향상시켰다. 참고문헌 [4]에서는 Convolution 연산의 입력 데이터를 Systolic 기반으로 전파하였고, 병렬 연산과 메모리 대역폭 최적화를 통해 높은 추론 속도를 보인다. 참고문헌 [5]에서는 데이터 통신 및 메모리 접근을 최소화하고, 효율적인 Convolution 연산을 위해 Unrolling, Tiling, Interchange 기법을 적용해 루프를 최적화하였다.

이미지 분류를 위해 앞서 소개한 CapsNet 모델을 기반으로 하는 추론 알고리즘의 효율적인 시스템에 대한 연구들도 진행되었다[6-10]. 이 중 참고문헌 [6], [7]에서는 소프트맥스 활성화 함수와 스쿼시 활성화 함수를 활성화 유닛에서 큰 크기의 룩업 테이블을 사용해 구현하였다. 참고문헌 [8]에서는 스쿼시 활성화 함수에서의 L^2-norm 근사화[11], 그룹 Convolution 병렬 연산 기법, 동적 라우팅 생략 유닛 등을 통해 높은 에너지 효율성을 달성하였다. 참고문헌 [9]에서는 다차원 병렬 선택, 지능형 워크로드 분배, 새로운 주소 매핑 메커니즘 등을 적용하였다. 참고문헌 [10]에서는 Off-Chip 메모리 접근을 최소화하기 위해 CapsNet 추론 알고리즘에 최적화된 On-Chip Scratchpad 메모리를 도입하였다. 위에 언급한 연구 결과들은 에너지 효율과 처리 속도 관점에서는 높은 성능을 보이지만 자원 효율성에 대해서는 심각하게 고려하지 않았기에, 한정된 자원을 갖는 FPGA를 위한 적용 가능성에는 한계가 있다.

본 연구는 CapsNet 추론 알고리즘의 최적화와 전용 하드웨어 Component에 기반한 효율적인 CapsNet 추론 시스템의 High-Level Synthesis(HLS) 설계 및 구현에 대한 것이다. 본 연구의 주요 사항은 다음과 같다.

1) 효율적인 구현을 위해 CapsNet 추론 알고리즘의 고정 소수점 최적화를 수행하였고, 활성화 함수를 근사화하여 초월 함수의 계산을 제거하였다. 또한, 추론 정확도가 유지되는 조건에서 동적 라우팅 반복 횟수를 최적화하였다.

2) CapsNet 추론 알고리즘의 병목으로 분석된 일부 단계들을 고속으로 수행하기 위한 전용의 Component들을 HLS를 이용하여 합성하였고, 이들을 내장한 효율적인 추론 시스템을 설계하였다. 전체 시스템을 Intel Cyclone V FPGA에서 구현한 결과 6348개의 ALM, 163K-bit BRAM, 3개의 DSP의 적은 자원 사용으로 구현되었으며, 2.35 GOP/s/DSP의 효율성을 달성한다. 또한 순수 소프트웨어 구현 대비, 193배 빠른 추론 속도를 달성하였다.

본 논문의 나머지 부분은 다음과 같이 구성된다. Ⅱ장에서는 CapsNet 추론 알고리즘을 설명하고, 각 단계별 연산 복잡도와 수행 시간을 분석한다. Ⅲ장에서는 효율적인 구현을 위한 CapsNet 추론 알고리즘 최적화 방법과 CapsNet 추론을 위한 HLS 설계 및 FPGA 구현에 대해 설명한다. Ⅳ장에서는 제안하는 CapsNet 추론 시스템의 구현 결과를 기존 연구 결과와 비교·분석해 본 논문에서 제안하는 추론 시스템의 효율성을 입증한다. Ⅴ장에서 본 논문의 결론을 도출한다.

Ⅱ. CapsNet 추론 알고리즘

본 절에서는 CapsNet 모델을 기반으로 하는 추론 알고리즘에 대해 설명한다. 그림 1은 MNIST 데이터셋 이미지 분류 작업을 위한 CapsNet 추론 알고리즘[2]을 보여주고 있으며, 다음과 같은 단계들로 구성된다.

Step 1: Convolution 및 활성화를 통해 하위 캡슐인 PrimaryCaps ($\mathbf{u}_i$, $1 \leq i \leq 1152$) 에 해당하는 Feature를 추출한다;

Step 2: 가중치 ($\mathbf{W}_{ij}$) 를 이용해 Prediction 벡터 ($\hat{\mathbf{u}}_{j|i}$, $1 \leq j \leq 10$) 를 구한다;

Step 3: 소프트맥스 활성화 함수를 통해 i번째 하위 캡슐(PrimaryCaps)과 j번째 상위 캡슐(Pre-DigitCaps) 간 연결 강도를 나타내는 Coupling Coefficient (c_{ij}) 를 Logit (b_{ij}) 기반으로 도출한다;

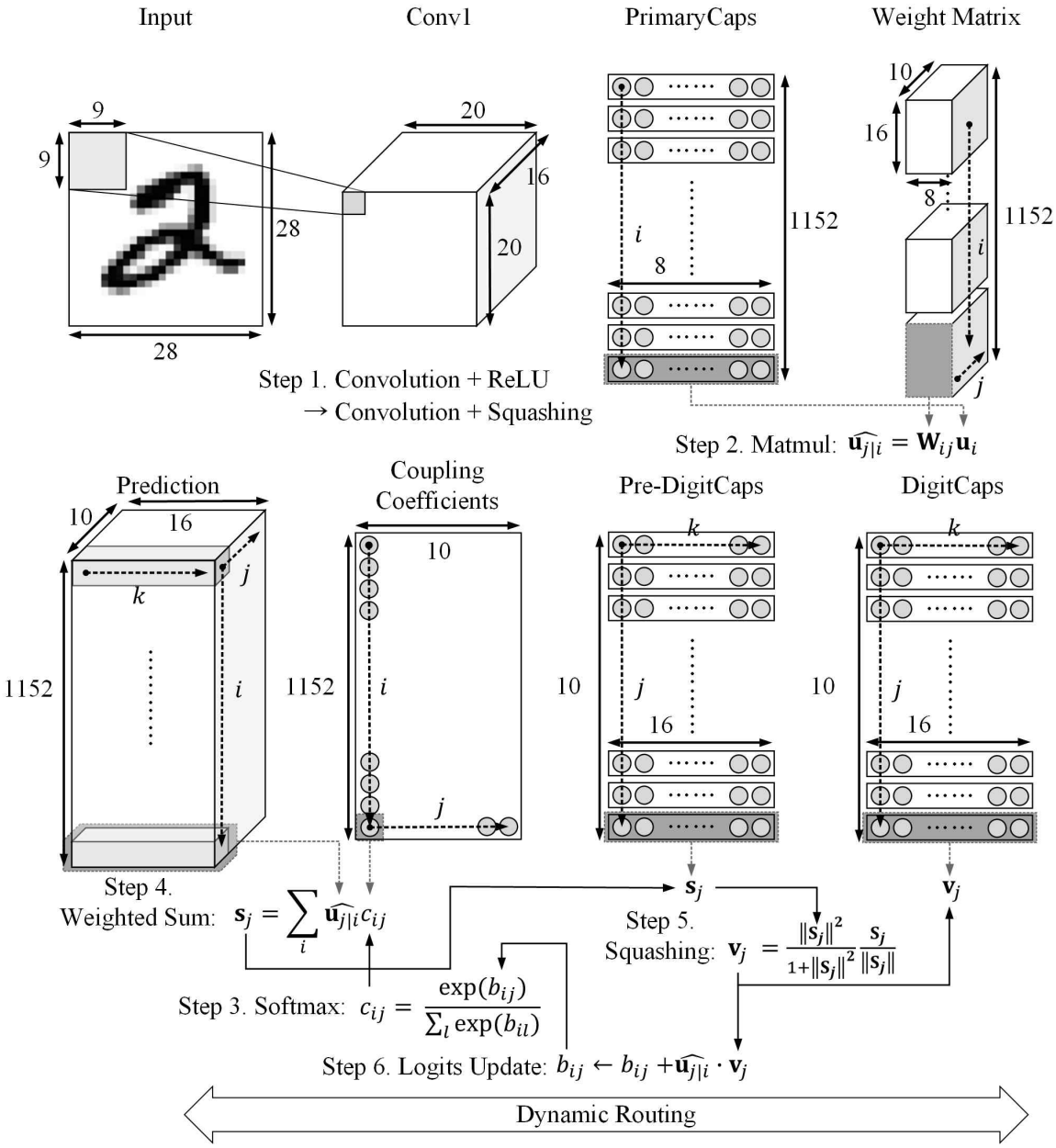


그림 1. CapsNet 추론 알고리즘의 전체 과정
Fig. 1. Overall flow of CapsNet inference algorithm.

Step 4 - 5: c_{ij}를 기반으로 Prediction 벡터의 가중합을 통해 Pre-DigitCaps ($\mathbf{s}_j$) 를 구하고, 스쿼시 활성화를 통해 DigitCaps ($\mathbf{v}_j$) 를 구한다;

Step 6: Prediction 벡터와 DigitCaps를 이용해 b_{ij}를 갱신한다. 여기서, Step 3 - 6을 반복하여 상·하위 캡슐 간 관계를 갱신하는 과정을 동적 라우팅(Dynamic Routing)이라고 하며, 여기서 b_{ij}는 0으로 초기화 한다 [2]. 그림에서 • 는 벡터 간 내적 연산을 나타낸다. 최종적으로, DigitCaps의 크기를 기반으로 클래스 별 확률을 도출한다.

CapsNet 추론 알고리즘을 구성하는 각 단계별 워크로드를 그림 2에서 분석하였다. 워크로드는 수행에 필요한 총 연산량(Operation Count; OP)관점에서 분석되었으며, 덧셈을 1 OP, 곱셈과 나눗셈을 4 OP, 제곱근 연산을 6 OP, 지수 함수 연산을 8 OP로 추산하였다.

그림에서 보는 바와 같이, PrimaryCaps를 구하기 위한 Step 1의 연산량이 압도적으로 큰데, 이는 Step 1의 Convolution 연산의 복잡성에 기인한다. 또한, 동적 라우팅을 구성하는 단계들 (Step 3 - 6)의 경우 나눗셈, 제곱근, 지수 함수 등의 연산을 포함하고 있기 때문에, 반복 횟수가 클 경우, 전체 워크로드에서 차지하는 비

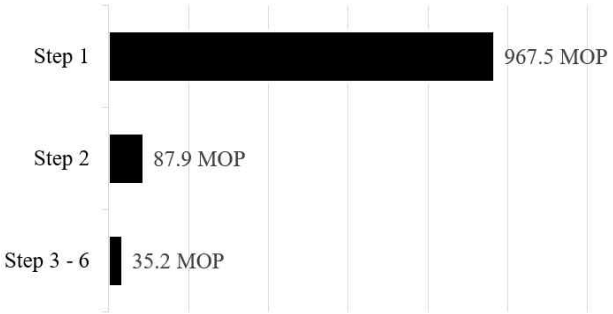


그림 2. CapsNet 추론 알고리즘에서의 단계별 Workload
Fig. 2. Workload for each step in CapsNet inference algorithm.

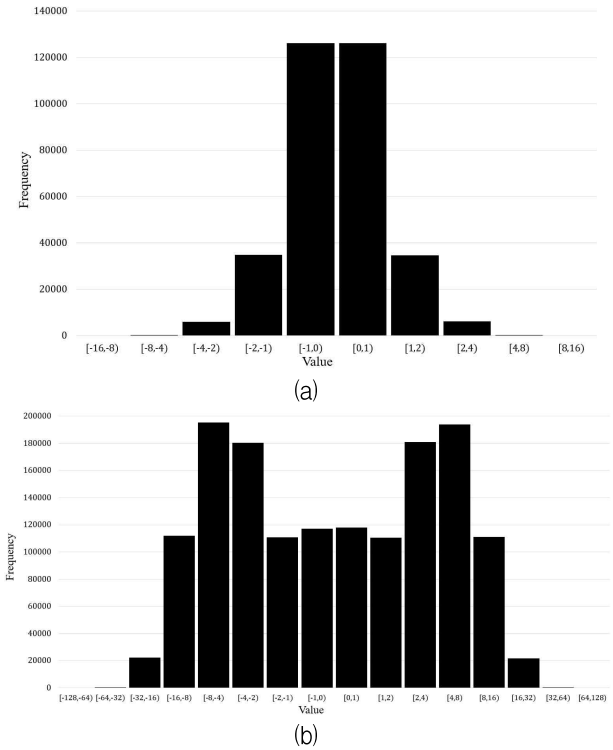


그림 3. 단계별 입력 Parameter의 분포: (a) Step 1, (b) Step 2
Fig. 3. Stepwise distribution of input parameters: (a) Step 1, (b) Step 2.

중이 커질 것이다. 따라서, 추론 결과에 유의미한 영향을 주지 않는 범위에서 동적 라우팅을 구성하는 복잡한 연산을 단순화하는 것과 더불어, 반복 횟수를 줄일 필요가 있다.

Ⅲ. 제안하는 CapsNet 추론 시스템

본 절에서는 효율적인 구현을 목표로 한 CapsNet 추론 알고리즘의 최적화와 FPGA를 기반으로 하는 추론 시스템의 HLS 설계 및 구현에 대해 설명한다.

1. 효율적인 구현을 위한 CapsNet 추론 알고리즘 최적화

제한된 자원을 갖는 FPGA에서의 효율적인 시스템 구현을 목표로 다양한 방법을 도입하여 CapsNet 추론 알고리즘을 최적화하였다. 먼저, 메모리 요구량과 로직 복잡도를 낮추기 위하여 고정 소수점 기반의 수 체계를 도입하였다. 부동 소수점 기반의 연산은 높은 정확도와 표현 범위를 갖는 반면, 로직 복잡도가 높기 때문에 제한된 자원을 갖는 FPGA에서의 효율적인 시스템 구현에 장애가 된다. 그림 3은 알고리즘의 단계별 입력 파라미터의 분포를 정리한 것이다. 그림에서 보여지는 데이터 분포에 따라, Step 1에서 모든 입력 및 파라미터를 정수부 4 bit, 소수부 4 bit의 고정 소수점 수 체계를 도입하여 표현하였으며, Step 2에서는 모든 입력 및 파라미터를 정수부 6 bit 소수부 2 bit의 수 체계를 도입하여 표현하였다. Step 2 이후로는 8 bit 동적 고정 소수점 수 체계를 도입했다. 이를 통해 32 bit 부동 소수점 수 체계를 기반으로 하는 구현과 비교하여 전체 메모리 요구량을 1/4 수준으로 줄일 수 있으며, 로직 복잡도 감소를 기대할 수 있다.

최적화된 동적 라우팅 반복 횟수를 통해 소프트맥스 활성화 함수의 복잡한 지수 함수 연산을 제거하였다. 동적 라우팅 과정 중 Step 3에서 소프트맥스 활성화 함수를 수행하는데 이는 지수 함수 연산을 포함한다. 지수 함수는 기본적으로 긴 처리 지연시간을 가지며, 전용의 하드웨어로 구현할 경우 로직 복잡도가 높다[6-8]. 앞 절에서 설명한 알고리즘[2]에서 동적 라우팅의 반복 횟수를 가변하여 실험한 결과, 3회의 동적 라우팅 반복을 통해 달성한 추론 정확도와 1회의 반복을 통해 달성한 추론 정확도의 차이는 0.05% 일 정도로 미미하다는 사실을 발견하였다. 이를 기반으로, 복잡한 지수 함수 연산을 포함하고 있는 동적 라우팅의 반복 횟수를 1회로 제한하였다. 이에 따라, 소프트맥스 계산에 필요한 지수 함수 연산을 효과적으로 제거할 수 있게 되고, Pre-DigitCaps를 구하기 위한 Coupling Coefficient와 Prediction 벡터 간의 연산을 단순 비트 이동 연산으로 대체할 수 있게 된다.

추가적으로, 스쿼시 활성화 함수의 근사화를 통해 제곱근 및 나눗셈 연산을 제거하였다. 추론 알고리즘의 Step 1과 Step 5에서 사용되는 스쿼시 활성화 함수는 벡터의 크기가 클 경우 1에 수렴하고, 벡터의 크기가 작을 경우 0에 수렴하게 하는 성질을 갖는다. 여기서

Step 5 \ Step 1	5bit Shift	6bit Shift	7bit Shift	8bit Shift
5bit Shift	86.65%	92.12%	97.12%	97.75%
6bit Shift	94.71%	96.87%	98.02%	97.95%
7bit Shift	97.95%	97.98%	98.17%	97.96%
8bit Shift	97.95%	97.96%	97.97%	97.86%

표 1. Step 1과 5에서의 비트 이동 기반 스쿼시 활성화 함수에 따른 추론 정확도

Table 1. Inference accuracy for the shift-based squash activation function in Steps 1 and 5.

Algorithm (a)

for all capsule i in layer m and capsule j in layer $m+1$ do
  $b_{ij} \leftarrow 0$
for r iterations do
  for all capsule i in layer m
    $c_{ij} \leftarrow \frac{\exp(b_{ij})}{\sum_l \exp(b_{il})}$ ▶ Softmax
  for all capsule j in layer $m+1$
    $\mathbf{s}_j \leftarrow \sum_i \hat{\mathbf{u}}_{j|i} c_{ij}$
  for all capsule j in layer $m+1$
    $\mathbf{v}_j = \frac{\|\mathbf{s}_j\|^2}{1+\|\mathbf{s}_j\|^2} \frac{\mathbf{s}_j}{\|\mathbf{s}_j\|}$ ▶ Squash
  for all capsule i in layer m and capsule j in layer $m+1$
    $b_{ij} \leftarrow b_{ij} + \hat{\mathbf{u}}_{j|i} \cdot \mathbf{v}_j$
return $\mathbf{v}_j$

Algorithm (b)

for all capsule j in layer m
  $\mathbf{s}_j \leftarrow \sum_i \hat{\mathbf{u}}_{j|i}$
for all capsule j in layer m
  $\mathbf{v}_j \leftarrow \mathbf{s}_j \gg (n+4)$ ▶ Softmax & Squash
return $\mathbf{v}_j$

표 2. 알고리즘 (a): 기존 동적 라우팅 알고리즘[2], 알고리즘 (b): 최적화 방법을 도입한 동적 라우팅 알고리즘

Table 2. Algorithm (a): Previous dynamic routing algorithm, Algorithm (b): Dynamic routing algorithm introduced optimization method.

벡터의 크기를 구하기 위해 L^2-norm을 포함하고 있는데, 이는 제곱근 연산과 크기 정규화를 위한 나눗셈 연산으로 이루어진다. 기존의 연구에서는 이러한 제곱근 연산을 피하기 위하여, L^2-norm을 L^1-norm과 $L^\infty-norm$의 선형 결합으로 근사하여 스쿼시 활성화 함수를 계산하도록 구현한 사례가 있으나[8], 나눗셈 연산은 여전히 수행되어야 한다. 제안하는 방법은 스쿼시 활성화 함수의 입력 벡터의 데이터 분포에 기반하여 이를 일괄적인 비트 이동을 통해 스케일링하여 큰 벡터의 경우 1에 가깝게, 작은 벡터의 경우 0에 가깝게 변환된다. 스케일링 계수의 경우 표 1에 정리된 결과에 기반하여 실험적인 방식을 통해 결정하였다.

이와 같은 최적화 방법을 도입하여 낮은 처리 지연시간 및 로직 복잡도 감소를 기대할 수 있으며, 이에 대한 타당성은 다음 절에서 시스템 구현 결과 및 성능으로 증명할 것이다. 표 2는 기존 동적 라우팅 알고리즘[2]과 본 절에서 설명한 최적화 방법을 도입한 동적 라우팅 알고리즘을 보여준다. 표 2에서 볼 수 있듯이, 특히 동적 라우팅 과정에 포함되는 지수 함수, 제곱근, 나눗셈 연산을 제거하여 해당 과정에서의 워크로드를 기존 방식의 33.7 MOP에서 3.7 MOP으로 89%나 줄일 수 있었다. 주목할 만한 점은, 이러한 최적화 방법을 도입함에도 불구하고 추론 정확도의 차이는 기존 대비 0.84 %로 미미하다는 것이다.

2. CapsNet 추론을 위한 HLS 설계 및 FPGA 구현

CapsNet 추론 알고리즘을 효율적으로 수행하기 위한 시스템을 HLS를 이용해 설계하였다. 제안하는 시스템은 Intel Cyclone V FPGA를 기반으로 설계되었다. 그림 4는 제안하는 시스템의 전체 구조를 보여주고 있다. FPGA 외부 64MByte 크기의 SDRAM은 CapsNet 모델의 파라미터와 Feature 데이터를 저장한다. Convolution과 Matrix Multiplication Component는 앞 절에서 설명한 추론 알고리즘의 Step 1 – 2 부분을 담당하고 있는 전용 Component 들로서, 이에 대한 자세한 내용은 후술한다. Microprocessor는 전술한 Component 들을 제어하여 추론 알고리즘의 전체적인 흐름 제어를 담당하며, Step 1 – 2를 제외한 나머지 부분에 해당하는 연산을 수행한다.

앞 절에서 설명한 최적화 방법을 적용한 CapsNet 추론 알고리즘의 수행 시간을 그림 4에 도시된 시스템을 기반으로 분석하였다. 정밀한 수행 시간 측정을 위해서, Performance Counter를 도입하였다. 표 3은 추론 알고리즘의 수행 시간을 각 단계별로 분석한 것이다. 해당 표에 정리된 결과는 Step 1 – 2를 포함하여 추론 알고리즘의 모든 연산을 Microprocessor로 처리하여 추산된 것이다. 여기서 주목할 만한 것은, Step 1 – 2의 수행 시간이 전체 수행 시간에서 압도적인 비중을 차지하는 병목이며, 나머지 단계들의 경우 앞 절에서 제안한 최적화 방법을 통해 수행 시간 측면에서 상대적으로 낮은 비중을 차지하고 있다는 점이다.

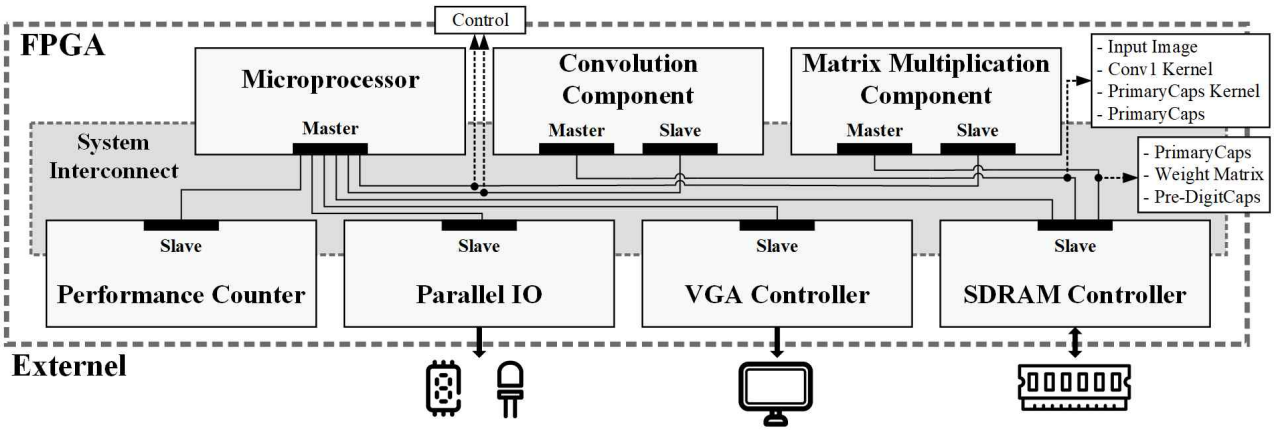


그림 4. 제안하는 시스템의 전체 구조
Fig. 4. Overall architecture of the proposed system.

	Step 1	Step 2	Step 3 - 6 (Dynamic Routing)	Total
Execution Time (ms)	71704	4213	133	76067

표 3. CapsNet 추론 알고리즘의 수행 시간 분석
Table 3. Analysis of the execution time of CapsNet inference algorithm.

수행 시간의 분석 결과에 기반하여 Step 1 - 2를 처리하기 위한 전용 Component를 도입하였다. 앞 절에서 설명한 바와 같이, Step 1은 Feature 추출을 위해 반복적인 Convolution과 활성화 과정으로 이루어지며, Step 2는 Prediction 벡터를 구하기 위한 Matrix Multiplication으로 이루어진다. 이러한 내부 수행 과정을 표출하도록 각 Component의 이름들을 정했다. 해당 단계들의 수행 과정을 C++ 언어를 사용하여 상위 수준으로 기술하였고, 이를 HLS Tool을 통해 HDL로 합성하여 전체 시스템에 효과적으로 통합되도록 하였다. 그림 4에서 볼 수 있듯이, Microprocessor는 Master Interface를 통해 Convolution, Matrix Multiplication Component의 동작을 제어한다. 해당 Component 들은 Master Interface를 갖도록 설계되어, 피연산자와 연산 결과에 해당하는 데이터를 직접적으로 메모리에 접근하도록 한다. 이러한 직접 메모리 접근 기능을 통해 Microprocessor를 통한 불필요한 데이터 복사를 최소화하면서 해당 과정들을 효율적으로 수행하게 된다.

Convolution, Matrix Multiplication Component는 외부 SDRAM에 저장된 대량의 데이터를 피연산자로 사용하고, 연산 결과도 해당 메모리에 저장하기 때문에, 해당 데이터의 접근을 최적화시킬 필요가 있다. 해당 데이터는 Tensor 형태로서, 연속적인 메모리 영역에 저장된다. 이런 성질을 이용하여 외부 메모리에 저장된 데이터의 요소들이 필요할 때마다 개별적으로 접근하지 않고, Burst 단위로 모아서 (Burst Coalescing) 접근하도록 외부 Memory의 특성을 반영하여, Variable Latency 옵션을 사용하고 Maxburst 값을 128로 설정하였다. Convolution Component의 처리 지연시간은 137ms이며, Matrix Multiplication Component의 경우 119ms이다. 해당 Component를 이용하여 추론을 수행한 결과, 표 3에서의 Step 1 - 2의 처리 시간이 각각 99.8%, 97.2% 단축되며, 전체 수행 시간은 99.5% 단축된다.

IV. 구현 결과

제안하는 CapsNet 추론 시스템을 Intel Cyclone FPGA (5CSEMA5)를 사용하여 구현하였다. 전체 추론 알고리즘은 C++로 기술되어 해당 시스템에 집적된 Microprocessor에서 수행되도록 컴파일 되었고, Convolution Component와 Matrix Multiplication Component는 해당 FPGA를 위한 HLS Tool인 i++를 이용하여 Verilog HDL 형태로 만들어졌으며, 전체 시스템은 Quartus Prime v18.1을 이용하여 합성되었다. 전체 시스템은 1.1V Supply, 85도의 조건에서 100MHz의 Clock으로 동작하도록 하여, 해당 타이밍 조건을 성공적으로 만족하였다. 표 4는 제안하는 시스템의 FPGA 자원 사용량을 분석한 것이다. Convolution Component와 Matrix Multiplication Component의 로직 자원 사용량은 각각 3190 ALM, 2 DSPs 와 3158 ALM, 1 DSP이다. 해당 FPGA에서의 가용 로직 자원 대비 전체 시스템 구현에 사용된 자원 비율은 ALM 기준으로 20%,

	ALM	BRAM(K-bit)	DSP
Convolution Component	3190	117	2
Matrix Multiplication Component	3158	46	1

표 4. 제안하는 시스템의 FPGA 자원 사용량
Table 4. FPGA resource usage of the proposed system.

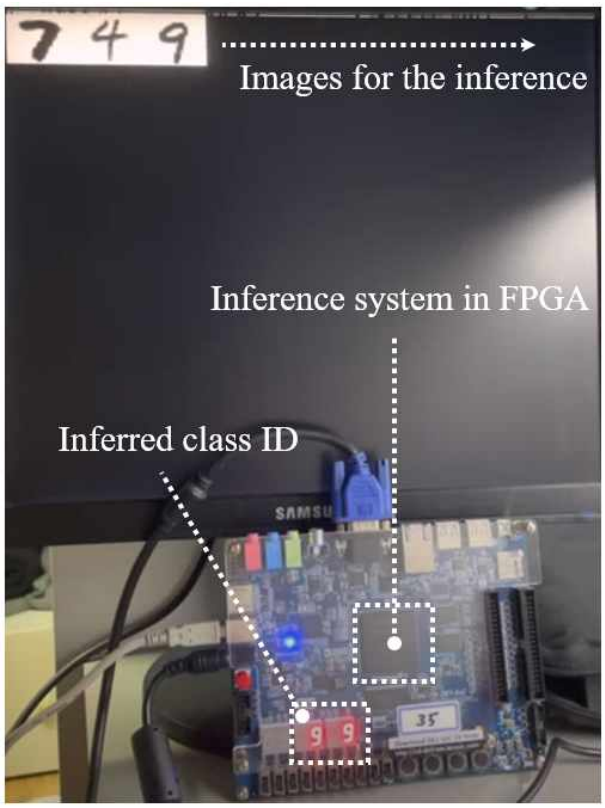


그림 5. 제안하는 시스템을 기반으로 하는 CapsNet 추론 시연 환경
Fig. 5. Demonstration environment for Capsnet inference based on the proposed system.

	Step 1	Step 2	Step 3 - 6 (Dynamic Routing)	Total
Execution Time (ms)	137	119	133	395

표 5. 제안하는 시스템의 수행 시간 분석
Table 5. Analysis of the execution time in the proposed system.

DSP 기준으로 3%이다. 해당 시스템에 사용된 FPGA는 저가형으로 가용 자원이 매우 제한적임에도 불구하고, 전체 시스템이 성공적으로 Fitting 되었다는 점에 주목할 필요가 있다.

제안하는 시스템을 이용한 추론을 시연하기 위한 환경을 그림 5에서 보이고 있다. 참고문헌 [2]에 제시된 CapsNet 모델을 기반으로 MNIST 데이터셋에 대한 이미지 분류 작업을 수행하도록 하였다. 먼저 외부 SDRAM에 추론 대상 이미지와 파라미터들을 로딩하고, 제안하는 시스템을 이용해 2절에서 설명한 CapsNet 추론 알고리즘을 통해 DigitCaps를 구하고, 가장 큰 크기를 갖는 DigitCaps값에 대응되는 Label을 Parallel IO를 이용해 외부로 출력한다. 제안하는 시스템은 해당 데이터셋의 이미지 분류 작업에 대해 98.17%의 정확도를 달성한다. 표 5는 제안하는 시스템의 수행 시간을 단계별로 분석한 것이다. 표 3에 정리된 순수 소프트웨어 구현에 의한 수행 시간 결과와 비교하여, 전체 추론 시간 측면에서 1/193수준으로 단축된 결과를 보인다. 이는 2절에서 설명한 CapsNet 추론 알고리즘의 수행 시간에 상당 부분을 차지하는 Step 1 - 2 를 전용 Component에 의해 고속으로 수행한 효과이다.

본 연구에서 제안한 CapsNet 추론 시스템의 FPGA 구현 결과를 기존의 결과들 참고문헌 [8], [13-14] 와 비교하여 표 6에 정리하였다. FPGA의 차이를 고려한 객관적인 로직 자원 사용량 비교를 위해 참고문헌 [12]에서 제시한 기준에 따라 해당 지표를 정규화 하였다. 제안하는 시스템은 기존 시스템 대비 자원 사용량이 적다. 이는 기존 시스템의 경우 추론 알고리즘의 전체 혹은 대부분을 전용의 Component로 구현해서 처리하는 반면, 제안하는 시스템의 경우 CapsNet 추론 알고리즘의 병목 부분으로 분석된 Step 1 - 2 만을 전용의 Component로 처리하며 나머지 부분은 범용 Microprocessor로 처리하기 때문이다. 그럼에도 불구하고, 제안하는 시스템의 추론 속도는 Peak Speed 관점에서 기존 시스템과 비교하여 그렇게 낮지는 않은데, 그 이유는 앞 절에서 제시한 동적 라우팅 반복 횟수 제한과 스쿼시 활성화 함수의 변형과 같은 최적화 방법에 따른 결과이다. 이에 따라, 제안하는 시스템은 자원 효율성 관점에서 기존 시스템에 비해 우월한데, 특히, GOP/s/DSP 형태로 정의된 DSP 효율성 지표에서, 기존의 가장 높은 결과와 비교했을 때 7배 정도 높다.

V. 결 론

본 연구는 한정된 자원을 갖는 FPGA를 위한 CapsNet 추론 시스템을 제안하였다. 고정 소수점 최적화 및 활성화 함수의 변형, 동적 라우팅 반복 횟수 최적화 등의 기법을 적용해 추론 알고리즘을 최적화하였다.

	ReConFig 2017[13]	ICCT 2019[14]	TCAS II 2020[8]	This work
Algorithm	CNN	CNN	CapsNet	CapsNet
Design Methodology	HLS	HLS	RTL	HLS
Platform	Virtex 7	Artix 7	Cyclone V	Cyclone V
Precision	16 bit fixed-point	16 bit fixed-point	4/8 bit fixed-point	8 bit fixed-point
(Total) DSP	2070	740	278	3
(Total) ALM	563.2 K [a)]	-	34.1 K	6.35 K
(Total) BRAM	72828 K-bit	-	2850 K-bit	163 K-bit
Peak Speed	611.5 GOP/s	22 GOP/s	96 GOP/s	7.06 GOP/s
DSP Efficiency	0.3 GOP/s/DSP	0.03 GOP/s/DSP	0.35 GOP/s/DSP	2.35 GOP/s/DSP

[a)] 1 LUT = 1.3 ALM 으로 정규화 함.

표 6. FPGA 기반 신경망 추론 시스템 구현 결과

Table 6. Implementation results of the neural-network inference systems based on FPGAs.

또한, Convolution과 Matrix Multiplication 연산 과정을 고속으로 수행하기 위한 전용의 Component를 HLS를 이용하여 구현하여, 시스템에 집적하였다. 제안하는 CapsNet 추론 시스템은 6348개의 ALM, 163K-bit BRAM, 3개의 DSP만으로 구현되었으며 2.35 GOP/s/DSP의 우수한 효율성을 달성한다.

REFERENCES

[1] G. E. Hinton, A. Krizhevsky, and S. D. Wang, “Transforming autoencoders,” in Proc. Int. Conf. Artificial Neural Networks, pp. 44 - 51, Espoo, Finland, Jun. 2011.

[2] S. Sabour, N. Frosst, and G. E. Hinton, “Dynamic routing between capsules,” in Proc. Conf. Neural Information Processing Systems, pp. 3859 - 3869, CA, USA, Oct. 2017.

[3] L. Bai, Y. Zhao, and X. Huang, “A CNN Accelerator on FPGA Using Depthwise Separable Convolution,” IEEE Trans. Circuits and Systems II, vol. 65, no. 10, pp. 1415 - 1419, Oct. 2020.

[4] H. Li, X. Fan, L. Jiao, W. Cao, X. Zhou, and L. Wang, “A High Performance FPGA-based Accelerator for Large-Scale Convolutional Neural Networks,” in Proc. Int. Conf. Field Programmable logic and Applications, pp. 1-9, Lausanne, Switzerland, Aug. 2016.

[5] Y. Ma, Y. Cao, S. Vrudhula, and J. Seo, “Optimizing the Convolution Operation to Accelerate Deep Neural Networks on FPGA,” IEEE Trans. Very Large Scale Integration Systems, vol. 26, no. 7, pp. 1354 - 1367, Jul. 2018.

[6] A. Marchisio, M. A. Hanif, and M. Shafique, “CapsAcc: An efficient hardware accelerator for CapsuleNets with data reuse,” in Proc. Design, Automation and Test in Europe Conf. Exhibit., pp. 964 - 967, Florence, Italy, Mar. 2019.

[7] A. Marchisio, V. Mrazek, M. A. Hanif, and M. Shafique, “FEECA: Design Space Exploration for Low-Latency and Energy-Efficient Capsule Network Accelerators,” IEEE Trans. Very Large Scale Integration Systems, vol. 29, no. 4, pp. 716 - 729, Apr. 2021.

[8] G. Park, D. Im, D. Han, and H. Yoo, “A 1.15 TOPS/W Energy-Efficient Capsule Network Accelerator for Real-Time 3D Point Cloud Segmentation in Mobile Environment,” IEEE Trans. Circuits and Systems II, vol. 67, no. 9, pp. 1594 - 1598, Sep. 2020.

[9] X. Zhang, S. L. Song, C. Xie, J. Wang, W. Zhang, and X. Fu, “Enabling highly efficient capsule networks processing through a PIM-based architecture design,” in Proc. IEEE Int. Symp. High Performance Computer Architecture, pp. 542 - 555, CA, USA, Feb. 2020.

[10] A. Marchisio, V. Mrazek, M. A. Hanif, and M. Shafique, “DESCNet: Developing efficient scratchpad memories for capsule network hardware,” IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, Oct. 2020.

[11] C. Seol and K. Cheun, “A low complexity Euclidean norm approximation,” IEEE Trans. Signal Processing, vol. 56, no. 4, pp. 1721 - 1726,

Apr. 2008.
[12] 1-CORE Technologies, “FPGA logic Cells Comparison”
[13] M. K. Hamdan and D. T. Rover, “VHDL generator for a high performance convolutional neural network FPGA-based accelerator”, in Proc. Int. Conf. ReConFigurable Computing and FPGAs, pp. 1 - 7, Dec. 2017.
[14] Q. Zhang, J. Cao, Y. Zhang, S. Zhang, Q. Zhang and D. Yu, “FPGA Implementation of Quantized Convolutional Neural Networks”, in Proc. IEEE Int. Conf. Communication Technology, pp. 16 - 19, Xi’an, China, Oct. 2019.

저 자 소 개

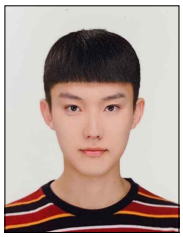
양 해 찬(학생회원)
2016년~현재 한국항공대학교 항공전자정보공학부 학사 과정.

<주관심분야 : 회로 및 시스템, VLSI 설계, ASIC 설계>

박 관 영(학생회원)
2016년~현재 한국항공대학교 항공전자정보공학부 학사 과정.

<주관심분야 : 회로 및 시스템, VLSI 설계, ASIC 설계>

김 태 환(평생회원)-교신저자
2005년 연세대학교 전기전자공학과 학사 졸업.
2007년 한국과학기술원 전기 및 전자공학과 석사 졸업.
2010년 한국과학기술원 전기 및 전자공학과 박사 졸업.
2011년~현재 한국항공대학교 항공전자정보공학부 교수.
<주관심분야 : 회로 및 시스템, VLSI 설계, ASIC 설계>



박 상 준(학생회원)
2016년~현재 한국항공대학교 항공전자정보공학부 학사 과정.

<주관심분야 : 회로 및 시스템, VLSI 설계, ASIC 설계>

사 재 현(학생회원)
2016년~현재 한국항공대학교 항공전자정보공학부 학사 과정.

<주관심분야 : 회로 및 시스템, VLSI 설계, ASIC 설계>