Using Fermat Number Transform to Accelerate Convolutional Neural Network

Weihong Xu[1,2], Xiaohu You[2], and Chuan Zhang[1,2,*]
[1]Lab of Efficient Architectures for Digital-communication and Signal-processing (LEADS)
[2]National Mobile Communications Research Laboratory, Southeast University, Nanjing, China
Email: {wh.xu, xhyu, chzhang}@seu.edu.cn

***Abstract*—Convolutional neural network (CNN), has achieved a significant breakthrough on image recognition and natural language processing. However, CNNs still suffer from the prohibitive computation complexity. Many efforts have been made to reduce the arithmetic complexity of direct convolution. Aiming at lower multiplication complexity, an efficient convolution architecture based on Fermat number transform (FNT) is proposed in this paper. We first present the FNT algorithm and Overlap-and-Add (OaA) method. To cope with convolution computation in CNNs, the FNT is extended to two-dimensional (2D) FNT and corresponding calculation methodology is introduced. The pipelined FNT architecture is also proposed for efficient realization of FNT. Then an overall architecture of CNN accelerator is given based on OaA FNT. Complexity analysis has demonstrated that the proposed OaA FNT convolution method reduces $5.59\times$ and $2.56\times$ of multiplication complexity compared to direct convolution and OaA FFT method. We also conduct evaluation on the FPGA platform Xilinx Virtex-7 XC7VX485t and comparison is made with respect to convolution throughput and resource efficiency (GOP/s/DSP). The proposed architecture achieves $1.41\times$ convolution throughput with $3.05\times$ less DSPs compared to the state-of-the-art design. $4.29\times$ improvement is gained in terms of the resource efficiency.**

***Index Terms*—Fermat number transform (FNT), Overlap-and-Add (OaA), pipelining, convolutional neural network (CNN).**

I. Introduction

Convolutional neural network (CNN) is becoming more and more popular in many fields. With the aid of powerful CNN, unprecedented performance has been obtained in recognition, reasoning, knowledge, planning, and so on [1]. One recent news is Google Deepmind's program first beat a world-top professional human Go player. However, CNN is a kind of computationally intensive algorithm. Many applications of CNN still suffer from the prohibitive computation complexity, which poses a big challenge to hardware implementation on embedded platforms with small footprint.

Recently, a considerable amount of work has been carried out to accelerate convolution computation in CNNs. Fast Fourier Transform (FFT) is an effective method to reduce the convolution complexity due to its cyclic convolution property. And significant acceleration has been achieved on GPU with FFT [2]. However the FFT-based convolution method only gains speedup effects when the sizes of input image and filter are similar and large, which demands large memories. Hence, FFT-based convolution is not suitable for area-constraint embedded devices. To overcome the sequence-length problem, Overlap-and-Add (OaA) FFT convolution is proposed in [3]. The idea is to divide large images into small tills and then apply FFT-based convolution to each till.

Low complexity and high throughput are two key metrics for boosting CNNs. There exist many drawbacks in current FFT-based schemes. One is the fact that FFT is performed in the complex number field. Complex multiplication is required and occupies most of the computation time. Moreover, additional registers are required to store all the power of twiddle factors as well as intermediate complex numbers. Acceleration strategies in [2, 3] are both based on floating point. However the other shortcoming is that substantial roundoff noise will be introduced at the output around 8 to 16 bits for hardware implementation of fixed-point FFT [4].

For state-of-the-art CNNs, convolution operation takes up over 90% of the total computation complexity [5, 6]. Compared to FFT, Fermat number transform (FNT) is more computationally efficient and hardware-friendly to accelerate convolution. FNT is performed on the finite field rather than complex field and can obtain exact results [7]. Besides, the similarity of algorithms and hardware structures between FNT and FFT makes it easy to implement and fine-tune corresponding architectures.

This paper focuses on an efficient low-complexity implementation to accelerate CNNs based on FNT. The overall architecture of 2-Dimensional (2D) FNT and OaA is proposed by exploiting advanced FFT structures and algorithms. The selection of design parameters is given with respect to the scenarios of CNNs. Data conversion scheme for FNT is also presented to obtain accurate results of convolution. Experiment results demonstrate the advantages of FNT in terms of complexity and throughput.

The remainder of this paper is organized as follows. Background of FNT and OaA method is introduced in Section II. In Section III, efficient FNT-based 2D convolution is first proposed and then the detailed implementation of accelerating architectures for CNNs based on OaA FNT is presented. Both complexity analysis and hardware implementation are shown in Section IV. Section V concludes the entire paper.

II. Preliminaries

A. Convolution in CNNs

Convolution is an effective operation to extract the features from input images in CNNs. The notations of spatial dimensions in convolutional layer are listed in Table I.

978-1-5090-6625-4/17/$31.00 ©2017 IEEE

TABLE I
PARAMETERS DESCRIPTION OF CONVOLUTIONAL LAYER

Description	Parameters
Input image size	$N_{img} \times N_{img}$
Filter kernel size	$K \times K$

Convolutional layer of CNN operates a 2D convolution for each input image $\mathbf{I}$. And the output feature map $\mathbf{O}$ can be computed as the following equation:

$$\mathbf{O}(x,y) = \sum_{x'=0}^{K-1}\sum_{y'=0}^{K-1} \mathbf{I}(x-x^{'}, y-y^{'})\mathbf{W}(x^{'}, y^{'}), \quad (1)$$

where $\mathbf{W}$ denotes the pre-trained convolution filter.

B. Fermat Number Transform (FNT)

Similar to FFT, Fermat number transform is a kind of number theoretic transforms (NNTs) [8]. The calculation of an N-length real sequence $x(n)$ can be defined as follows:

$$X(k) = \mathcal{F}(x(n)) = \sum_{n=0}^{N-1} \left\langle x(n)\alpha^{nk} \right\rangle_{F_t}, \quad (2)$$

where $k = 0, 1, \cdots, N-1$. F_t, $\langle a \rangle_b$ and $\mathcal{F}$ denote the t-th Fermat number, modular arithmetic and the 1D FNT operation, respectively. And α equals to 2 in this paper since it can be calculated very efficiently through shift registers.

The corresponding inverse Fermat number transform is expressed as:

$$x(n) = \mathcal{F}^{-1}(X(k)) = \frac{1}{N}\sum_{k=0}^{N-1} \left\langle X(k)\alpha^{-nk} \right\rangle_{F_t}, \quad (3)$$

where $\mathcal{F}^{-1}$ denotes the inverse FNT operation.

To obtain the correct results, the parameters of FNT should be carefully selected [7]. The t-th Fermat number in Eq. (2) is calculated by:

$$\begin{cases} b = 2^t, \\ F_t = 2^b + 1. \end{cases} \quad (4)$$

Note that b is the number of bits of filtering data and weights that will not cause overflow. Under the situation of $\alpha = 2$, the maximum transform length $N = 2^{t+1}$.

C. OaA Convolution Method

Although the transformed FNT sequences do not have any physical meanings, we can make use of its convolution property [7] to accelerate convolution computation.

The direct convolution for two sequences can be calculated through taking the inverse FNT of Hadamard product (point-wise product) of their FNT sequences. But this is efficient only when the sizes of data and weights are near.

For the convolution in CNNs, the filter size is normally much smaller than the input image. OaA method can divide long convolution into multiple short convolutions:

$$y(n) = \sum_{k} \mathcal{F}^{-1}\left\{\mathcal{F}(x(n-kL)) .* \mathcal{F}(h(n))\right\}, \quad (5)$$

where L is an arbitrary segment length and $L \leq N - K + 1$.

III. PROPOSED FNT CONVOLUTION ARCHITECTURES

A. Multiple Dimensional FNT

To process 2D input images, we can extend the 1D FNT in Eq. (2) to 2D FNT [9] by the following expression:

$$\mathbf{X}(p,q) = \mathcal{F}_2(\mathbf{x}(m,n)) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} \langle \mathbf{x}(m,n)\alpha^{mp}\alpha^{nq} \rangle_{F_t}, \quad (6)$$

where $\mathcal{F}_2$ denotes the 2D FNT and $\mathbf{x}$ is an $M \times N$ 2D matrix.

According to [3], we can 2D FFT based on 1D FFT. The 2D FNT can be computed in a similar fashion:

1) First, do 1D FNT on each row of the input image.
2) Then, do 1D FNT on each column to the intermediate results from 1).

Thus the 2D FNT of an $N_{img} \times N_{img}$ input image requires a total of $2 \times N_{img}$ 1D FNTs. For 2D IFNT, the sequence to perform row-wise and column-wise 1D FNT is inverse compared to 2D FNT.

B. Basic FNT Modules

By observing Eq. (2), we can see that FNT is a FFT-like algorithm. Therefore, the FNT also has a fast transform similar to FFT when N is highly composite [7]. The data flow graph (DFG) of the 8-point radix-2 FNT is given in Fig. 1 as an example. It has the very similar structure like decimation-in-time (DIT) radix-2 FFT algorithms.

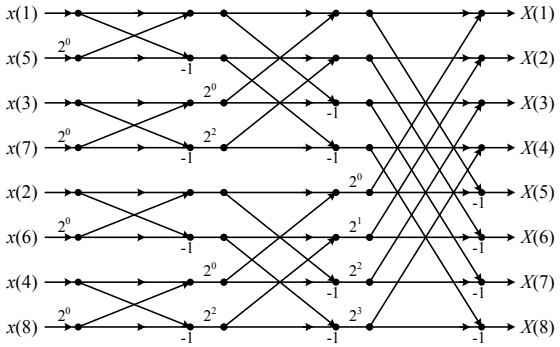
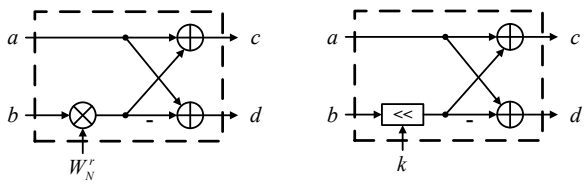


Fig. 1. Data flow graph (DFG) of 8-point radix-2 FNT.

The fast FNT can be implemented by recursively applying the butterfly modules. Fig. 2 illustrates the butterfly modules of FNT and FFT. Compared to FFT, the k-bit shift register replaces the complex multiplier and each FNT butterfly module only requires 2 real adders instead of 2 complex adders.



(a) FFT butterfly (b) FNT butterfly

Fig. 2. Butterfly modules of FFT and FNT.

The straightforward implementation of an N-point FNT consists of $\log_2 N$ stages, each containing $N/2$ butterfly

modules. To reduce hardware complexity, the existing efficient FFT-like architectures [10, 11] can be fully utilized due to the similarity between FFT and FNT. The pipelined implementation of N-point 1D FNT is illustrated in Fig. 3, which consists of $\log_2 N$ FNT butterfly modules.

The FNT length is selected as $N = 32$ in this paper. We have $b = 16$ and $F_4 = 2^{16} + 1$. It has been proven that 16-bit quantization will not introduce performance degradation for CNNs [12]. Besides, 32-point FNT is sufficient for OaA FNT-based convolution since most commonly used filter sizes vary from 3×3 to 11×11.

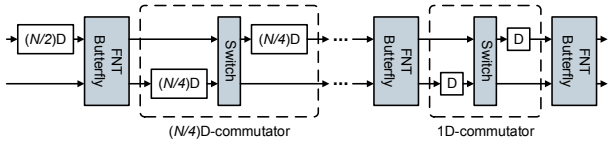


Fig. 3. Pipelined N-point 1D FNT module.

C. Data Conversion

The FNT is performed in the finite field rather than real number field. Consequently, preprocessing of the input and output data is necessary to obtain accurate results.

For a t-th Fermat number $F_t = 2^b + 1$, the total range of filtering data is $\left[0, 2^b\right]$ and its resolution is 2^{-b}. In order to fit the convolution in state-of-the-art CNNs, the input data need to be transformed into the finite field before feeding into the FNT calculating modules. One possible scheme is to shift input data x with range $\left[-2^{b-1}, 2^{b-1}\right]$ into $\left[0, 2^b\right]$ as follows:

$$x^{'} = \begin{cases} x, & \text{if } 0 \le x \le 2^{b-1}, \\ x + F_t, & \text{if } -2^{b-1} \le x < 0. \end{cases} \tag{7}$$

After finishing computation, the output data can be conversed back to real numbers as Eq. (8).

$$y = \begin{cases} y^{'}, & \text{if } 0 \le y^{'} \le 2^{b-1}, \\ y^{'} - F_t, & \text{if } 2^{b-1} \le y^{'} \le 2^b. \end{cases} \tag{8}$$

Here is an example. Assume that we have $x = [1, -2, 3, -4]$, $h = [1, 1, 1, 1]$ and $F_2 = 2^4 + 1 = 17$. Sequence x first needs to be transfered into $x^{'} = [1, -2 + 17, 3, -4 + 17]$ and the FNT based convolution result is $y^{'} = [1, 16, 2, 15, 14, 16, 13, 0]$. According to Eq. (8), the final result is $y = [1, 16-17, 2, 15-17, 14-17, 16-17, 13-17, 0]$, which is identical to the correct result.

D. Efficient Convolution Architectures

The computational intensive convolution in CNNs can be accelerated by stacking the aforementioned modules. Fig. 4 shows the overall block diagram of proposed efficient convolution architectures based on OaA FNT. The speedup of convolution layers in CNNs is realized as follows:

1) $N_{img} \times N_{img}$ input image is first conversed into the finite field by *Data Conversion Unit* according to Eq. (7) and then split into $\lceil \frac{N_{img}^2}{L^2} \rceil$ tiles. Each tile has a size of $L \times L$, where $L = N - K + 1$.
2) Pad each $L \times L$ image tile to $N \times N$ and then feed it into *2D FNT Kernel*, which is composed of $2N$ N-point *1D FNT Modules* and one *Transpose Unit*. The first N *1D FNT Modules* perform row-wise FNT for the $N \times N$ tile. The intermediate results are transposed by *Transpose Unit* and then fed into the second N *1D FNT Modules* to perform column-wise FNT.
3) The *Multipliers Unit* contains $2N$ real multipliers and receives data from *2D FNT Kernel* to perform the Hadamard product of each output 2D FNT tile with 2D FNT of corresponding filter.
4) The *2D IFNT Kernel* implements the 2D inverse FNT of received $N \times N$ data. The difference is that it first performs column-wise and then performs row-wise FNT.
5) Iterate Step 2) to Step 4) until the calculation of $N_{img} \times N_{img}$ input image is finished. For each output tile of *2D IFNT Kernel*, a $(K-1)$ point-width region on the boundary overlaps with its surrounded tiles.
6) The $\lceil \frac{N_{img}^2}{L^2} \rceil$ tiles are concatenated according to OaA algorithm. *Data Conversion Unit* converses the convolution output back to real numbers according to Eq. (8).

IV. Experiment Results

A. Complexity Analysis

In order to compare the arithmetic complexity of different convolution schemes, the comparison between direct, OaA FFT-based and OaA FNT-based convolution in terms of complexity is presented in the following analysis:

- **Direct convolution**. The computation complexity for direct method is estimated as $\mathcal{O}(N_{img}^2 K^2)$.
- **OaA FFT-based convolution**. The input feature map is split into $\lceil \frac{N_{img}^2}{(N-K+1)^2} \rceil$ tiles with $N \times N$ FFT. The complexity of 2D FFT for each tile is $\mathcal{O}(N^2 \log_2 N)$. Hence the complexity of OaA FFT-based method is $\mathcal{O}(N_{img}^2 \log_2 N)$. Note that this is complex operation.
- **OaA FNT-based convolution**. Analogous to OaA FFT-based method, the complexity of OaA FNT-based method is about $\mathcal{O}(N_{img}^2 \log_2 N)$. Note that this is real operation.

OaA FNT-based convolution substitutes a great amount of complex multiplications for shift, modular operation and real multiplication. Fig. 5 illustrates the multiplication complexity of convolution for VGG-16 [1]. Proposed OaA FNT convolution only requires 17.9% and 39.07% complexity compared to direct convolution and OaA FFT convolution in [3].

B. Hardware Implementation

We implement the proposed design on Xilinx Virtex-7 XC7VX485t. We use the pre-trained VGG-16 module [1] and all convolution kernel weights are quantized as 16-bit. The frequency is 150 MHz. Table II lists the FPGA utilization. Only the convolutional layer part is realized on FPGA.

Performance comparison with the other advanced acceleration designs is also conducted and summarized in Table III. It can be seen that proposed design increases $1.41\times$ convolution throughput with $3.05\times$ less DSPs compared to

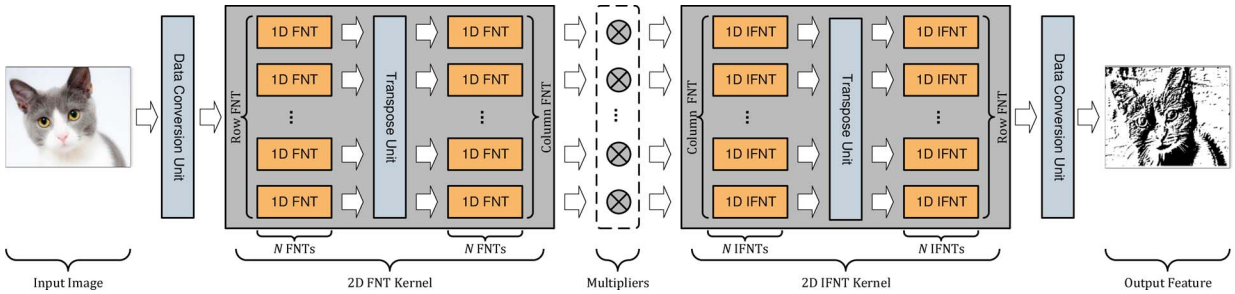


Fig. 4. Overall architecture of OaA FNT-based CNN accelerator.

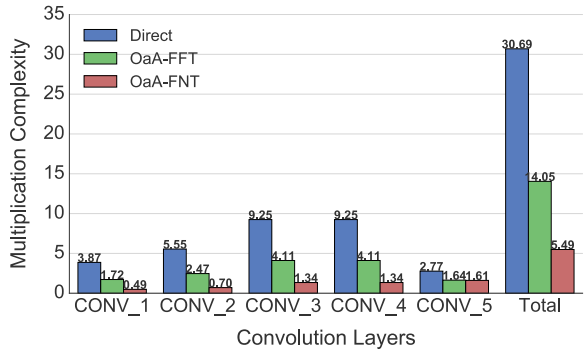


Fig. 5. VGGNet convolution complexity in terms of multiplications (GOPs).

TABLE II
FPGA RESOURCE UTILIZATION

Resource	LUT	FF	DSP	BRAM
Used	188,928	142,592	256	512
Available	303,600	607,200	2,800	2,060
Utilization	62.23%	23.48%	9.14%	24.85%

[12]. Furthermore, our design achieves $4.29\times$ and $36.89\times$ resource efficiency (GOP/s/DSP) compared to [12] and [13].

TABLE III
FPGA COMPARISON OF VARIOUS CNN ACCELERATION DESIGNS

Design	[13]	[12]	This work
Platform	Virtex-7 VX485t	Zynq XC7Z045	Virtex-7 VX485t
Clock(MHz)	100	150	150
CNN Model	AlexNet	VGG16-SVD	VGG16
Quantization	32-bit float	16-bit fixed	16-bit fixed
LUT	186,251	182,616	188,928
FF	205,704	127,653	142,592
DSP	2,240	780	2,56
BRAM	1,024	486	512
CONV Throughput (GOP/s)	61.62	187.80	264.60
Resource Efficiency (GOP/s/DSP)	0.028	0.241	1.033

V. CONCLUSION

In this paper, a 2D FNT convolution for CNNs is presented. Then an efficient OaA FNT-based acceleration architecture with high throughput is proposed. Experiment results have demonstrated the advantages. Future work will be directed towards optimization and complete FPGA implementation.

ACKNOWLEDGEMENT

This work is supported in part by NSFC under grant 61501116, Jiangsu Provincial NSF under grant BK20140636, Huawei HIRP Flagship under grant YB201504, the Fundamental Research Funds for the Central Universities, State Key Laboratory of ASIC & System under grant 2016KF007, ICRI for MNC, and the Project Sponsored by the SRF for the Returned Overseas Chinese Scholars of MoE.

REFERENCES

[1] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
[2] M. Mathieu, M. Henaff, and Y. LeCun, "Fast training of convolutional networks through FFTs," *arXiv preprint arXiv:1312.5851*, 2013.
[3] C. Zhang and V. Prasanna, "Frequency domain acceleration of convolutional neural networks on CPU-FPGA shared memory system," in *Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2017, pp. 35–44.
[4] A. V. Oppenheim and C. J. Weinstein, "Effects of finite register length in digital filtering and the fast fourier transform," *Proceedings of the IEEE*, vol. 60, pp. 957–976, 1972.
[5] Y. H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 52, pp. 127–138, 2017.
[6] W. Xu, Z. Wang, X. You, and C. Zhang, "Efficient fast convolution architectures for convolutional neural network," in *Proceedings of IEEE International Conference on ASIC (ASICON)*, 2017, accepted.
[7] R. Agarwal and C. Burrus, "Fast convolution using fermat number transforms with applications to digital filtering," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 22, pp. 87–97, 1974.
[8] R. Agarwal and J. Cooley, "New algorithms for digital convolution," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 25, pp. 392–410, 1977.
[9] I. S. Reed, T. K. Truong, Y. S. Kwoh, and E. L. Hall, "Image processing by transforms over a finite field," *IEEE Transactions on Computers*, vol. C-26, pp. 874–881, 1977.
[10] M. Ayinala, M. Brown, and K. K. Parhi, "Pipelined parallel FFT architectures via folding transformation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, pp. 1068–1081, 2012.
[11] C. Zhang, J. Yang, X. You, and S. Xu, "Pipelined implementations of polar encoder and feed-back part for SC polar decoder," in *Proc. IEEE Inter. Symp. on Circ. and Syst. (ISCAS)*, 2015, pp. 3032–3035.
[12] J. Qiu, J. Wang, S. Yao, K. Guo, B. Li, E. Zhou, J. Yu, T. Tang, N. Xu, S. Song *et al.*, "Going deeper with embedded FPGA platform for convolutional neural network," in *Proceedings of ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2016, pp. 26–35.
[13] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing FPGA-based accelerator design for deep convolutional neural networks," in *Proceedings of ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2015, pp. 161–170.