전기 공학설계 프로젝트 결과 보고서

딥러닝 알고리즘의 하드웨어 설계 및 실습

2016-12457 전기정보공학부 정시우

서론

딥러닝(deep learning)이 여러 분야에 적용되면서 딥러닝의 중요성은 꾸준히 증가 중이다. 따라서 딥러닝 연산을 위한 하드웨어 또한 중요해졌다. 기존 CPU의 연산능력에는 한계가 있고 GPU는 전력 소모가 심해서 FPGA를 이용하면 빠르면서도 전력소모가 적은 딥러닝 가속기(deep learning accelerator)를 만들 수 있다.

최적화된 딥러닝 가속기를 설계하기 위해서는 FPGA에서 소모되는 자원을 고려하지 않을 수 없다. 네트워크 구조는 정말 다양하기 때문에 네트워크에 따라 최적화를 하는 방법 또한 다르다. 해당 연구는 SISR(Single Image Super Resolution)을 위해 설계된 네트워크에서 최대한 자원(LUT, DSP 등)을 줄이는 것을 목표로 한다. 네트워크에 따른 최적화를 booth multiplier 근사하는 것을 통해 실행한다.

본론에서는 네트워크의 구조와 booth multiplier, double MAC(Multiplier-accumulator) 등 자원을 줄이기 위한 기법을 소개하고 결과에서는 실제 기법을 적용해서 얻은 여러가지 표와 이미지들을 정리한다. 결론에서는 연구를 요약하고 한계점, 의의를 다룬다.

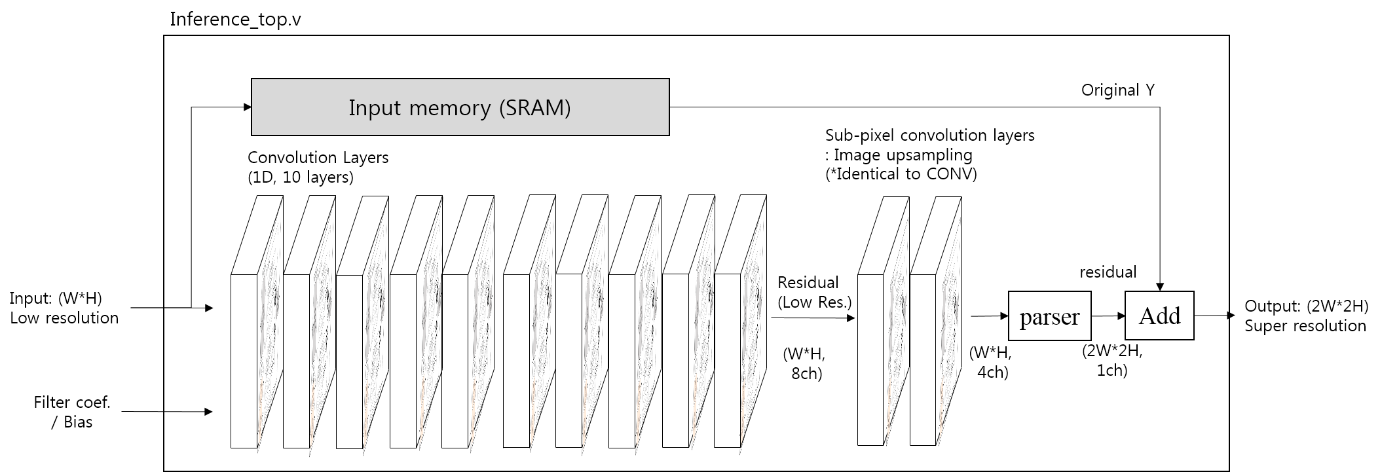
본 연구에는 Zynq UltraScale+ ZCU102 Evaluation Board를 이용했고 해당 보드에는 DSP48E2가 사용되었다. 또한 해당 연구에 이용된 네트워크는 Accurate Image Super-Resolution Using Very Deep Convolutional Networks[[1]](#endnote-1)에 제시된 네트워크의 구조를 따른다.

연구에 적용된 MAC을 이용하면 40%~47%의 LUT를 줄일 수 있었다.

본론

1. 네트워크 구조

본 연구에서 이용한 SISR을 위해 이용하는 데이터는 12bit(a sign bit, 3 integer bits, 8 fractional bits)이다. Fig 1은 전체 네트워크 구조이다. 해상도를 늘릴 이미지로는 채널 1개(monochrome)를 input image로 이용한다. 첫 10개의 레이어는 각각 output channel이 8개인 convolutional layer고 activation function으로는 ReLU(Rectified Linear Unit)를 이용한다. 10개의 레이어에서 사용하는 필터(filter)의 크기는 3x1, 7x1을 교차해가면서 쓴다. 마지막 2개의 레이어는 sub-pixel convolution layers로 각각 output channel이 4개인 convolutional layer이고 activation function은 따로 없이 총 12개의 레이어를 통과한다. 이후 parser를 통해 residual image를 기존 이미지와 합쳐서 output을 얻는다.

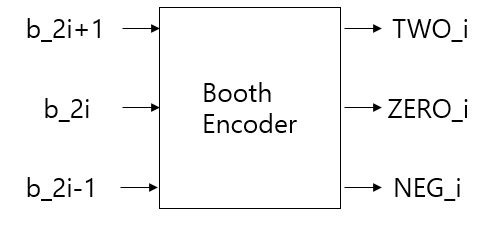
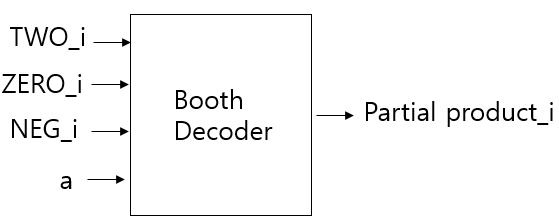


**Fig 1**: SISR CNN 구조

2. 4-Radix booth multiplier(이하 booth multiplier)

Booth multiplier는 1bit씩 bit 자릿수를 증가시켜 가면서 연산하는 Baugh Wooley와는 다르게 2bit씩 자릿수를 증가시킨다. Figure는 Baugh Wooley multiplier와 booth multiplier의 차이이다. Fig2는 Booth multiplier가 encoding, decoding 과정을 통해 partial product를 계산한다는 것을 보여준다.

기존 multiplier는 operand에 1 혹은 0을 곱해서 partial product를 계산한다면 booth multiplier는 기존 operand에 encoding 결과에 따라 -2부터 2까지의 정수를 곱한다. 기존 방식과 비교해서 shifter와 inverter를 추가로 필요로 하지만 필요한 partial product의 더하기 연산이 절반으로 줄어들면서 필요한 자원을 최소화 시킬 수 있다. 연산할 bit가 충분히 많으면 critical path 역시 adder를 ripple carry adder를 사용시 줄어들어서 delay가 줄어들게 된다.[[2]](#endnote-2)

(a) (b)

**Fig 2**: (a) Booth encoder. (b) Booth decoder.

3. Double MAC[[3]](#endnote-3)

본 연구에 사용되는 데이터끼리의 연산은 주로 곱셈, 덧셈이다. CNN 특성상 곱셈에 자원을 많이 사용하기 때문에 곱셈에 이용되는 자원을 아끼는 것이 가장 중요하다. 보통 Single MAC을 이용하면 곱셈에 DSP 한 개 혹은 LUT 다수를 이용하게 된다.

그러나 만약 double MAC을 이용해서 곱셈을 해보면 2가지 연산에 DSP 한 개와 LUT 소수를 쓸 수 있다. 즉, 평균 자원 소모량을 줄일 수 있다. Double MAC은 2개의 signed integer를 1개의 unsigned integer와 곱셈을 할 때 유용하게 쓸 수 있다. 전 레이어가 activation function으로 ReLU를 이용했다면 무조건 feature data는 무조건 unsigned integer가 되므로 double MAC을 하드웨어에 이용한다면 자원 소모량을 줄일 수 있다.

본 연구에서는 *Double MAC: Doubling the Performance of Convolutional Neural Networks on Modern FPGAs*에서 제시된 double MAC과 다르게 guard bit을 따로 두지 않고 operand2는 무조건 unsigned integer이기 때문에 중간에 0을 11bit까지 줄어서 자원 소모량을 최소화 시켰다.

4. Approximation

이미지 데이터 특성상 실제 값과 연산을 통해 나온 값이 꼭 같지 않아도 데이터 값의 차이가 아주 적다면 눈으로 다른 점을 느끼기 어렵다. 따라서 적절한 근사를 통해 FPGA에 사용되는 자원을 최대한 아낄 수 있다. Partial product를 encoding시 LSB 몇 bit만을 xor 연산을 통해 계산한다면 큰 데이터 손실 없이 MAC 연산이 가능하다. *Design of Approximate Radix-4 Booth Multipliers for Error- Tolerant Computing[[4]](#endnote-4)*에서 제안된 Radix-4 Approximate Booth Encoder 2는 error rate 25%이면서 획기적으로 area를 줄이는 방법이다. 본 연구에서는 해당 방법을 이용해서 multiplier를 최적화 시켰다.

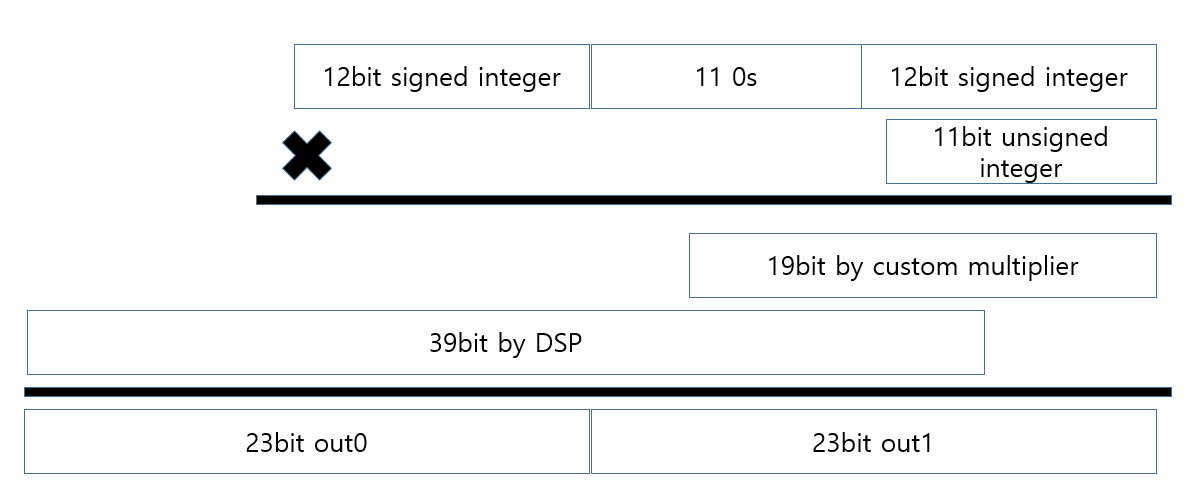
5. Designing deep learning accelerator

첫 번째부터 열 번째 레이어까지 모두 activation function으로 ReLU를 이용했기 때문에 double MAC을 이용할 수 있다. 데이터의 bitwidth가 모두 12bit이기 때문에 DSP48E2 1개에 2개의 weight를 모두 연산할 수 없어서 Fig 3과 같이 일부분은 booth multiplier를 이용했다. 나머지 2개의 레이어는 ReLU를 이용하지 않기에 accurate booth multiplier을 이용했다.

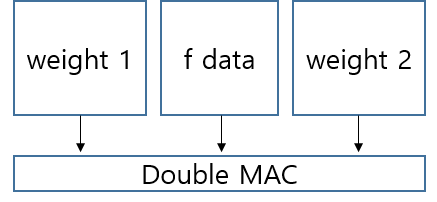
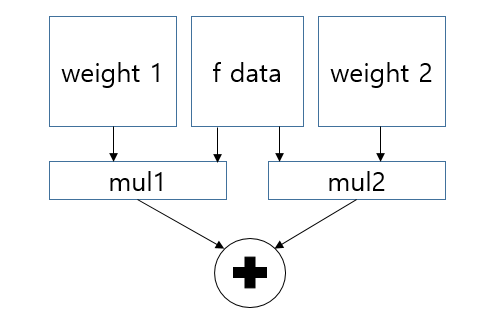
기존 IP에서는 쓰는 방식과 double MAC을 이용하는 방식은 Fig 4에서 확인할 수 있다. Double MAC에 사용되는 custom multiplier에는 BM(accurate Booth Multiplier), ABMp6(Approximate Multiplier, p=6), ABMp8, ABMp10이 쓰였다. 각 multiplier는 unsigned 11bit integer x unsigned 7bit integer이다. Fig 5는 각각의 multiplier 계산 방법을 나타낸 것이다.

본 연구에 사용된 네트워크는 convolutional layer에 ReLU를 이용한 레이어가 대부분이기에 자원의 대부분이 곱 연산에 사용이 된다. 따라서 위 방법과 같이 double MAC과 approximate booth multiplier를 쓴다면 필요한 자원을 많이 줄일 수 있다.

Table 1에서는 각각의 multiplier에 관한 오차 지표를 볼 수 있다.

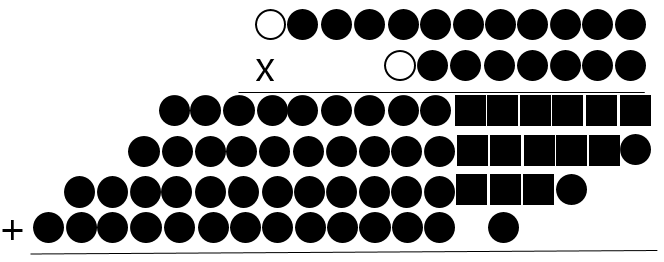
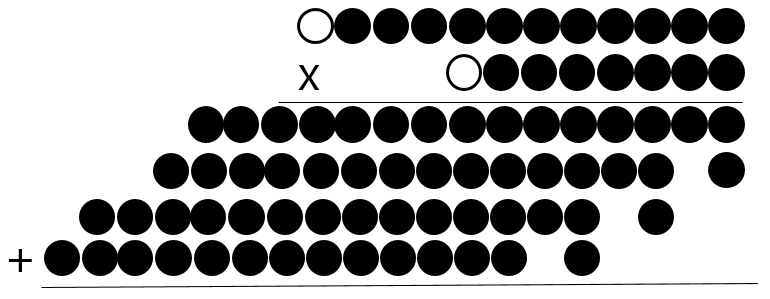


**Fig 3**: 본 연구에서 사용된 double MAC의 구조, custom multiplier는 booth algorithm을 이용해서 설계했다.

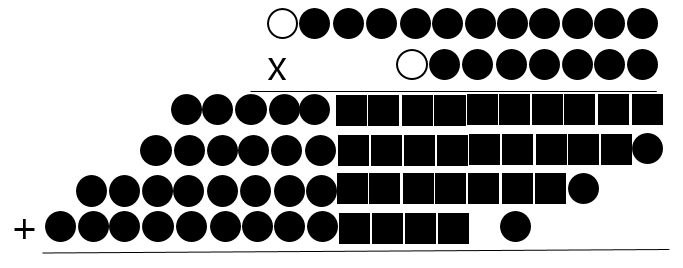
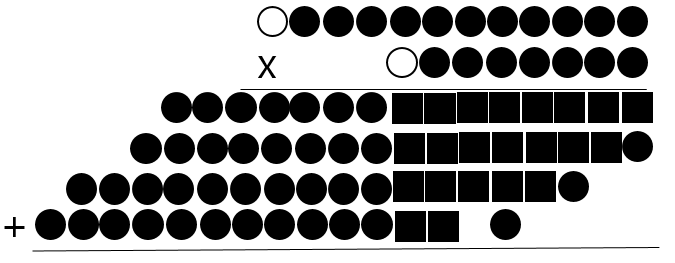


1. (b)

**Fig 4**: (a) 기존 IP의 MAC 구조. (b) double MAC을 이용한 MAC 구조.

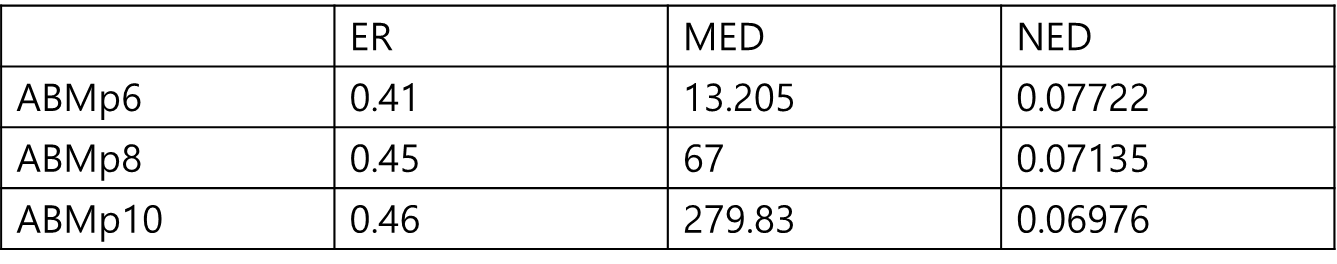


1. (b)

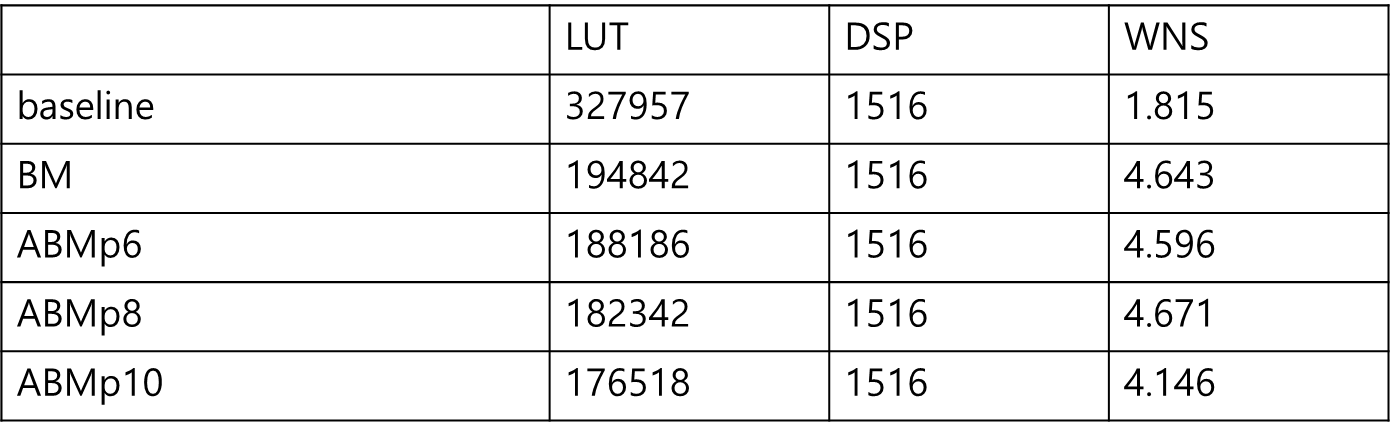


(c) (d)

**Fig 5**: 는 0을 나타내고 는 기존 operand 혹은 정확한 partial product를 나타낸다. 는 approximate partial product를 나타낸다.



**Table 1**: 각 multiplier를 적용시킨 double MAC의 ER(Error Rate), MED(Mean Error Distance), NED(Normalized Error Distance)



**Table 2**: 사용하는 DSP의 개수를 동일하게 했을 때의 baseline과 각 multiplier를 적용시킨 딥러닝 가속기의 LUT, DSP, WNS. Baseline에서 이용되는 LUT의 개수는 이용 가능한 LUT보다 많기에 실재로는 합성이 불가능하다.

6. Result

Table 2을 보면 기존 baseline과 본 연구에서 사용된 하드웨어의 자원 소모량과 WNS(Worst Negative Slack)을 알 수 있다. 자원도 approximate bit수를 증가시킴에 따라 줄어들고 double MAC을 이용하면 WNS역시 증가하기 때문에 timing margin이 더욱 확보되어서 성능 측면에서도 개선됐다고 볼 수 있다. 이번 연구에서는 비교하기 편하게 DSP의 개수를 모두 똑같이 맞춰줬는데 필요에 따라서 DSP를 늘리면서 LUT를 줄이는 것도 가능하다. 더욱 유연하게 자원관리를 할 수 있다.

Fig 6은 data bitwidth를 14bit로 하고 tensorflow를 이용해 얻은 결과와 본 연구에서 설계된 여러가지 multiplier를 이용한 하드웨어를 통해 SISR을 실행한 결과이다. 해당 결과의 PSNR은 Table 3를 통해 확인이 가능하다.

approximation에서 ER는 큰 차이는 없으나 ED가 늘어날수록 PSNR이 줄어들고 육안으로도 tensorflow로 얻은 이미지보다 결과가 좋지 않은 것을 알 수 있다. 필요한 정확도에 따라서 multiplier를 이용할 수 있다.

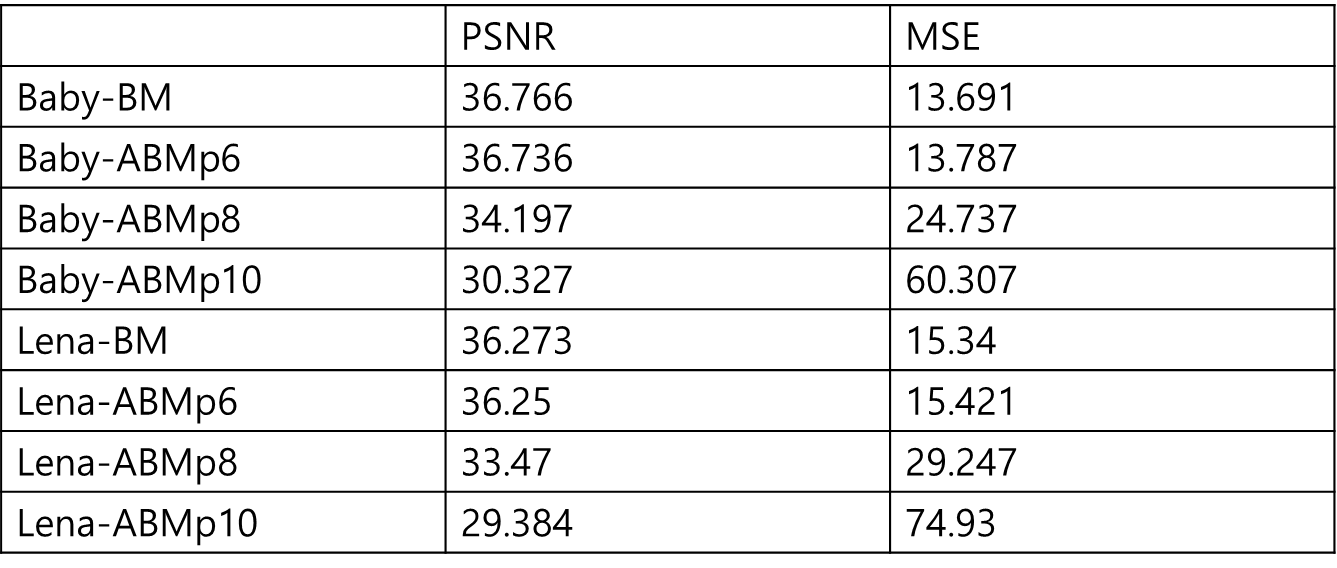
    

(a)

(b)

**Fig 6**: (a) baby. 왼쪽부터 순서대로 tensorflow를 이용해서 얻은 이미지, 각각 BM, ABMp6, ABMp8, ABMp10에서 얻은 이미지. (b) Lena. 왼쪽부터 순서대로 tensorflow를 이용해서 얻은 이미지, 각각 BM, ABMp6, ABMp8, ABMp10에서 얻은 이미지.



**Table 3**: Fig 6에 관한 PSNR

결론

성능은 CPU보다 뛰어나고 전력은 GPU보다 훨씬 덜 쓰는 장점이 있으면서 네트워크나 목적에 따라 다양하게 최적화를 시킬 수 있다는 장점으로 인해 여러가지 딥러닝 가속기 최적화 연구가 진행되는 중이다. 본 연구에서는 제시된 SISR CNN에 필요한 딥러닝 가속기를 최적화 시키는 데에 중점을 두었다.

기존 네트워크에 맞춰서 딥러닝 가속기의 multiplier에 double MAC, approximate booth multiplier 기법을 도입해서 기존 IP에 비해 자원 소모가 적으면서 뛰어난 성능의 딥러닝 가속기를 얻을 수 있었다.

Double MAC은 본 연구에서 이용한 네트워크처럼 전 레이어에서 ReLU를 이용하면서 convolutional layer를 통과시는 유용하나 같은 feature data를 공유하지 않는 fully connected layer, ReLU를 이용하지 않는 layer, 혹은 여러가지 pooling layer 등 타 레이어에는 이용하기가 어렵다는 단점이 있다. 또한 DSP의 성능이나 데이터의 bit 수에 따라서도 single MAC을 이용할 때보다 개선되지 않을 수 있기에 quantization이 추가로 필요할 수 있다. 이러한 제한에도 불구하고 DSP의 bitwidth와 data bitwidth가 적절하다면 필요한 자원 수를 획기적으로 줄일 수 있다. 또한 ReLU는 간단하면서도 강력한 activation function이고, convolutional layer는 실제로도 자주 쓰이는 레이어이면서 딥러닝 가속기의 자원을 많이 차지하기 때문에 본 연구에 제안된 방법으로 레이어당 FPGA에서 소모하는 자원을 최소화 시켜서 네트워크를 더욱 깊게 만드는 데에 큰 도움을 줄 것이다. 또한 DSP와 LUT간의 관리가 더욱 유연해진다.

Resource reusing, dynamic quantization과 같은 자원 소모를 줄이는 다른 기법을 적용하거나 성능을 개선시키는 여러가지 기법을 같이 목적에 따라 잘 혼합해 준다면 보다 최적화된 딥러닝 가속기 설계가 가능하다.

1. Jiwon Kim, Jung Kwon Lee and Kyoung Mu Lee, "Accurate Image Super-Resolution Using Very Deep Convolutional Networks," Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016. [↑](#endnote-ref-1)
2. Sumit Vaidya and Deepak Dandekar, ”DELAY-POWER PERFORMANCE COMPARISON OF MULTIPLIERS IN VLSI CIRCUIT DESIGN,” International Journal of Computer Networks & Communications (IJCNC), Vol.2, No.4, July 2010 [↑](#endnote-ref-2)
3. Dong Nguyen, Daewoo Kim and Jongeun Lee, “Double MAC: Doubling the Performance of Convolutional Neural Networks on Modern FPGAs,” Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017 [↑](#endnote-ref-3)
4. Weiqiang Liu, Senior Member, IEEE, Liangyu Qian, Chenghua Wang, Honglan Jiang, Jie Han, Senior, Member, IEEE, and Fabrizio Lombardi, Fellow, IEEE, “Design of Approximate Radix-4 Booth Multipliers for Error- Tolerant Computing,” IEEE TRANSACTIONS ON COMPUTERS, Vol.66, pp. 1435-1441, Feb 2017 [↑](#endnote-ref-4)