공학 학사 학위논문

BitBlade 구조에 AND-net을 활용한 Multi-Precision 뉴럴 네트워크 가속기 하드웨어 설계

2022년 12월

서울대학교 공과대학

전기정보공학부

전 혜 성

BitBlade 구조에 AND-net을 활용한 Multi-Precision 뉴럴 네트워크 가속기 하드웨어 설계

지도교수 김 재 준

이 논문을 공학 학사 학위논문으로 제출함

서울대학교 공과대학

전기정보공학부
전 혜 성

전혜성의 공학 학사 학위논문을 인준함

2022년 12월 21일

지도교수 김 재 준 (인)

초 록

본 연구는 2-bit multiplication을 단위로 하는 multi-precision 하드웨어 가속기 core인 BitBlade 아키텍처에 convolution decomposition의 아이디어를 확장하고 적용하여 1-bit multiplication을 단위로 하여 binary neural network 및 1-bit precision을 지원할 수 있는 multi-precision 하드웨어 가속기 core를 제안한다. Convolution decomposition은 본래 BNN의 in-memory computing을 위하여 XNOR-net의 NAND-net으로의 conversion을 위해 제시되었다. 이를 활용하여 1-bit precision을 지원하기 위하여 xnor gate와 and gate의 연산 결과를 muxing 하는 것이 아닌 and gate만 사용되는 bitbrick 구조를 채택할 수 있도록 하였다. 나아가 연산 특징을 고려하여 logic을 더욱 optimize하며 동일한 수준의 latency으로 정확한 계산을 하도록 2가지 dynamic term을 한 번에 계산하는 logic을 추가하였고, 그 결과 효과적인 resource reduction이 관찰되었다.
본 논문은 latency와 throughput, 그리고 resource 모두 overhead 없이 multi-precision을 지원하는 하드웨어 가속기 core 구현에 대한 내용을 담고 있다.

주요어 : DNN, 하드웨어 가속기, multi-precision, BitBlade 아키텍처, XNOR-net

목 차

표 목차

그림 목차

제 1 장 서 론

제 1 절 연구 배경

오늘날 Artificial Intelligence(AI)는 컴퓨터 비전과 음성 인식 및 번역 등 다양한 기능을 수행하고 있다. AI 모델의 정확도 및 실행 속도 수준을 크게 발전시킨 요소 중 하나는 Deep Neural Network(DNN)의 제안이다. DNN은 입력 정보에 가중치를 곱하여 출력 정보를 계산하고 이를 전달하는 perceptron을 단위로 하여 다중 perceptron layer 구조를 가지며 결과값의 loss를 통해 각각의 가중치가 학습되도록 구성되어 있다. 특히 2D Image를 활용하는 컴퓨터 비전 등의 분야에서는 Convolution 곱이 활용되는 Convolutional Neural Network(CNN)이 주로 활용되기도 한다.

더 크고 깊은 네트워크일수록 더 많은 implicit information이 활용되면서 더 정확한 모델을 만들 수 있게 되는 경향이 있는데, 이에 따라 모델의 크기가 커지면서 컴퓨터가 초당 수행해야 하는 연산량이 늘어나게 되었다. 큰 모델을 사용하면서도 컴퓨터가 적은 에너지를 소모하고 짧은 train time과 inference time이 가능하도록 효율적인 네트워크를 구조를 사용하고, 모델에 기여하지 않는 부분을 제거하는 pruning, 연산 단위를 더 작은 bit로 표현하는 quantization을 통해 모델을 경량화 하려는 시도가 이루어지고 있다. 이러한 SW적인 접근과 더불어 Central Processing Unit(CPU) 보다 병렬적인 연산이 가능한 기존의 Graphic Processing Unit(GPU)를 활용하거나 SW적인 접근에 보다 최적화된 하드웨어 가속기를 사용하는 HW적인 접근도 동시에 이루어지고 있다.

Quantization이 적용된 모델의 경우 CPU나 GPU는 연산 단위의 bit 수가 고정되어 있어 새롭게 특화된 하드웨어 가속기를 필요로 하는 경우가 많다. 이러한 하드웨어 가속기에 요구되는 기능 중 하나는 모델에 따라 전체적인 연산 단위 bit 수가 달라지거나 DNN의 각 layer마다 가능한 최소의 연산 단위의 bit 수가 달라지는 ‘multi-precision 네트워크’가 되는 경우를 모두 지원하도록 하는 것이다. Binary Neural Network(BNN)에 사용되는 최소의 1-bit에서 최대 8-bit 또는 그 이상의 수를 곱하고 더할 수 있는 되는 computing core를 필요로 하는 것이다.

Multi-Precision을 지원하는 core를 위해 2-bit 단위로 multiply-and-accumulate(MAC) 연산기 및 shifter가 있는 unit을 고정된 개수로 두고 input과 weight의 precision에 따라 unit들을 배분하는 BitFusion 구조가 제안된 바 있다. 그러나 이 구조에는 모든 unit 마다 shifter가 포함되므로 많은 resource를

사용하게 되고 이에 따른 power overhead가 존재한다. 이를 해결하기 위해 unit의 묶음에 한 개의 shifter만 있도록 하는 개선된 BitBlade 구조가 제안되었다. 하지만 두 구조 모두 1-bit의 binary precision 연산을 고려하지 않는데, 우선 연산 단위가 2-bit부터 지원되므로 1-bit의 경우 2-bit처럼 취급하여 연산해여야 해서 BNN을 적용한다면 실행 속도가 2배 느려지며, binary precision 곱에 필요한 xnor 연산을 지원하지 않는다.

BNN에서의 binary precision에서는 (0, 1)이 각각 (-1, +1)을 의미하기 때문에 곱셈이 and가 아닌 xnor 연산으로 표현된다. 이러한 차이점으로 인하여 binary precision만 고려한 가속기나 2-bit 이상의 precision 부터 연산 가능한 multi-precision 가속기가 제안된다. Binary precision을 고려하는 경우 bit-serial한 MAC unit을 사용하며, and 연산기와 xnor 연산기를 모두 unit 내에 두고 muxing하는 방법이 제안되었다. 그러나 이러한 경우 unit마다 shifter가 포함되며 and gate와 xnor gate를 동시에 사용하므로 resource overhead가 크다는 단점이 있다.

본 연구에서는 binary precision에서의 xnor 곱 연산을 and 연산들로 decompose 하여 xnor gate 없이 and gate만으로 MAC unit을 구성하는 것을 제안하는 AND-net 구조를 BitBlade 구조에 적용하여 shifter과 xnor gate로 인한 overhead가 없는 bit-parallel한 computing core logic인 BitBAND 구조를 제안하고자 한다. 이를 통해 binary precision부터 연산 가능한 multi-precision 가속기 하드웨어를 구현하고자 한다.

제 2 절 연구 내용

본 연구에서는 in-memory computation(CIM)을 위해 xnor gate가 아닌 nand gate만을 활용한 binary precision MAC을 위해 제안된 NAND-net 구조의 multiplication decomposition 원리를 이용해 BitBlade 구조의 unit인 bitbrick을 기존의 2-bit 단위가 아닌 1-bit 단위로 input과 weight를 받으며 최대 4-bit * 4-bit 연산이 가능하도록 설계하였다. 2-bit precision 이상부터 sign bit가 존재함을 고려하여 sign bit가 input으로 있는 bitbrick만이 아닌 sign bit를 input으로 갖지 않는 경량화 된 bitbrick 두 가지를 교대로 배치하여 resource overhead를 줄였다. 이러한 최적화가 모두 적용되었을 때에도 xnor gate를 muxing하지 않는 BitBAND 구조가 가장 적은 resource 및 power overhead를 가진다는 것을 Xilinx 시뮬레이션으로 확인하였다.

제 2 장 본 론

제 1 절 선행 연구

뉴럴 네트워크는 주어진 input에 대해 weight를 곱하고 필요에 따라 비선형 함수를 통과한 후 이를 output으로 하는 perceptron을 기본 구조로 한다. [1] 이러한 perceptron들이 있는 layer를 여러 층으로 배치하여 한 layer의 output이 다음 layer의 output으로 전달되는 multi-layer perceptron이 DNN의 기본 구조이다. Train 과정에서 결과를 참값과 비교하여 loss를 계산하고, 이에 대해 gradient descent 등의 analytic computation을 하여 weight의 값을 조정하는 iteration을 반복하며, 학습 후에는 고정된 weight 값으로 새로운 input에 대해 output을 도출하는 inference를 하게 된다. 크고 깊은 layer를 가져 implicit한 information을 더욱 잘 고려하는 DNN 모델이 발전하면서 더욱 빠르고 정확한 컴퓨터 비전 및 자연어 처리 어플리케이션의 개발이 가능하다. Virtual Reality(VR)과 augmented reality(AR)을 지원하거나 음성이나 주변 환경을 인식하는 웨어러블 디바이스가 이러한 기술을 바탕으로 한다. [2]

컴퓨터 비전의 경우 물체 인식(object recognition) 및 물체 분류(object classification)의 주요 task가 있으며 이들 모델에서는 image의 부분적인 특징을 확인할 수 있는 convolution 연산이 사용된다. [3, 4] 따라서 이들을 CNN으로 지칭하기도 한다. 여러 개의 layer로 이루어진 block을 반복적으로 사용하고, block의 input을 output과 augment 하거나 기존의 pre-trained NN model을 활용하는 등의 시도를 통해 정확도 지표인 mean average precision(mAP)가 80% 이상인 모델이 등장하고 지금까지도 발전 중에 있다. [5] 최근에는 CNN이 아닌 자연어 처리에 사용되던 transformer 모델을 기반으로 한 vision transformer(ViT) 모델이 더 높은 성능을 보이고 있다. [6]

수억 개의 weight parameter를 갖는 ViT 모델들은 물론, CNN 모델들도 많은 matrix multiplication을 필요로 하며 Giga scale의 초당 실수 연산(floating-point operations per second, FLOPs)이 필요하다. Parameter 수와 연산량의 증가로 인한 memory bottleneck과 training 및 inference 시간을 줄이기 위하여 pruning과 quantization의 모델 경량화 기법이 제안되었다. Pruning은 training 과정에서 weight 값이 일정 수준 이하인 connection을 제거하여 0의 값으로 고정하고, 다시 training을 하며 이와 같은 과정을 반복해 가능한 최소의 connection을 갖도록 하는 방법이다. [7] Quantization은 training이 완료된 후나 training 중 parameter들의 표현에 필요한 bit 수를 줄이는 것으로 전자는 post-training quantization(PTQ), 후자는 quantization-aware training(QAT)에 해당된다. PTQ와

QAT는 각각 그 목적이 명확한데, 추가적인 training time 없이 quantization을 적용하고자 한다면 PTQ를, quantization을 고려한 training time overhead를 감수하고 보다 높은 정확도를 추구한다면 QAT를 사용하는 것이 적절하다. Layer 마다 input 혹은 weight의 bit 수를 학습 parameter로 사용하는 QAT 방법들은 물론, PTQ 방법도 각 layer에 적절한 quantization 방법을 사용하여 layer마다 다른 quantization이 적용될 수 있다. [8, 9]

모델 경량화와 더불어 모델에 맞게 설계된 하드웨어 가속기 사용의 중요성이 증가하고 있다. 위에서 서술한 DNN 모델 관련 논문들도 모두 CPU보다 GPU가 matrix multiplication 등에 해당하는 많은 양의 연산을 처리하도록 하고 있다. [5-7] 나아가, quantization을 고려하여 기존의 32-bit 실수가 아닌 특정 bit-width의 수를 곱하고 더하는 연산들을 수행하기 위해서는 GPU보다 효율적인 하드웨어 가속기를 필요로 한다. 앞서 서술한 바와 같이 모델 마다 quantization 방법이 다르고 한 모델 내에서도 각 layer나 channel 마다 bit-width가 다름에도 이를 처리할 수 있으려면 multi-precision 하드웨어 가속기를 필요로 하게 된다.

Input과 weight 모두 quantization 하는 경우 발생하는 multi-precision 경우들을 고려하여 bit-parallel하게 연산하는 코어인 BitFusion 아키텍처가 state-of-the-art로 제시된 바 있다. [10] 이를 바탕으로 새롭게 제안된 BitBlade 아키텍처는 BitFusion의 MAC unit인 bitbrick마다 높은 area 및 resource overhead를 야기하는 shifter logic이 있어 생기는 overhead를 제거하기 위하여 bit-parallel한 연산들 중 같은 값만큼 shift 되는 연산 값들을 모두 더한 후 이를 1번만 shift 하도록 하여 41%의 area reduction 및 36% 이상의 energy reduction을 달성하였다. [11] Figure 1.에서 두 아키텍처 모두 shared input으로 같은 형태의 dataflow를 사용하였으며, input과 weight의 multiplication을 수행하는 unit인 bitbrick이 2-bit multiplication을 하며 bitbrick으로 processing element(PE)를 구성하고, PE로 PE array가 구성하며, 서로 다른 weight 마다 이러한 PE array를 할당하였음을 확인할 수 있다. BitShift에서 BitBlade로의 conversion을 통해 input 과 weight의 다양한 precision 경우들을 고려하며 bit-serial 연산기나 BitShift 아키텍처보다 효율적인 area와 power를 가질 수 있는 것이다.

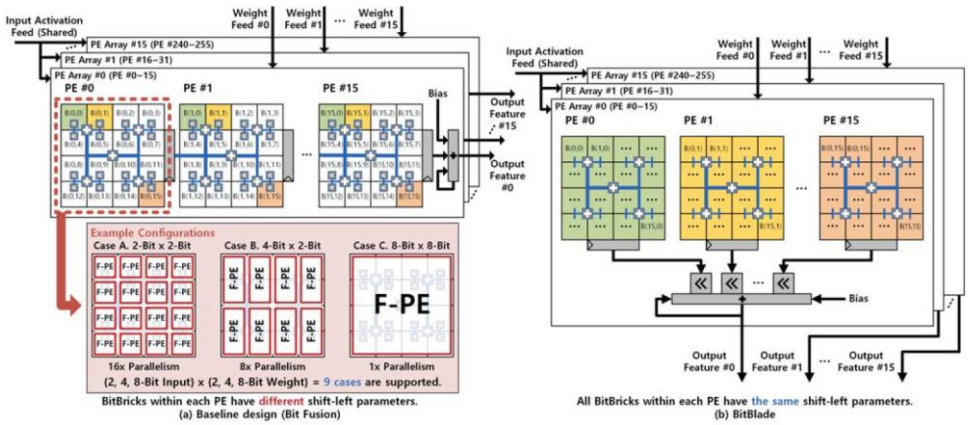

Figure 1. 같은 dataflow 상에서의 (a) BitFusion과 (b) BitBlade의 아키텍처 비교.
(a)의 하단과 같이 input과 weight의 bit-width이 달라짐에 따라 bit-parallel 연산 각각에 할당되는 bitbrick이 달라지는 방식을 통해 input과 weight의 multi-precision을 지원한다. BitFusion의 경우 PE 내의 각 bitbrick마다 다른 위치의 multiplication이 일어날 수 있어(초록색과 노란색 등이 혼재) shifter logic을 포함하고 있는 반면, BitBlade의 경우 bitbrick 내에는 shifter logic이 없으며 같은 위치의 multiplication 결과를 더한 후 PE 당 1개 있는 shifter logic으로 shift를 한다.

한편, 주어진 값에서 부호만을 취하여 {-1, +1}으로 quantization을 하고 이를 표현하기 위해 1-bit을 사용하는 BNN 모델이 제안된 바 있다. AlexNet 등을 대상으로 실험 결과 대비 정확도가 16% 이내로 감소하나 memory 사용이 32배 절약할 수 있고 연산 속도는 58배 빨라지는 결과가 도출되었다. [12] 1-bit quantization의 경우 (-1, +1)의 값이 연산 logic에서 각각 (0, 1)으로 mapping 되며 $\{-1,+1\}\times\{-1,+1\} \leftrightarrow (0,1)\times(0,1)$ 의 네 가지 곱셈 결과가 $\{+1,-1,-1,+1\} \leftrightarrow (1,0,0,1)$으로 mapping 되므로 1-bit precision에서는 곱셈이 2:1 xnor gate와 동일한 기능을 하게 된다.

회로의 집적도가 물리적인 한계에 거의 다다른 오늘날, 프로세서의 성능으로 인한 computing bound보다 프로세서에 충분한 memory bandwidth로 상응하지 못하는 memory bound로 인한 한계가 대두되며 memory 내에서도 computing을 하도록 하는 CIM 기술이 주목받고 있다. [13] BNN을 6T cell 기반의 SRAM이나 high density를 갖는 DRAM의 메모리 상에서 CIM으로 train 및 inference하기 위해서는 8T cell의 xnor로의 conversion이 필요하거나 single cycle 내에 xnor gate 연산을 해결할 수 없다. 이를 해결하기 위해 수식 (1)과 같은 convolution decomposition으로 BNN을 XNOR-net이 아닌 NAND-net으로 conversion 하는 방법이 제안되었다. [14]

$$I_o * W_0 = (I_b + 2I_f) * (W_b + 2W_f)$$
$$= I_b * W_b + 2(I_b * W_f + I_f * W_b) + 4I_f * W_f \quad \dots (1)$$

Figure 2.에서와 같이 모든 element가 -1인 base term에 +1의 위치에만 +1의 element를 갖는 filtered term을 2배 더하면 기존과 같은 term이 되며 이렇게 decomposed 된 convolution term은 xnor가 아닌 and gate를 이용해 계산할 수 있게 된다. 특히 inference 시 input feature는 미리 그 값을 알 수 없는 dynamic이지만 kernel weight는 constant로 생각되므로 BF-BK term과 BF-FK의 term은 pre-compute 하여 bias 값과 합쳐질 수 있다.

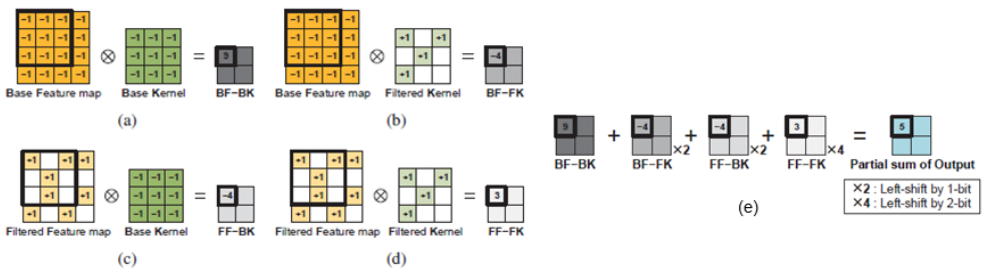

Figure 2. Convolution Decomposition에 의한 (a) BF-BK (b) BF-FK (c) FF-BK (d) FF-FK term (e) 각 term의 계수인 (1, 2, 2, 4)를 곱하고 더하여 도출한 partial sum.
Kernel size에 따라 (a)의 값이 정해지며 constant인 kernel weight의 분포에 따라 (b)의 값이 정해지므로 두 값은 static term으로 생각할 수 있다. Input feature의 분포에 따라 값이 달라지는 (c)와 (d)는 dynamic term으로 생각할 수 있다.

CIM 뿐만 아니라 하드웨어 가속기 디자인에 이러한 아이디어를 이용하면 1-bit precision과 그 이상의 precision을 수행하기 위하여 multiplication을 수행하는 MAC unit 또는 PE의 sub-unit에서 xnor gate와 and gate를 함께 두고 1-bit precision이면 xnor gate의 결과를 이용하도록 muxing하는 기존의 방법보다 발전할 수 있을 것으로 예상된다. [15] 즉, and gate만으로도 multiplication이 가능하므로 area 및 resource overhead를 줄여 power overhead를 아낄 수 있을 것으로 기대한다. 또한, 1-bit precision을 지원하지 않는 BitBlade보다 효율적인 BNN 수행이 가능할 것이다.

제 2 절 연구 방법

본 연구에 앞서 기존의 BitBlade 아키텍처를 이용하여 YOLOv3-tiny 모델 inference용 Figure 3.와 같은 overall 아키텍처를 갖는 하드웨어 가속기를 Zync UltraScale+ MPSoC ZCU102 보드에 구현하였다. 2-bit, 4-bit, 8-bit의 input과 weight bit-width에 대해 all-layer test로 정확성을 확인하였고, PTQ를 적용하여 INT8 precision으로 quantized 된 mAP 79.35%의 YOLO 모델을 보드 상에서 약 20FPS의 속도로 실행할 수 있었다. 본 연구에서도 이 프로젝트에서 사용하였던 dataflow를 가정하고, bias와 weight 용 BRAM buffer과 이의 logic, result saver logic이 1-bit, 2-bit, 4-bit를 지원하도록 새롭게 설계된 core에 맞게 re-design 되어도 더 작은 bit-width의 data를 다루므로 resource 측면의 overhead가 크지 않음을 가정하고 BitBlade와 이의 변형된 core의 logic만을 비교할 것이다.

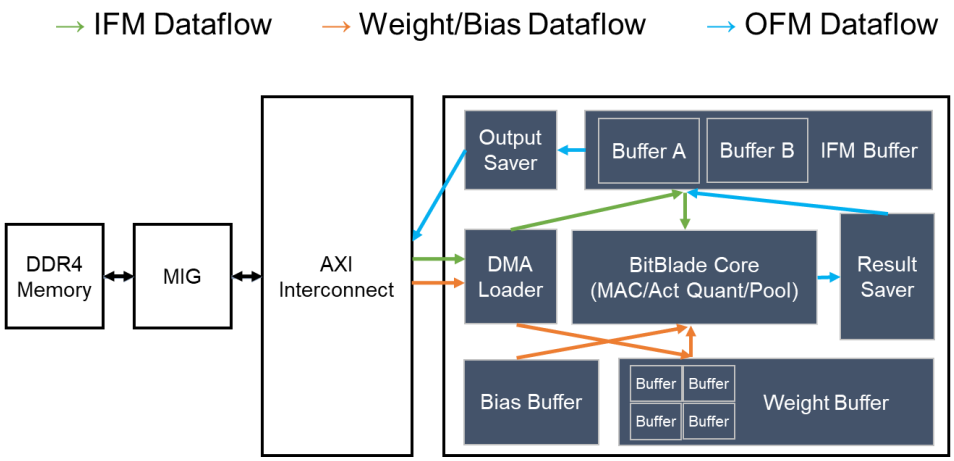

Figure 3. YOLOv3-tiny inference 하드웨어 가속기의 overall architecture. BitBlade core를 이용해 구현하였으며, weight, bias, initial image data가 DDR4 memory로부터 DMA로 각각 BRAM인 weight buffer, bias buffer, IFM(Input Feature map) buffer에 저장된다. 각 buffer logic이 core에 적절한 data를 제공하며, result saver는 연산 결과를 OFM(Output Feature map) buffer에 저장한다. Box recognition과 classification에 필요한 특정 layer의 output은 output saver를 통해 다시 memory로 전달된다.

BitBlade 아키텍처의 bitbrick unit의 logic을 2-bit multiplication에서 1-bit multiplication으로 수행하도록 변경할 때, XNOR-net 구현을 위해 and gate와 xnor gate를 muxing하는 구조를 BitXNOR로 지칭하고 AND-net 구현을 위해 and gate만을 갖는 구조를 BitBAND로 지칭할 것이다. BitBAND의 경우 NAND-net의 convolution decomposition을 이용하므로 input 또는 weight가 1-bit precision 일 때 앞서 서술한 바와 같이 Filtered Feature와 Base Kernel 또는 Filtered Kernel의 multiplication인 dynamic term 만을 계산하고 bias에 static term이 포함되었음을 가정한다.

Figure 4.와 같이 BitBlade의 bitbrick에서 2-bit multiplication을 할 때 2-bit 보다 높은 precision의 연산에서 input 또는 weight의 MSB 2-bit를 포함하는 경우 1-bit extension을 하여 연산을 한다. 즉 sign bit s_x, s_y를 포함하여 실질적으로 3-bit과 3-bit multiplication을 하는 logic을 갖는다. 이와 비교하여 Figure 5.와 같이 BitXNOR과 BitBAND는 공통적으로 1-bit multiplication을 담당하는 bitbrick을 새로운 unit으로 가지는데, 1-bit precision 보다 높은 precision의 연산에서 input의 MSB 1-bit를 연산하는 경우를 위해 1-bit extension을 하여 연산을 한다. 즉, signed bit을 포함하여 실질적으로 2-bit 과 1-bit 간의 multiplication을 하는 logic을 갖는다. 1-bit 단위 bitbrick에서는 이처럼 input 만의 sign bit를 포함하는 이유는 multiplication overflow가 일어나지 않게 할 때 input의 MSB가 곱해진 결과 bit만 signed extension을 해서 더하면 되기 때문이다.

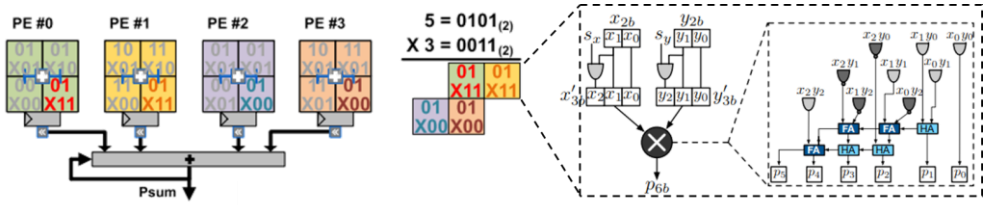

Figure 4. BitBlade에서 4-bit precision의 input과 weight를 계산할 때 2-bit multiplication이 각각 bitbrick에 할당되고, 같은 위치의(같은 shift 값을 갖는) bitbrick이 4개씩 모여 하나의 PE를 이루고 있다. 각 bitbrick은 input 또는 weight의 MSB를 포함하는 경우를 위하여 sign extension bit s_x, s_y가 포함된 3-bit multiplication을 하는 logic이다.

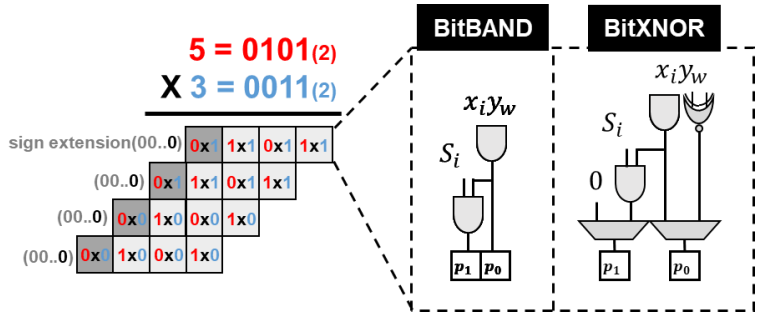

Figure 5. BitBAND와 BitXNOR에서 1-bit multiplication이 각각 bitbrick에 할당된다. 각 bitbrick은 input의 MSB를 포함하는 경우를 위하여 sign extension bit s_i를 포함하여 2-bit과 1-bit을 multiply 하는 logic이다. BitXNOR의 경우 1-bit precision인지에 따라 xnor 결과와 and 결과가 mux 된다.

세 아키텍처 모두 같은 core throughput을 갖도록 PE array와 각 PE array 내의 PE 수를 16개로 동일하게 설정하였고, input과 weight가 최대 8-bit precision을 가지며 8개의 bitbrick이 하나의 PE를 이루는 BitBlade에서 하나의 PE array 당 매 cycle마다 32개의 (4-bit, 4-bit) parallel multiplication이 가능하므로 input과 weight가 최대 4-bit precision을 가지는 BitBAND나 BitXNOR에서는 32개의 bitbrick이 하나의 PE를 이루도록 하였다. 이처럼 1-bit bitbrick 디자인이 본 연구의 첫 번째 approach로 볼 수 있다.

두 번째 approach로서 실제로 sign extension이 필요한 경우는 input이 2-bit 이상의 precision에 해당하는 연산의 MSB가 연산 되는 경우임을 이용하여 한 PE array 내의 16개의 PE 중 8개의 PE에 해당하는 bitbrick은 sign bit을 포함하고 나머지 8개의 PE에 해당하는 bitbrick은 sign bit 없이 true 1-bit multiplication인 and gate로만 이루어진 logic이 되도록 하였다. 이는 input과 weight가 모두 2-bit precision일 때 sign extension이 필요한 연산을 하는 PE가 8개가 되기 때문이다. 따라서 실제 bitbrick은 Figure 6.와 같다. 또한, 각 경우의 PE에 대해 적절한 bit-width의 bitbrick 결과들을 더하는 adder를 갖도록 하였다. 이처럼 2-bit에서 1-bit bitbrick으로의 conversion 과정에서 bitbrick 수는 4배 증가하지만 6-input logic에서 3-input logic과 2-input logic의 조합으로 unit의 area 및 resource를 4배 이상 절감하여 power reduction을 기대할 수 있다.

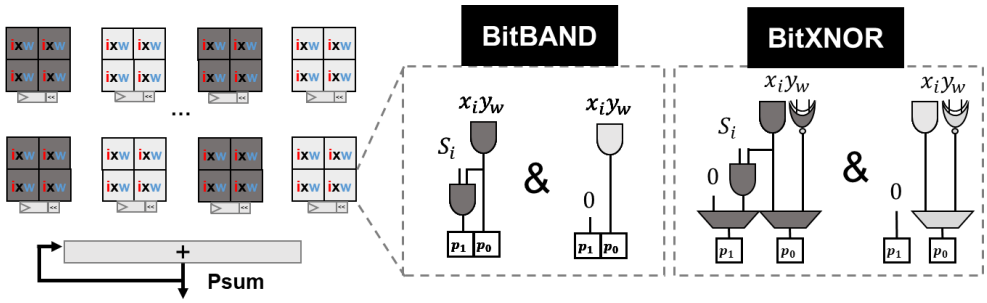

Figure 6. BitBAND와 BitXNOR의 2가지 bitbrick logic. Sign extension이 필요한 2-bit 이상의 precision에서 사용되는 PE에 해당하는 bitbrick만 sign bit을 input으로 한다.

마지막으로 BitBAND에는 BitBlade나 BitXNOR에는 없는, dynamic term 2가지를 한 번에 연산하기 위한 logic이 추가적으로 필요하다. Input과 weight의 precision에 따라 하나의 multiplication 당 최대 2개의 dynamic term이 나올 수 있는데, 2개의 dynamic term을 2 cycle 동안 계산한다면 BitBlade나 BitXNOR 보다 2배의 latency를 소요하는 것이므로 2개의 dynamic term을 한 cycle 내에 연산하기 위해 각 PE마다 할당되는 adder logic을 이에 맞게 디자인하여야 한다.

Figure 7.과 같이 NAND-net에서 제시한 convolution decomposition에 대한 확장된 수식을 생각해볼 수 있다. 우선 input과 weight가 모두 1-bit 인 경우 2개의 dynamic term이 발생하며 각각 1-bit, 2-bit의 left shift가 필요하다. Weight만 1-bit인 경우에도 2개의 dynamic term이 발생하나 각각 0-bit, 1-bit의 shift가 필요하다. Input만 1-bit인 경우에는 Base feature와 weight의 곱이 static term이므로 1개의 dynamic term이 발생하며 1-bit shift가 필요하다. 마지막으로 input과 weight가 모두 2-bit 이상인 경우 BitBlade와 같이 단일한 dynamic term을 계산하게 된다. 이때, dynamic term에 대한 shift는 앞에서 서술한 같은 위치의 bitbrick 연산 결과를 모두 더하여 shift를 해주는 과정 이전에 발생하는 shift로서 이에 대한 logic도 기존의 shifter와 마찬가지로 각 PE마다 할당되는 adder logic에 1개씩 존재하게 된다.

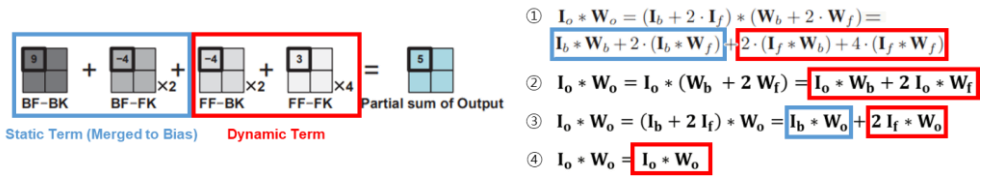

Figure 7. Binary precision 이상의 경우를 포함하는 확장된 convolution decomposition 수식. Bias로 merge 될 수 있는 static term과 core logic에서 계산하여야 하는 dynamic term이 각각 푸른색과 빨간색으로 강조되었다.
1-bit precision이 아닌 input 또는 weight는 decomposition이 일어나지 않고 MSB의 경우 sign bit를 갖게 된다. Decomposition이 일어나지 않더라도 input 또는 weight 중 1-bit precision이 있다면 dynamic term에 해당되어 shift가 필요하다.

Dynamic term이 2개 이상 존재하는 경우는 weight가 1-bit일 때이며, (filtered) input feature와 filtered kernel과의 multiplication에 (filtered) input feature와 base kernel과의 multiplication을 더해주어야 한다. 이때, base kernel은 모든 element가 -1인 window로 고정되어 있고 (filtered) input은 core로 들어오는 값이므로 보다 dynamic한 Feature-FK term은 bitbrick에 할당하여 계산한다. 그리고 각 PE array마다 bitbrick의 계산 결과인 Feature-FK term을 모두 더하는 adder 뿐만 아니라 input을 모두 더하고 이에 -1을 곱해주는 adder를 두어 Feature-BK term을 더해주도록 한다.

기존의 프로젝트에서 사용하였던 BitBlade core의 Verilog code를 참고하여 위의 3가지 approach를 수행하는 Verilog code를 작성하고, 이를 simulation으로 계산 결과를 검증하는 code를 작성한 후, Xilinx Vivado 프로그램 상에서 core logic 전체를 synthesis 하여 approach에 따른 LUT 및 FF(Flip-Flop) resource utilization을 비교하였다. Resource utilization은 area 및 power consumption에 직결되는 요소이므로 타당한 비교가 될 것이며, synthesis 결과 중 power report를 통해 power estimation 및 logic과 i/o 등에 의한 power breakdown을 확인하였다. Throughput이 2배가 되도록 bitbrick 수를 2배씩 scale-up 하였을 때의 결과도 함께 확인하였다.

제 3 절 연구 결과

BitBlade, BitXNOR, 그리고 BitBAND core에 대한 Verilog code는 다음 GitHub repository에 게시하였다. (https://github.com/hjeon2k/2022_GradPrj) Core logic 중 주로 bitbrick이 구현된 adder 모듈, bitbrick의 연산 결과를 합하는 PE의 일부인 sip_dot과 pe 모듈에 세 아키텍처 간 차이가 나타난다.

Table 1.은 세 아키텍처의 synthesis 결과 LUT와 FF의 수를 나타낸다. Throughput은 각각의 아키텍처에서 PE 당 bitbrick의 수를 2배로 하여 parallel multiplication 수를 2배로 늘린 경우가 2x에 해당한다. 단, 연구 방법에서 서술한 바와 같이 세 아키텍처 모두 1x와 2x끼리의 throughput은 같다. Optimization A와 B는 앞서 서술한 바와 같이 각각 BitXNOR과 BitBAND에 2-bit bitbrick에서 1-bit bitbrick으로의 conversion과 절반의 PE에 해당하는 bitbrick만 sign bit을 포함하도록 optimize 한 것을 의미하며, Optimization C는 BitBAND의 경우만 필요한 2가지 dynamic term의 연산을 위한 logic을 추가한 것을 의미한다.

Table 1. BitBlade, BitXNOR, BitBAND의 synthesis 결과 LUT 및 FF utilization 결과. Power consumption 비교 대상인 최종 아키텍처에 해당하는 값들을 회색으로 강조하였다. BitBlade의 경우 throughput 1x는 8 bitbricks/PE, 2x는 16 bitbricks/PE를 의미하며, BitXNOR과 BitBAND는 각각 32 bitbricks/PE, 64 bitbricks/PE를 의미한다. Optimization A를 적용할 시 sign extension bit을 고려한 bitbrick의 logic과 이로 인한 보다 큰 bit-width의 adder tree로 인한 resource overhead가 발생하므로 optimization B로 이를 완화하여야 한다. BitBAND의 dynamic term을 고려한 optimization C를 적용할 시 resource가 증가하지만 BitXNOR보다 적은 LUT를 사용하고 있다.

Architectures	Throughput	Optimization	LUTs	FFs
BitBlade	1x	**None**	38716	3005
	2x	**None**	67844	3277
BitXNOR	1x	A	67080	2349
		B	29804	1453
	2x	**B**	51912	1597
BitBAND	1x	A	44780	2349
		B	22014	1597
		C	27012	1597
	2x	**C**	51628	1741

모든 Bitbrick에 sign bit를 input으로 사용하는 경우와 그렇지 않은 경우 모두 bitbrick 당 1개의 LUT를 사용하는 반면 BitBlade의 bitbrick이 5개의 LUT를 사용하며, bitbrick의 수는 BitXNOR과 BitBAND가 4배 많으므로 대략적인 resource reduction이 예상되었으나 실제로 1.73배로 증가하게 된다. 이는 우선 bitbrick 하나가 아닌 전체적인 logic을 synthesis할 때 LUT optimization이 되었기 때문일 것이며, 이와 더불어 PE 마다 존재하는 adder tree의 bit-width는 감소하나 PE 당 bitbrick 수가 4배로 늘어나며 그만큼 많은 element를 더하여야 하는 큰 tree가 되었기 때문일 것으로 예상되었다.

Optimization B를 적용하였을 때 1x throughput scale에서 BitBlade 아키텍처 대비 BitXNOR은 24%, BitBAND는 44%의 LUT reduction 효과가 있었다. FF 수 또한 47%~52% 감소하였다. Adder tree에서 더하는 element의 bit-width가 6-bit에서 2-bit로 감소하였고, sign bit가 필요하지 않은 bitbrick이 곧 2-input and gate로 대체될 수 있었기 때문일 것이다.

BitBAND에 optimization C를 적용하였을 때 1x throughput scale에서 이전의 경우 대비 23%의 LUT utilization이 증가하였다. 각 PE마다 1개씩 있던 adder를 2개로 만들어 2개의 dynamic term을 계산할 수 있도록 추가적인 logic을 사용하였기 때문일 것이다. 이러한 BitBAND의 최종적인 아키텍처는 BitBlade 대비 30%의 resource reduction과 동시에 1-bit precision 연산을 2배의 throughput으로 수행하며 그 이상의 precision에 대해서는 동일한 throughput을 가지게 한다. 또한, 이는 BitXNOR 대비 9%의 LUT reduction 효과가 있다. Figure 8.은 optimization에 따른 1x throughput scale BitBAND 아키텍처의 LUT utilization 변화 및 1x와 2x throughput scale에서의 세 아키텍처의 최종적인 LUT 및 FF utilization을 나타낸다. Scale-up 할 때의 LUT 증가 비율이 높은 것은 adder tree의 element가 증가함에 따라 logic의 복잡성이 증가하였기 때문일 것이다.

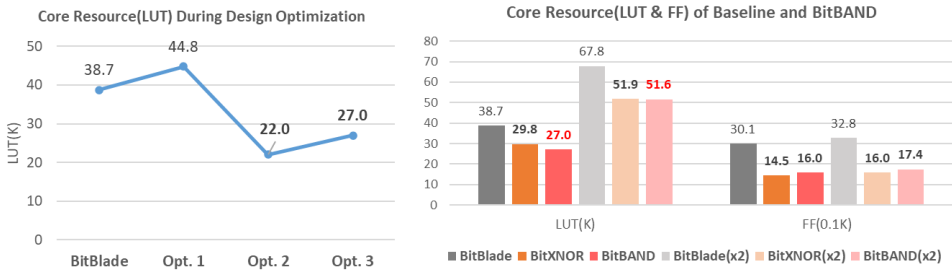

Figure 8. BitBlade에서 BitBAND으로의 optimization 적용에 따른 resource(LUT) utilization 변화 및 BitBlade, BitXNOR, BitBAND 아키텍처의 최종적인 resource utilization 결과. BitXNOR과 BitBAND는 scale-up 시 resource 증가율이 높다. BitBAND의 resource utilization이 가장 낮으며 이로 인한 power reduction을 기대할 수 있다.

세 아키텍처의 power consumption을 비교하기 위하여 1x throughput synthesis 결과 중 power report에서 power estimation 및 power breakdown을 확인한 결과는 Table 2.와 같다(단위: W). Total power가 BitBlade보다 21% 감소하였으나 BitXNOR에 비하여 약 1%의 overhead가 관찰되었다. 이는 Input과 output port를 buffer logic이나 result saver logic과 assign 해주지 않아 관련 confidence level이 낮아 발생하는 minor overhead 이거나, 앞서 분석한 resource utilization에서 확인하였듯 10%에 해당하는 144개의 FF를 더 많이 사용하고 있기 때문일 수 있다. Logic power는 BitXNOR 대비 5.4% 이상 감소하였으며, BitBlade 대비 58% 이상 감소하였다.

Table 2. 1x throughput에서의 BitBlade, BitXNOR, BitBAND의 synthetic power estimation breakdown(단위: W). Device 전반적인 static power과 dynamic power에 해당하는 i/o & clock, signal, 그리고 logic에 의한 power으로 breakdown 한 것이다. BitBAND의 logic power가 가장 적으나 i/o & clock으로 인한 overhead가 존재한다.

Architecture	Static	I/O & CLK	Signals	Logic	total
BitBlade	0.628	0.772	0.240	0.351	1.991
BitXNOR	0.625	0.542	0.153	0.234	1.554
BitBAND	0.626	0.571	0.156	0.222	1.575

최종적으로 세 아키텍처의 synthetic power estimation 결과와 LUT utilization과 static power를 제외한 dynamic power를 종합하여 비교한 결과는 figure 9.와 같다. BitBAND는 BitXNOR과 비교하였을 때 1%의 dynamic overhead 내에서 9%의 resource reduction을 가지며, logic의 resource 및 logic power reduction이 가능하다.

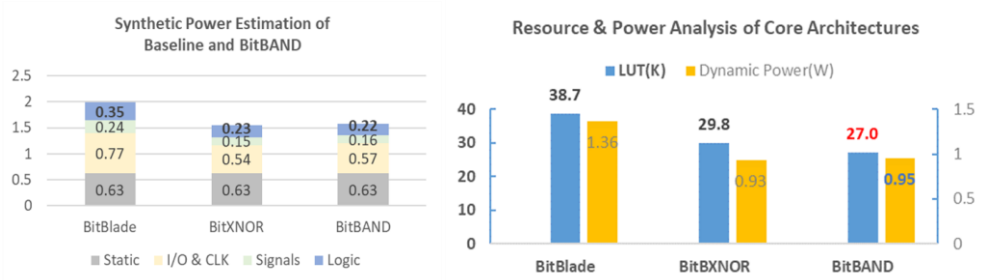

Figure 9. BitBlade, BitXNOR, BitBAND의 synthetic power estimation의 breakdown과 LUT utilization 및 Dynamic power의 비교. BitBAND는 BitXNOR 대비 1%의 power overhead 내에서 9%의 resource reduction을 가지며 이에 따른 area reduction도 함께 기대된다.

제 3 장 결 론

DNN 모델이 높은 성능을 달성하기 위해 parameter 수와 연산량이 증가하면서 이와 동시에 training 및 inference time을 줄이기 위해 정확도 감소가 적게 일어나도록 하면서 모델을 경량화 하는 quantization 등의 기법이 제안되었다. 그 결과 layer-wise 또는 channel-wise한 bit-width를 갖게 되고 multi-precision model들을 지원할 수 있는 optimized 하드웨어 가속기를 필요로 하고 있으며 BitFusion, BitBlade 등의 아키텍처가 제시된 바 있다. 한편, BNN에서의 binary precision은 다른 precision과 number representation 방법이 다르기 때문에 1-bit multiplication까지 지원하며 area 및 power overhead를 줄이기 위해 optimized 된 multi-precision 하드웨어 가속기를 연구하고자 하였다.

Bit-parallel한 core의 특징을 더욱 활용하여 shifter로 인한 resource overhead를 줄인 BitBlade 아키텍처를 바탕으로, BNN의 CIM 분야에서 제안된 convolution decomposition에 의한 XNOR-net에서 AND-net으로의 conversion을 이에 적용한 core 아키텍처인 BitBAND를 제안하였다. 연산에서 sign bit의 특성을 이용해 효율적인 1-bit bitbrick 구조를 디자인하였고, 이것이 적용된 BitXNOR에서는 xnor과 and가 muxing 되도록 하였고 BitBAND에서는 2개의 dynamic term을 같은 cycle 내에 연산할 수 있도록 PE마다 추가적인 adder를 두었다. 그 결과 BitBAND는 BitXNOR 대비 1%의 dynamic power overhead를 가지며 9%의 resource reduction을 달성하며 BitBlade 대비 1-bit precision 연산을 2배의 throughput으로 수행하며 그 이상의 연산은 동일한 throughput을 유지하도록 할 수 있었다.

본 연구에서는 Synopsys와 같은 보다 optimized 된 tool이 아닌 Vivado synthesis report의 power estimation을 사용하여 분석하였고, 전체 시스템이 아닌 core logic에 대해서만 synthesis 하여 output port를 buffer logic이나 result saver logic과 assign 해줄 수 없어 i/o & clock power에 대한 confidence level이 낮은 결과를 확인하게 되었다.

제안된 BitBAND 아키텍처는 PE 내에서 input과 weight의 (4-bit, 4-bit) precision을 최대로 지원하는데, 더 높은 bit-width를 갖는 data에 대해 4-bit group으로 나누어 서로 다른 PE array에 이를 할당하여 연산하는 효율적인 접근이 가능할 것으로 생각된다. 또한, 보다 정확한 분석 프로그램을 사용하여 결과를 비교할 수 있을 것으로 기대한다.

참고 문헌

[1] Gardner, Matt W., and S. R. Dorling, "Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences", *Atmospheric environment,* vol. 32, no. 14-15, pp. 2627-2636, 1998.

[2] Gottmer, M. L., “Merging reality and virtuality with microsoft HoloLens”, MS thesis, New Media & Digital Culture, Utrecht University, 2015.

[3] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton, "Imagenet classification with deep convolutional neural networks", *Communications of the ACM,* vol. 60, no. 6, pp. 84-90, 2017.

[4] Szegedy, Christian, et al., "Going deeper with convolutions", *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015.

[5] Bochkovskiy, Alexey, Chien-Yao Wang, and Hong-Yuan Mark Liao, "Yolov4: Optimal speed and accuracy of object detection", *arXiv preprint arXiv:2004.10934*, 2020.

[6] Wei, Yixuan, et al., "Contrastive Learning Rivals Masked Image Modeling in Fine-tuning via Feature Distillation", Microsoft, *arXiv preprint arXiv:2205.14141*, 2022.

[7] Han, Song, Huizi Mao, and William J. Dally., "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding", *arXiv preprint arXiv:1510.00149*, 2015.

[8] Esser, Steven K., et al., "Learned step size quantization", *arXiv preprint arXiv:1902.08153*, 2019.

[9] Défossez, Alexandre, Yossi Adi, and Gabriel Synnaeve, "Differentiable model compression via pseudo quantization noise", *arXiv preprint arXiv:2104.09987,* 2021.

[10] Sharma, Hardik, et al., "Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural network.", *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, IEEE, 2018.

[11] Ryu, Sungju, et al., "Bitblade: Area and energy-efficient precision-scalable neural network accelerator with bitwise summation", *Proceedings of the 56th Annual Design Automation Conference*, 2019.

[12] Rastegari, Mohammad, et al., "Xnor-net: Imagenet classification using binary convolutional neural networks", *European conference on computer vision*, 2016

[13] Zhang, Jintao, Zhuo Wang, and Naveen Verma, "In-memory computation of a machine-learning classifier in a standard 6T SRAM array", *IEEE Journal of Solid-State Circuits*, vol. 52, no. 4, pp. 915-924, 2017.

[14] Kim, Hyeonuk, et al., "Nand-net: Minimizing computational complexity of in-memory processing for binary neural networks", *2019 IEEE international symposium on high performance computer architecture (HPCA)*, IEEE, 2019.

[15] Knag, Phil C., et al., "A 617-TOPS/W all-digital binary neural network accelerator in 10-nm FinFET CMOS", *IEEE Journal of Solid-State Circuits,* vol. 56, no. 4, pp. 1082-1092, 2020.

Abstract

This study applies the concept of convolution decomposition to the BitBlade architecture, which is a multi-precision hardware accelerator core that uses 2-bit multiplication units, in order to support binary neural networks and 1-bit precision in 1-bit multiplication units. We propose a multi-precision hardware accelerator core. Convolution decomposition was originally proposed for converting XNOR-net to NAND-net for in-memory computation of BNN. Using this idea, it is possible to adopt a bitbrick structure that only uses the and gate for 1-bit precision, rather than multiplexing the result of the xnor gate and the and gate. In addition, for improved performance, we have added logic to optimize the logic and calculate two dynamic terms simultaneously, resulting in accurate calculation with the same level of latency. This resulted in an effective reduction of resources.

This paper presents information about the implementation of a hardware accelerator core that supports multi-precision without overhead for latency, throughput, and resources.

Keywords: DNN, Hardware accelerator, Multi-precision, BitBlade architecture, XNOR-net