2023 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia) | 979-8-3503-4431-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICCE-Asia59966.2023.10326358

An Energy-Efficient GAN Processor for Mobile Image Translation

Jinhoon Jo[1], Sangho Lee[2], Ghangmin Yun[2] and Kyuho Lee[1,2]
Graduate School of Artificial Intelligence[1] / Department of Electrical Engineering[2]
Ulsan National Institute of Science and Technology (UNIST), Ulsan, Republic of Korea
jinhoonjo@unist.ac.kr / kyuho.jsn.lee@unist.ac.kr

Abstract

An energy-efficient generative adversarial network (GAN) accelerator is proposed for mobile image-to-image translation. In image translation, low precision bits below 8 were not employed due to significant output image quality degradation. In addition, due to the zero injection of transposed convolution, effective PE utilization decreased up to 89%. To address these problems, this paper proposes two key features: 1) we apply layer-wise dynamic fixed-point quantization and implement bit-combined PE to increase throughput by 2×; 2) By analyzing the matching pattern of transposed convolution, data remapping for transposed convolution that simultaneously computes 4 outputs is proposed. The proposed processor is implemented on ZCU 104 FPGA, achieving energy efficiency of 76.38 GOPS/W while consuming 6.08 W of power for mobile image-to-image translation.

Keywords: generative adversarial network (GAN), Image-to-image translation, FPGA

1. Introduction

Recently, generative adversarial networks (GANs) have been widely adopted for various applications of image-to-image translation such as style transfer [1-2], semantic segmentation [3-4], and data augmentation [5]. As shown in the fig. 1, GAN comprises 2 sub-networks: a generator and a discriminator. The discriminator is trained to classify the fake (generated) from the real (input images), while the generator is trained to disguise the discriminator by generating fake images similar to real input. By training two subnetworks to contest against each other, GAN can generate high-quality and realistic images. The acceleration of GAN on mobile devices can expand the applicability of AI into various fields.

However, inferencing GAN involves massive multiply-and-accumulation (MAC) operations and external memory access (EMA) to generate the high-resolution output image. Moreover, it requires high bit precision for weight and activation, due to the

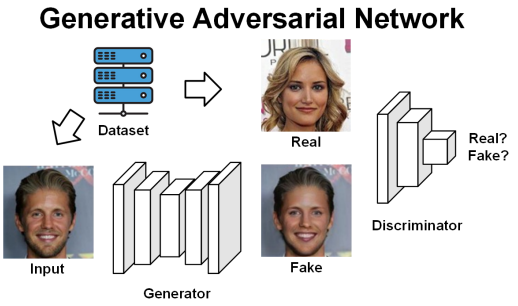

Fig. 1. Image-to-image translation with generative adversarial network

Fig. 2. Difference in generated image quality based on the datatype

performance degradation. As shown in fig. 2, the image generated with lower bit precision than 8-bit exhibits low quality. Thus, the previous researches adopt at least 16-bit fixed-point (FXP) [6-7] or 8-bit floating-point (FP) [8], increasing the power consumption of inferencing GAN. Additionally, the generator includes the standard convolution (Conv) layers with stride 2 and Transposed Convolution (TConv) layers to generate a high-resolution output image from internal low-resolution features. Without a dedicated computation unit, redundant EMA and degradation of PE utilization is inevitable. Since mobile devices should meet strict power restrictions and have limited computation resources, energy-efficient implementation of the GAN accelerator is crucial.

To solve the abovementioned challenges of inferencing GAN, this paper proposes the Hardware-algorithm co-optimization to achieve high energy efficiency for mobile implementation. Layer-wise dynamic FXP quantization with Bit-Combined PE architecture increases the resource efficiency (GOPs/DSP). Moreover, the proposed activation shared register and dual-mode aggregation engine

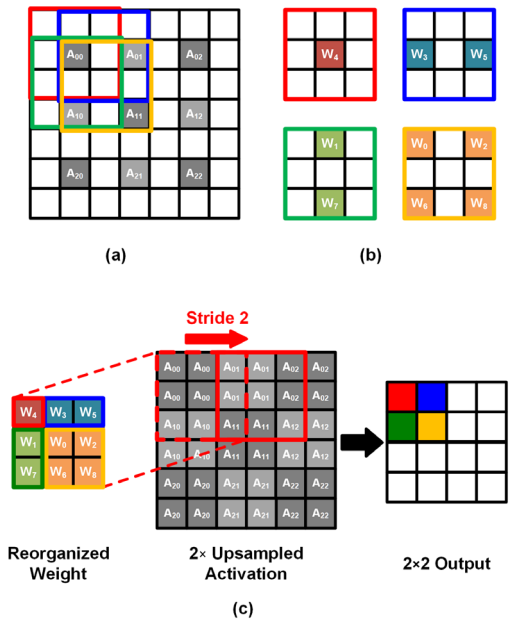

Fig. 3. Proposed data remapping for TConv

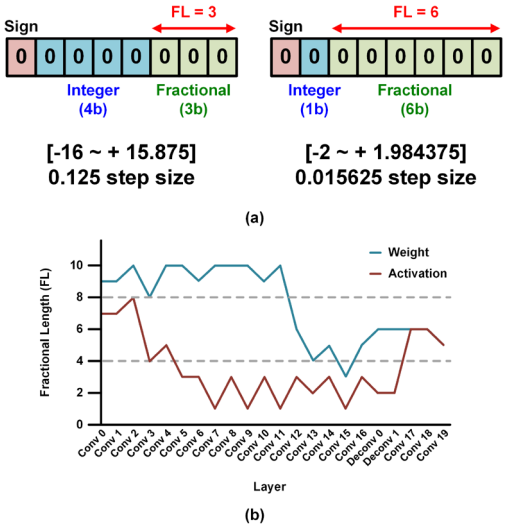

Fig. 4. (a) Explanation of dynamic FXP quantization and (b) Layer-wise fractional lengths of StarGAN

support the dedicated data mapping for TConv and stride 2 Conv, increasing the throughput and reducing the redundant EMA caused by meaningless zero computation. We implemented the proposed GAN accelerator on Xilinx's ZCU 104 FPGA designed for surveillance and drones.

The rest of this paper is organized as follows. Section II introduces the proposed algorithm optimization process, including data remapping for TConv and dynamic FXP quantization. Section III describes the proposed hardware and its operation. Implementation results and conclusion will be followed in Section IV and V, respectively.

2. Data remapping for TConv and layer-wise dynamic FXP quantization

Fig. 3 shows the proposed data remapping for TConv (DRT) which alleviates redundant computation caused by injected zero input during TConv operation. The 4 patterns that occur in the 3×3 TConv operation are shown in Fig. 3(a). As highlighted in the 4 colors of matching patterns in Fig. 3(b), which are repeated every 2 rows and columns, they cause unnecessary zero multiplications and decrease effective PE utilization by 89%.

Since the 4 patterns are associated with different weight values, it is possible to reorganize patterns that correspond to the same activation. Therefore, as depicted in Fig. 3(c), upsampling the activation by a factor of 2 and sliding the reorganized weights with a stride of 2, make it possible to obtain four sets of TConv results, simultaneously. Thanks to DRT, not only ineffective MAC operation caused by injected zero, but also loading meaningless zero data can be skipped. By managing the dataflow, EMA caused by TConv is reduced by 50.6% and 4× higher throughput is achieved by getting 4 output activation in a single convolution process.

To efficiently utilize the limited resources of FPGA, the layer-wise dynamic FXP quantization (LDX-Quant) with 8-bit precision is applied to weight and activation data of StarGAN [1], target network of this work. Dynamic FXP representation is commonly used in DNN to support variable bit-precision. Fig. 4(a) describes the representation of dynamic FXP. The fractional length (FL) in the dynamic FXP can determine the location of decimal points and quantization steps. Since each layer in StarGAN has its own ranges of weight and activation, applying LDX-Quant enables the network to maximize peak signal-to-noise ratio (PSNR) when compared to an unquantized network. Fig. 4(b) shows the decided FL of each layer to achieve maximum PSNR, 33.95 dB, which means the result shows almost similar quality compared to the results of original network. The optimization is conducted using the randomly sampled images from the CelebA dataset.

3. Hardware architecture

Fig. 5 illustrates the overall architecture of the proposed accelerator. It consists of 8 dual-mode bit-combined convolution cores (DB-Conv Cores), an activation shared register (ASR), a top controller, and on-chip memories. All on-chip memories and the top controller are connected to external memory with a 32-bit AXI bus. An IMEM is composed of double buffers, thus, the data from external memory is loaded to a single buffer while the other produces the data to DB-Conv Cores, effectively hiding EMA latency and increasing the system throughput.

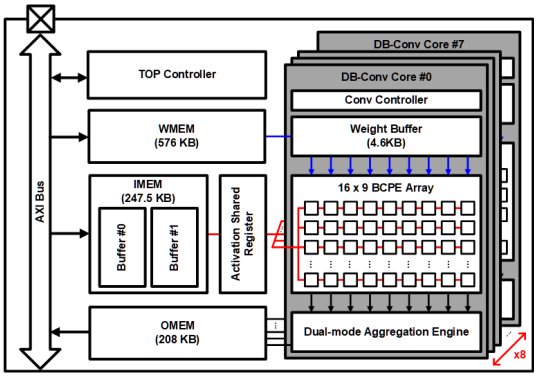

Fig. 5. Overall architecture of the proposed GAN accelerator

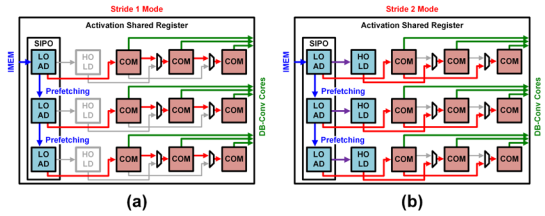

Fig. 6. Data path for (a) stride 1 and (b) stride 2 Convolution in ASR

Fig. 6 shows the architecture and operation modes of ASR suggested to support the various stride types of convolutions. Shift-register-based ASR consists of three registers for perfecting the data from IMEM, threes for holding the data for stride 2 Conv operation, and nines for producing input data to DB-Conv Core. From the IMEM, 1×1 size of inputs along 256 input channels are fetched and ASR broadcasts the 3×3 input feature map to 8 DB-Conv Cores.

For Conv operation with stride 1, three input activations are loaded from IMEM for prefetching during 3 cycles. After the prefetch of input data, the data waits for CB-Conv Core to be done and then propagated to the right registers. The propagation of data to the right column involves a stride 1 sliding, and the DB-Conv Core initiates computations using the propagated data in ASR.

In the case of stride 2 convolution, data is loaded into the prefetch register in the same way as in the stride 1 convolution operation. However, after the prefetch, the data is not shifted to registers for processing, but for holding. Then, the prefetched and held data is directly propagated next to the nearby register. Since, operation of DB-Conv Core takes 16 cycles, preparation of data for operation can be done without stall of DB-Conv Core operation.

DB-Conv Core has a 16×9 Bit-combined PE Array (BCPEA), a weight buffer which holds two weight kernels and a dual-mode aggregation engine (DAE), and a Conv controller.

Fig. 7 shows the schematic of the proposed bit-combined PE (BCPE). Each BCPE consists of a DSP, a shift register for bit-combination, and an overflow estimator for error detection and correction. In an

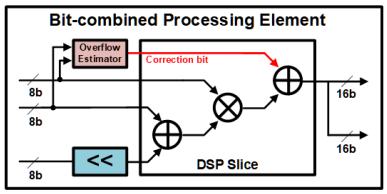

Fig. 7. The hardware structure of the BCPE

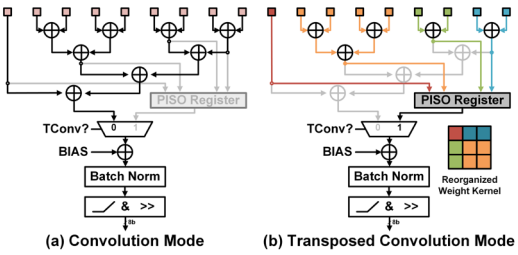

Fig. 8. Schematic and different operation of DAE

embedded DSP module in FPGA, circuits for 25-bit addition, 26-bit and 18-bit multiplication, and 48-bit accumulation are built in. A 25-bit input port of BCPE is utilized for multiplication of two FXP8 weights and activation, shown in equation (1).

$$(W_0 \times 2^{17} + W_1) \times A = AW_0 \times 2^{17} + AW_1 \quad (1)$$

A multiplication result of two 8-bit data is represented in 16-bit, which means that two multiplications can be operated simultaneously by shifting an 8-bit data over 16 bits. However, when the sign of the LSB multiplier and multiplicand (A and W_1) are different, an overflow occurs due to the negative sign bit of the result, and it affects the result in MSB multiplication result (AW_0). Overflow estimator circuit detects the sign bit for multiplier and multiplicand in LSB, and inserts the correction bit for obtaining two multiplication results without any error.

Computation results of BCPEA are accumulated along input channel direction, and left operations are progressed in DAE. The different operations of DAE for Conv, and TConv are explained in detail in Fig. 8. DAE determines different accumulation paths for different types of convolution operation, and progress bias addition, batch normalization, ReLU activation, and descaling according to the FL of the next layer.

In case of Conv operation, single output activation is accumulated and post processing is progressed. On the other hand, in operating TConv, the accumulation path is separated and 4 output activations are accumulated simultaneously. Thus, a parallel-in serial-out register temporarily stores data, and outputs it sequentially.

Thanks to stride 2 mode operation in ASR and TConv mode operation in DAE, redundant zero skipping of DRT can be implemented on our system. Input data for TConv can be upscaled in the prefetch process in ASR, which increases throughput of the system by 4× without latency for complex data

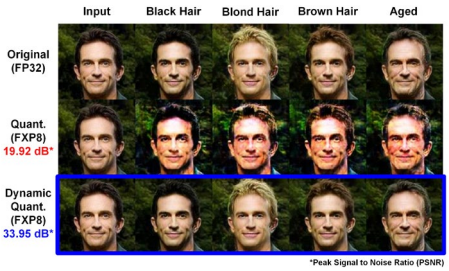

Fig. 9. Result of LDX-Quant and comparison with conventional Quant result

Table 1: Performance summary

Frequency	100MHz	
Power	6.08 W (2.74 W for system)	
Peak Performance	398.8 GOPS (Conv) 1,595 GOPS (T-Conv)	
Energy Efficiency	65.59 GOPS/W (Conv) 262.34 GOPS/W (T-Conv)	
Resource Usage	Available	Usage
LUT	230,400	193,835 (84.13%)
Block Memory	312	261 (83.65%)
DSP	1,728	1,188 (68.8%)
Resource Efficiency	0.336 GOPS/DSP (Conv) 1.343 GOPS/DSP (T-Conv)	

alignment or EMA for loading overlapped data. In addition, partial sums are only accumulated in DB-Conv Core, so descaled FXP8 final output results are stored in OMEM, which can downsize 71.4% of memory size when storing partial sums.

4. Implementation results

Fig. 9 shows the results of inferencing StarGAN with FXP8 quantization and LDX-Quant. The performance degradation of FXP quantization with 8-bit precision results is noticeable, making style transfer tasks are meaningless. However, LDX-Quant results have almost negligible performance degradation compared to the unquantized results.

Table 1 summaries specifications of the accelerator. Proposed accelerator is implemented on Xilinx ZCU104. Under the 100MHz frequency, it consumes 6.08W power and achieves 398.8 GOPS and 1595 GOPS of Conv and TConv peak performance, respectively. Table 2 shows comparisons with previous FPGA GAN accelerators. By employing BCPE and DRT, the proposed accelerator is the most energy and resource efficient FPGA-based GAN accelerator with 76.38 GOPS/W, which is 1.93 x improvement over [6].

5. Conclusion

An energy-efficient GAN processor for mobile image translation is proposed. The accelerator employs FXP8 operations through layer-wise dynamic fixed-point quantization and adopt bit-combined PE, resulting in a 2× increase in throughput. Moreover, thanks to the proposal of data remapping

Table 2: Performance comparison

	ICFPT 2019 [6]	ASSCC 2020 [9]	This Work
Processor	Xilinx ZCU102	Intel Cyclone V	Xilinx ZCU104
Bit Precision	16 fixed	32 float	8 fixed
Model	Disco-GAN	MaskGAN	StarGAN
Power [W]	15.6	5.1	6.08
DSP	2,520	160	1,188
Average Performance [GOPS]	616.1	44.77*	464.4
Energy Efficiency [GOPS/W]	39.49	8.78*	76.38
Resource Efficiency [GOPS/DSP]	0.244	0.280*	0.391

* FLOPS

for transposed convolution, throughput of transposed convolution increase 4×. As a result, the proposed GAN processor achieves the peak performance of 1595 GOPS on transposed convolution while 6.08W power consumption. As a result, it achieves energy efficiency of 76.38 GOPS/W for mobile image-to-image translation.

6. Acknowledgment

This work was supported in part by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2020-0-01336, Artificial Intelligence Graduate School Program (UNIST)) and in part by the U-K BRAND Research Fund (1.230019.01) of UNIST (Ulsan National Institute of Science & Technology)'

References

[1] Choi, Y. et. al., (2018). Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In CVPR (pp. 8789-8797).
[2] Karras, T et. al., (2019). A style-based generator architecture for generative adversarial networks. In CVPR (pp. 4401-4410).
[3] Isola, P. et. al., (2017). Image-to-image translation with conditional adversarial networks. In CVPR (pp. 1125-1134).
[4] Mondal, A. K. et. al., (2019). Revisiting CycleGAN for semi-supervised segmentation. arXiv preprint arXiv:1908.11569.
[5] Hammami, M. et. al., (2020). Cycle GAN-based data augmentation for multi-organ detection in CT images via Yolo. ICIP (pp. 390-393). IEEE.
[6] X. Di et. al., "Exploring Resource-Efficient Acceleration Algorithm for Transposed Convolution of GANs on FPGA," 2019 ICFPT, pp. 19-27.
[7] A. Yazdanbakhsh et al., "FlexiGAN: An End-to-End Solution for FPGA Acceleration of Generative Adversarial Networks," 2018 FCCM, pp. 65-72.
[8] S. Kang et al., "7.4 GANPU: A 135TFLOPS/W Multi-DNN Training Processor for GANs with Speculative Dual-Sparsity Exploitation," 2020 ISSCC, pp. 140-142.
[9] S. Kim, et al., "An Energy-Efficient GAN Accelerator with On-chip Training for Domain Specific Optimization," 2020 A-SSCC, pp. 1-4.