A Winograd-Based Highly-Parallel Convolution Engine for 8-bit CNN Acceleration

Yong-Tai Chen
Department of Electrical Engineering
National Tsing Hua University
Hsinchu, Taiwan
ytchen0922@gapp.nthu.edu.tw

Yu-Feng Ou
Department of Electrical Engineering
National Tsing Hua University
Hsinchu, Taiwan

Chao-Tsung Huang
Department of Electrical Engineering
National Tsing Hua University
Hsinchu, Taiwan

***Abstract*—Convolutional neural network (CNN) accelerators for computational imaging typically use 8-bit fixed-point models for efficient computation, but the convolution engine still dominates the chip area. Quantizing models in lower bitwidths can cut down resource demand effectively, but it results in a significant loss of output quality. Another approach to reducing computational complexity is through Winograd convolution which lessens the demand for logic gates without diminishing model quality. Nevertheless, the resource reduction ratio of Winograd convolution declines with input bitwidths, and it needs even more gates than direct convolution at 8-bit. In this paper, we realize an area-efficient convolution engine for 8-bit computational imaging models by considering Winograd convolution and quantization jointly. First, we elaborate hardware sharing techniques for highly-parallel Winograd convolution. Then we propose an uneven scheme for Winograd-domain quantization that yields only up to 0.16 dB of PSNR drop on computational imaging models. Finally, we implement a highly-parallel Winograd convolution engine for 8-bit CNN inference. Synthesized with TSMC 40nm technology, the engine uses 2.17M of logic gates for delivering 5.12 TOPS of inference capability, saving 29.5% and 41.1% of logic gates compared to a direct convolution engine and a naïve Winograd implementation respectively. On modified FFDNet and EDSR baselines, it achieves up to Full HD 20 fps with merely 0.09 dB of PSNR drop on average.**

***Index Terms*—Winograd convolution, highly-parallel, computational imaging, CNN, quantization**

I. Introduction

Convolutional neural networks have demonstrated exceptional quality in image processing applications, such as denoising [1], and super-resolution [2]. Different from networks for object recognition, these networks keep high-resolution feature maps to generate detailed textures and demand numerous computation resources. Computational imaging CNN accelerators [3], [4] use quantized 8-bit fixed-point models for their lower resource requirements. However, further optimization through bitwidth reduction is unfavorable. With 8-bit input and output images, the output qualities drop rapidly as the bitwidths get lower.

Another approach to saving computation resources is Winograd convolution [5]. Introduced to CNN acceleration by [6], Winograd convolution can be conceptually divided into three steps as shown in Fig. 1: domain transformations, element-wise multiplications, and reconstruction transformation. It has optimum performance in small and consistent filter size, meeting the trends of state-of-the-art CNN models to use deep 3×3 convolutional layers for better output qualities. As a consequence, it is widely used in GPU [7], edge processors [8], and FPGA [9], [10] CNN accelerator designs.

However, for application specific integrated circuit (ASIC) accelerators, the effectiveness of Winograd algorithm decline with model bitwidth due to their customized computational units. Unlike reducing multiplier or DSP usage, the saving of multiplications does not reflect directly in the hardware resources in ASICs. As shown in Fig. 3(a), naïve implementations of Winograd convolution require more gates than direct convolution engines below eight bits. Two kinds of overheads in Winograd convolution process emerge as the bitwidths become lower: transform overheads and bitwidth growth overheads. The former refers to additional hardware required by the domain transforms. Adders and shifters of the transformations are no longer neglectable once the multipliers become smaller due to lower input bitwidths. The latter is larger multipliers than direct convolution due to dynamic range growth — it requires extra bits to represent the transformed values. Therefore, finding a way to solve these overheads is crucial for efficient circuit implementation as using 8-bit models becomes the trend of computational imaging CNN accelerators nowadays.

In this paper, we jointly consider Winograd convolution with quantization and present an area-efficient convolution engine design for 8-bit computational imaging CNN models. We first explore hardware sharing opportunities to cut down the transform overheads and reach 16.6% of area reduction compared to direct convolution without loss of quality. Then, we propose an uneven scheme for Winograd-domain quantization to reduce the bitwidth growth overheads. It causes only a 0.16 dB of PSNR drop on an 8-bit denoising model and 0.07 dB of PSNR drops on super-resolution models. Finally, we implement a Winograd-based highly-parallel convolution engine combining the methods above. Synthesized with TSMC 40nm technology, the engine uses 2.17M of logic gates and provides 5.12 TOPS of inference capability, achieving up to 20 fps on the super-resolution network. Our design saves 29.5% and 41.1% of gate count compared to an 8-bit direct convolution engine and a naïve Winograd convolution implementation, with only 0.09 dB of PSNR drop on average.

This work was supported by Grant MOST 109-2622-8-007-022.
We thank to National Center for High-performance Computing (NCHC) for providing computational and storage resources.

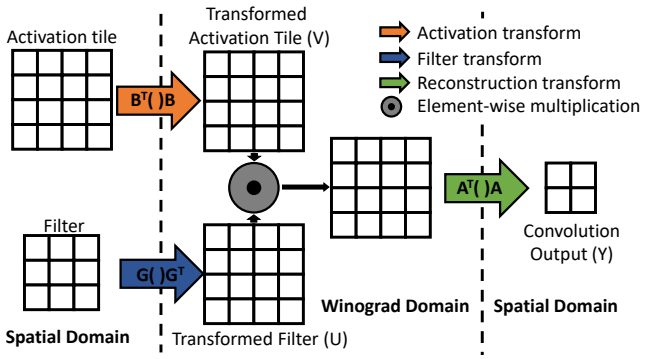

Fig. 1. Winograd convolution of 2×2 output tile. First, the domain transforms are done to the filter and the activation tile. Then element-wise multiplication is performed on the transformed data. Finally, the reconstruction transform converts the multiplication output back to the spatial domain.

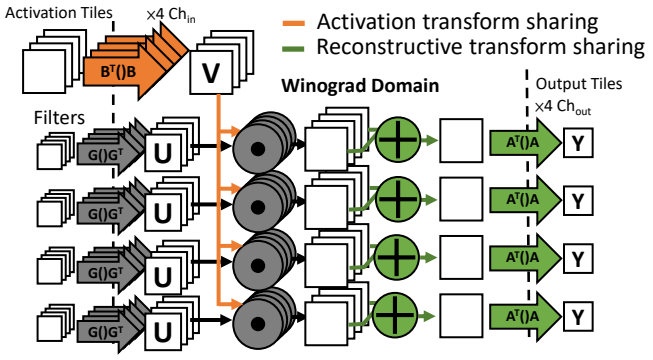

Fig. 2. Hardware sharing of Winograd transforms for four-channel inputs and four-channel outputs. Each transformed activation tile is broadcast and multiplied with four transformed filters to share the activation transform. The multiplication outputs are added up in Winograd domain to share the reconstruction transform among channels.

II. Transform Hardware Sharing

In this section, we elaborate on the transformation hardware sharing technique. We first demonstrate how the transform units are shared among channels, and then analyze the channel parallelism.

A. Sharing Opportunities

Winograd convolutions use more additions and some shiftings in the transformation stages to achieve fewer total multiplications. Nevertheless, the cost gap between addition and multiplication shrinks as input bitwidth gets lower, thus the effectiveness of this algorithm diminishes. The overheads outweigh the area benefit of multiplication reduction below 8-bit as shown in Fig. 3(a). Fortunately, there exist plenty of reusing opportunities in the multi-layer tensor filtering process. A. Podili, C. Zhang and V. Prasanna [9] transform the filters offline to reuse them over the whole feature map. On the other hand, input and output activation can hardly be calculated in advance, but the on-the-fly activation and reconstruction transforms can be shared among channels as shown in Fig. 2. As multiple output channels are computed simultaneously, one transformed activation tile can be multiplied with several transformed filters. For the reconstruction transform, we use a strategy similar to [11] which treats the element-wise multiplication in Winograd domain as vector-matrix multiplication. However, [11] calculates the multiplications sequentially to exploit the sparsity in Winograd domain filter but makes low utilization for reconstruction transform circuits. On the contrary, our strategy is to process the multiplications at once to fully utilize the transform hardware. We add up the multiplication results before reconstruction transform so the transform hardware can be shared among input channels.

B. Experiment and Analysis

To analyze how sharing affects the gate count, we implement convolution engines with different levels of parallelism. We adopt complex Winograd convolution in [12] for higher area reduction ratio and simple transformations which only involve additions and subtractions. Since the sharing of activation transform and reconstruction transform are independent, we set the same number of channels for input and output. The synthesis result compared to direct convolution is depicted in Fig. 3(b). Divide by gate count of direct convolution with the corresponding throughput, shared Winograd convolution exhibits more reduction for higher parallelism. The activation transform unit does not scale with the number of output channels, thus its ratio is inversely proportional to output channel parallelism. Although reconstruction transform and accumulator slightly grow with the number of input channels, the overall proportion of their gates decreases drastically. The analysis shows that sharing can bring down the transform overhead effectively, yet the resource-saving is still bounded by the multipliers. Hence we go on to explore Winograd domain quantization.

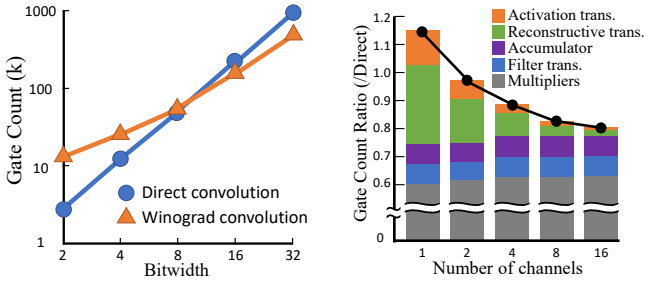

Fig. 3. Hardware performance for Winograd convolution. (a) Gate count without hardware sharing. (b) Gate count ratio for hardware sharing versus channel number.

III. Uneven Quantization

In this section, we propose an uneven quantization scheme to reduce the resource demand of Winograd domain multiplications. The transformations of Winograd convolution are linear combinations of spatial-domain tiles. The linear operations, consisting of additions and substractions, cause

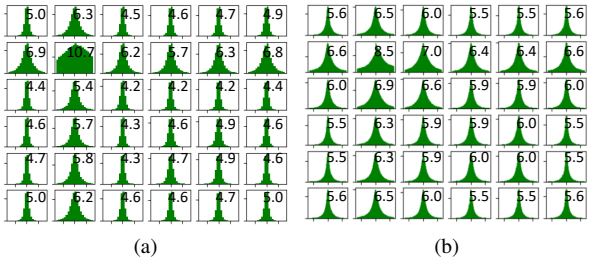

Fig. 4. Histogram and entropies of transformed activation tiles of corresponding positions in EDSR×4 model. (a) Worst case layer. (b) Overall.

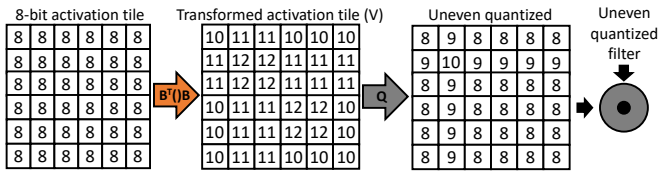

Fig. 5. Uneven Winograd-domain quantization for 8-bit inputs. The presented numbers indicate the bitwidths for the corresponding positions, and the same bitwidth setting is applied to Winograd-domain filter quantization.

TABLE I
PSNR PERFORMANCE ON MODIFIED FFDNET AND EDSR BASELINES

Quantization Schemes	PSNR / PSNR Drop to 8-bit Model (dB)			
	FFDNet	EDSR×2	EDSR×4	Average
Spatial 8-bit	31.21 / –	33.81 / –	28.56 / –	31.19 / –
Spatial 7-bit	31.02 / 0.19	33.66 / 0.14	28.45 / 0.11	31.05 / 0.14
Spatial 6-bit	30.86 / 0.36	32.63 / 1.18	28.11 / 0.45	30.57 / 0.62
Winograd 9-bit	30.66 / 0.56	33.74 / 0.07	**28.49 / 0.07**	30.96 / 0.23
Winograd 8-bit	29.82 / 1.39	33.60 / 0.21	28.39 / 0.17	30.60 / 0.59
Winograd uneven	**31.05 / 0.16**	**33.75 / 0.06**	**28.49 / 0.07**	**31.10 / 0.09**

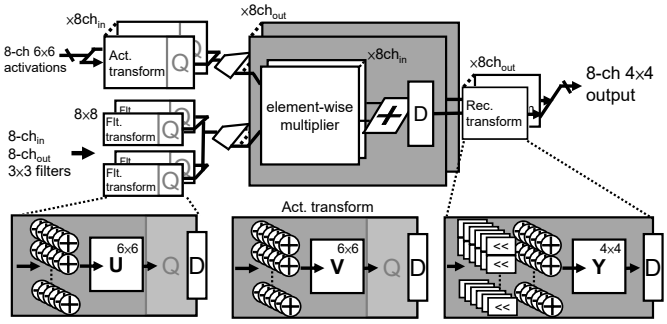

Fig. 6. Architecture of the area-efficient highly-parallel convolution engine.

dynamic-range expansion on the input values. As the input bitwidth gets lower, these enlarged Winograd-domain multipliers become much bigger than the multipliers in the spatial domain and damage the benefit of using fewer multiplications. As shown in Fig. 3(b), these multipliers take up 61% of gate counts of direct convolution even though the number of multiplications is cut down to 39%. Therefore, we further quantize the intermediate values in Winograd domain for a better reduction ratio.

Most of the work on Winograd-domain quantization uses floating-point sources and does not take advantage of low storage of fixed-point models. With the tile size expansion during filter transform, the memory demand for storing quantized data in Winograd domain is 2–4× of that quantized and saved in the spatial domain. L. Meng and J. Brothers [12] apply even 9-bit Winograd-domain quantization to 8-bit fixed-point classification models. However, the even-bitwidth quantization induces significant PSNR drops on the computational imaging CNNs. J. Fernandez-Marques et al. [13] use learning to modify the transformations and save the quality loss on fixed-point models, but the coefficients can no longer be realized with merely additions and subtractions.

We propose an uneven quantization in Winograd domain to keep better qualities on computational imaging models. Transformations in Winograd convolution extract the linear combination of input tiles, and the property of high correlation of nearby pixels in natural images has been verified to be preserved throughout layers in image-related CNNs. As a result, feature maps tend to form similar patterns in Winograd domain.

Statistic result in Fig. 4 shows that some positions of Winograd-domain activations have entropies of about 1–6 bits higher than the others. These values are more vulnerable to severe quantization, and we can effectively increase the output quality by assigning extra bits to them. Fig. 5 demonstrates the bitwidth overhead and the proposed uneven quantization scheme. The unevenly quantized values yield higher output qualities and require fewer logic gates for the element-wise multiplication than the even 9-bit ones.

IV. EXPERIMENTAL AND IMPLEMENTATION RESULTS

In the following, we will first demonstrate our evaluation results of the proposed quantization scheme on classic computational imaging CNN models. Then we show the synthesis result of our highly-parallel Winograd convolution engine. Finally, we compare the proposed convolution engine with other Winograd convolution implementations.

A. Quality Evaluation on Classic Networks

We verify the uneven quantization with 8-bit models on denoising and super-resolution models. Using the quantization aware training method, we acquire fixed-point networks of different bitwidth. Then we apply even or uneven quantization in Winograd domain on the 8-bit models.

For the denoising model (σ=25), we use 96-channel FFDNet [1] with the BN layers removed and a skip connection added to improve the training stability as mentioned in [14]. The denoising models are tested with Set5, Set14 and CBSD68 datasets. For super-resolution, we choose EDSR [2] baseline models of ×2 and ×4 scales and test the models with Set5, Set14, BSD100 and Urban100 datasets. The PSNR results are summarized in Table I. The proposed uneven quantization, with higher area efficiency, demonstrates 0.4 dB higher PSNR on the modified FFDNet and similar quality on EDSR baselines compared to the even one. Compared to 7-bit fixed-point

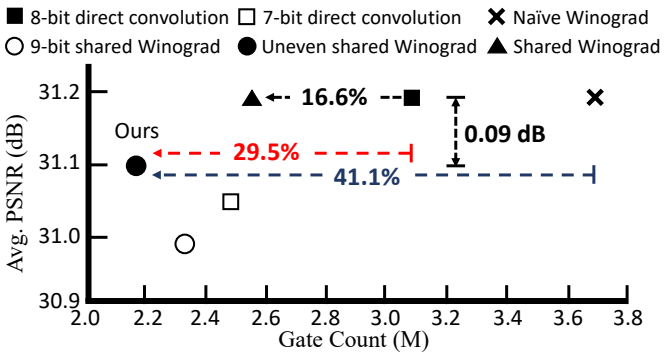

Fig. 7. Gate count and PSNR comparison of different implementations. Each design computes a 3×3 convolution of eight input and eight output channels for 4×4 output tile in one cycle.

models, the uneven quantization in Winograd domain causes less PSNR drop.

B. Synthesis Result of Highly-Parallel Convolution Engine.

We implement an eight-channel input and eight-channel output, 4×4 output tile complex Winograd convolution engine with uneven quantization in Winograd domain. Fig. 6 shows the architecture of our convolution engine. Synthesized with TSMC 40nm technology at 250 MHz, our engine uses 2.17M of logic gates to deliver 5.12 TOPS of inference capability, achieving Full HD 20 fps on the EDSR×4 model. For comparison, we also implement direct convolution and Winograd 9-bit quantized engines of the same clock rate and throughput. The synthetical results are summarized in Fig.7. Our engine saves 29.5% of logic gates compared to the direct convolution and 41.1% compared to the naïve Winograd convolution engine.

C. Comparison with Related Works

Table II shows different Winograd convolution implementations. We compare our engine with state-of-the-art Winograd convolution implementations on FPGA and ASIC. [10] proposed an optimized Winograd processing element (WinoPE) and maps Winograd convolutions on systolic arrays with WinoPEs on ZCU102 FPGA. [15] implemented a real-time super-resolution CNN accelerator that achieves a throughput of Full HD 60fps. Compared to [15], our engine achieves 20× of throughput with 439k fewer logic gates.

V. Conclusion

In this paper, we present an area-efficient convolution engine design. Firstly, We investigate the hardware-sharing and demonstrate the advantage of Winograd algorithm in highly-parallel convolution. Moreover, we propose an uneven quantization scheme in Winograd domain that saves 10.8% of logic gates and yields 0.11 dB higher PSNR on average on top of an even 9-bit quantization. Finally, our implementation of the highly-parallel Winograd convolution engine saves 29.5% and 41.1% of gate count compared to the 8-bit direct convolution engine and the naïve Winograd implementation respectively, with merely a 0.09 dB of PSNR drop.

TABLE II
Comparison of Winograd Convolution Implementations.

Related Works	[10]	[15]	Ours
Platform or Technology	ZCU102	SYNOPSYS 32nm EDK	TSMC 40nm
Frequency	214 MHz	200 MHz	250 MHz
Hardware resource	2345 DSP slices [1]	2048 mul.	3584 mul.
Model Precision	8–16 bit	12-bit	8-bit
Throughput	3.12 TOPS	0.26 TOPS	5.12 TOPS
Gate Count	-	2.61 M	2.17 M

[1] Each DSP slice can compute a 27×18 bit multiplcation.

References

[1] K. Zhang, W. Zuo, and L. Zhang, "FFDNet: Toward a fast and flexible solution for CNN-based image denoising," IEEE Transactions on Image Processing, vol. 27, 2018.

[2] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee,"Enhanced deep residual networks for single image super-resolution," in IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2017.

[3] C.-T. Huang, Y.-C. Ding, H.-C. Wang, C.-W. Weng, K.-P. Lin, L.-W. Wang, and L.-D. Chen, "eCNN: A block-based and highly-parallel CNN accelerator for edge inference," in IEEE/ACM International Symposium on Microarchitecture (MICRO), 2019.

[4] J. Lee, D. Shin, J. Lee, J. Lee, S. Kang, and H. J. Yoo, "A Full HD 60 fps CNN super resolution processor with selective caching based layer fusion for mobile devices." In IEEE Symposium on VLSI Circuits (VLSIC) , 2019.

[5] S. Winograd, Arithmetic complexity of computations. CBMS-NSF Regional Conference Series in Applied Mathematics. Society for Industrial and Applied Mathematics, 1980.

[6] A. Lavin and S. Gray, "Fast algorithms for convolutional neural networks," in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[7] J. Liu, D. Yang, and J. Lai, "Optimizing Winograd-based convolution with tensor cores," in International Conference on Parallel Processing (ICPP), 2021.

[8] A. Xygkis, D. Soudris, L. Papadopoulos, S. Yous and D. Moloney, "Efficient Winograd-based convolution kernel implementation on edge devices," in ACM/ESDA/IEEE Design Automation Conference (DAC), 2018.

[9] A. Podili, C. Zhang and V. Prasanna, "Fast and efficient implementation of convolutional neural networks on FPGA," in IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP), 2017.

[10] X. Liu, Y. Chen, C. Hao, A. Dhar and D. Chen, "WinoCNN: Kernel sharing Winograd systolic array for efficient convolutional neural network acceleration on FPGAs," in IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP), 2021.

[11] L. Lu, and Y. Liang, "SpWA: An efficient sparse Winograd convolutional neural networks accelerator on FPGAs," in ACM/ESDA/IEEE Design Automation Conference (DAC), 2018.

[12] L. Meng and J. Brothers, "Efficient Winograd convolution via integer arithmetic," arXiv:1901.01965, 2019.

[13] J. Fernandez-Marques, P. N. Whatmough, A. Mundy, and M. Mattina, "Searching for Winograd-aware quantized networks," arXiv:2002.10711, 2020.

[14] C.-T. Huang, "ERNet family: Hardware-oriented CNN models for computational imaging using block-based inference," in IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2020.

[15] P.-W. Yen, Y.-S. Lin, C.-Y. Chang, and S.-Y. Chien, "Real-time super resolution CNN accelerator with constant kernel size Winograd convolution," in IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS), 2020.