2021 31st International Conference on Field-Programmable Logic and Applications (FPL) | 978-1-6654-3759-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/FPL53798.2021.00014

MP-OPU: A Mixed Precision FPGA-based Overlay Processor for Convolutional Neural Networks

Chen Wu
Electrical and Computer Engineering
University of California, Los Angeles
Los Angeles, USA
chenwu1989@g.ucla.edu

Jinming Zhuang
School of Electronic Science and Engineering
University of Electronic Science and Technology of China
Chengdu, China
zhuangjinming@std.uestc.edu.cn

Kun Wang*
Electrical and Computer Engineering
University of California, Los Angeles
Los Angeles, USA
wangk@ucla.edu

Lei He*
Electrical and Computer Engineering
University of California, Los Angeles
Los Angeles, USA
lhe@ee.ucla.edu

***Abstract*—Low precision quantization in convolutional neural network (CNN) inference has been proved effective for reducing computation complexity and bandwidth requirement. Mixed precision CNNs manage to benefit from low precision while maintaining accuracy. In this paper, we propose a Mixed Precision FPGA-based Overlay Processor (MP-OPU) to fully leverage the advantages of mixed precision for both conventional and lightweight CNNs. The micro-architecture of MP-OPU considers sharing of computation core with mixed precision weights and activations to improve computation efficiency. In addition, run-time scheduling of external memory access and data arrangement are optimized to further leverage the advantages of mixed precision data representation. Our experimental results show that MP-OPU reaches 4.92 TOPS peak throughput when implemented on the Xilinx VC709 FPGA (with all DSPs configured to support 2-bit multipliers). Moreover, MP-OPU achieves 12.9× latency reduction and 2.2× better throughput/DSP for conventional CNNs while 7.6× latency reduction and 2.9× better throughput/DSP for lightweight CNNs, all on average compared with existing FPGA accelerators/processors, respectively. To the best of our knowledge, this is the first in-depth study on mixed precision FPGA-based overlay processor for both conventional and lightweight CNNs.**

***Index Terms*—Mixed precision, CNN, FPGA, Hardware Acceleration**

I. Introduction

Recent deep convolutional neural networks (CNNs) are widely used in real-time applications, such as autonomous driving, where model size and inference latency are the main constraints. Many researchers have introduced low-precision quantization techniques to reduce the model size, computation complexity and communication bandwidth so that the inference latency is reduced [1].

However, conventional quantization approaches take the same bit width of weights and activations for all layers, and 8-bit fixed point has been proved effective in hardware implementation while maintaining accuracy [2], [3]. Lower precision networks are also possible to achieve high accuracy, but require expert design and re-training [4]. To this end, mixed precision comes to be a significant solution that assigns different precision for different layers and different networks [5]. However, the variety of the precision of each layer makes the architecture design more challenging.

Trying to overcome the challenge, Nvidia releases Turing GPU architecture which supports mixed precision arithmetic operations (1-bit, 4-bit, 8-bit and 16-bit) [6]. BitFusion [7] is another accelerator which can support multiplication on 2, 4, 8 and 16 bits. However, these architectures cannot fully leverage the advantages of the quantized model. For example, when the network is quantized to be 6-bit, the model has to be modified to 4-bit or to 8-bit when mapped on these architectures, which either leads to accuracy reduction or latency increment. On the other hand, bit-serial multipliers, which can support more flexible mixed precision multiplications, are used in Bismo [8]. However, Bismo uses large amount of look-up-tables (LUTs) and Block RAMs (BRAMs) to implement matrix multiplication on the FPGA, which makes it difficult to support CNNs.

To this end, we propose the Mixed Precision FPGA-based Overlay Processor (called **MP-OPU**) that effectively accelerates the inference of mixed precision models. By using the similar instructions and compilation flow as Light-OPU [9], we redesign the computation core and memory system to expand the support from 8-bit fixed point CNNs to 2-bit to 8-bit mixed precision CNNs. More specifically, the computation core can be configured to operate with different number of multipliers according to the given bit width of activations and weights, while the memory system in ping-pong architecture is capable of run-time data rearrangement to fully utilize the bandwidth of external memory.

The main contributions of the proposed **MP-OPU** can be summarized as follows:

- **High Flexibility.** Different from the previous work that

*Corresponding authors: Kun Wang (wangk@ucla.edu) and Lei He (lhe@ee.ucla.edu).

978-1-6654-3759-2/21/$31.00 ©2021 IEEE
DOI 10.1109/FPL53798.2021.00014

only support a subset of the the possible precision choices [10], our **MP-OPU** manages to address mixed precision CNNs varying from 2-bit to 8-bit. In our design, the computation core is developed to be programmable during run-time to support mixed precision efficiently. By setting different parameter registers in the instructions, **MP-OPU** can support mixed precision without re-implementing the design. Moreover, the memory system is designed with run-time data pre-fetch and placement modules to optimize the external memory access for mixed precision.

- **High Scalability.** Our implementation is highly scalable as it can be easily scaled up or down to different FPGAs by adding or removing PEs. We only implement batch parallelism among different PEs so that the PEs are separate to each other and can be simply added or removed.
- **High Performance.** With all the DSPs configured to support 2-bit multiplication, **MP-OPU** can reach 4.92 TOPS peak throughput on Xilinx VC709 evaluation board. Furthermore, we take conventional and lightweight CNNs as benchmarks to be tested on **MP-OPU**. All these networks are accelerated and **MP-OPU** achieves 12.9× latency reduction and 2.2× better throughput/DSP for conventional CNNs, while 7.6× latency reduction and 2.9× better throughput/DSP for lightweight CNNs on average compared with existing FPGA accelerators, respectively.

II. Background and Related Work

A. Low Precision Quantization

Extensive explorations have been made on compressing and accelerating neural networks by using quantization. Deep compression methods [1] quantize the network weights to reduce the model size by rule-based strategies. More aggressive quantization strategies use 1-bit or 2-bit to represent the weights [11]. Neural architecture search (NAS) based mixed precision quantization is also proposed to improve the performance and efficiency of quantizing a network. FBNet [12] builds a lookup table of latency for different operations running on the hardware, and optimizes the latency during the design process. HAQ [5] computes the hardware feedback directly from two customized accelerators and optimizes the quantization policy until the resource constraints (*e.g.,* latency, energy) are met. All these studies manage to quantize the full precision network (32-bit) into low precision (less than or equal to 8-bit) with negligible accuracy loss. Therefore, we utilize the quantization results from these methods and the quantization approach is not included in the scope of this work.

B. Fixed Precision Processors/Accelerators

Customized processors/accelerators on FPGA are proposed to accelerate the inference of the quantized networks. OPU [2] accelerate 8-bit fixed point CNNs, while the work in [3] addresses 8-bit floating point CNNs. Light-OPU [9] expands the work from conventional CNNs to light-weighted CNNs with 8-bit fixed point as well. Different from these studies, our work focuses on mixed precision CNNs.

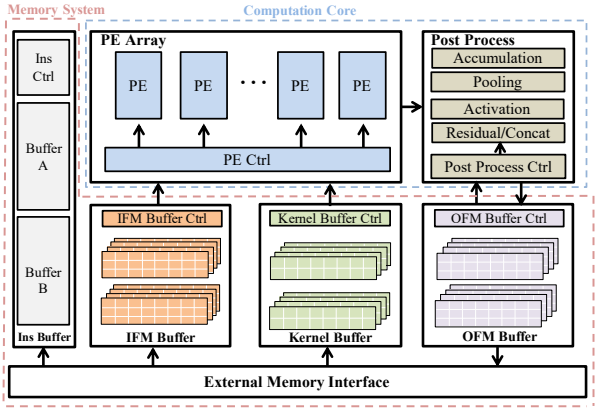

Fig. 1: The micro-architecture of **MP-OPU**. The computation core has multiple PEs for convolutional layers followed by a post process module for other types of layers. The memory system is designed with on-chip buffers in ping-pong manner to save communication time with the external memory.

C. Mixed Precision Processors/Accelerators

Mixed precision processors/accelerators on FPGA have also been also proposed recently. Power-of-2 and fixed point data are supported by the auto-generated accelerators on FPGA [13]. The approach in [10] keeps the activations to be 8-bit fixed point while the weights vary from 1 to 16-bit. However, they only support a limited part of the bit width combinations, while **MP-OPU** is optimized for all the combinations of low precision (2 to 8 bits).

III. Micro-Architecture of MP-OPU

The micro-architecture of **MP-OPU** discussed in this section is shown in Fig. 1, consisting of *Computation Core and Memory System*. In the *Computation Core*, the *PE Array and Post Process* modules perform the computation for all the conventional convolution, depth-wise convolution, fully-connected, pooling, activation, residual and concatenation layers. In the *Memory System*, we have 4 on-chip buffers, which are designed in ping-pong manner, to perform data communication with the external memory. Moreover, in each module, we have a separate control module to update the parameter registers defined in instructions, as well as to control the data flow in these modules.

A. Computation Core

In the *Computation Core*, the *PE Array* finalizes the convolution of one layer block, while the *Post Process* handles the accumulation of the intermediate results of different layer blocks and other types of layers, such as pooling, residual and concatenation layers. To fully leverage the advantages of mixed precision CNNs, both the *PE Array* and the *Post*

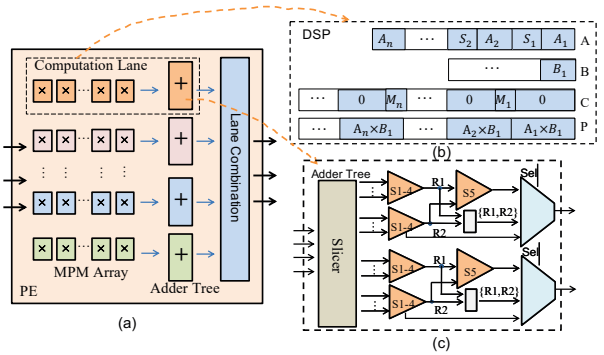

Fig. 2: The architecture of one PE.

Process modules are designed to support mixed precision operations and can be programmed by the parameter registers defined in the instructions.

1) Architecture of PE: As shown in Fig. 2(a), each PE has multiple computation lanes to perform computation in parallel, and one lane combination module is followed to combine or select the results according to different parallelism modes. Each computation lane is composed of a mixed precision multiplier (MPM) array and an adder tree. In order to fully utilize the resources on FPGA, we use the DSP slices to implement the mixed precision multipliers, and we decompose one DSP slice into several low precision multipliers to increase the resource usage efficiency. As the weights and activations are always quantized by different bit width in a CNN, we propose the decomposition rule to support different bit width of multiplier and multiplicand, as shown in Fig. 2(b). We share the same multiplier B_1 and use n multiplicands (A_1, A_2 to A_n) to compact n multipliers into one DSP. In our implementation, we configure each DSP to perform as a multiply-adder ($P = A \times B + C$), and fit different operands (A, B and C) to the multiply-adder according to different precision combinations. As the signed multiplication of lower significant bits may impact the higher significant bits, we introduce a modification bit for each low precision multiplier. We explore all the combinations of A_i and B_1 in an exhaustive way and conclude that the ith modification bit follows:

$$M_i = \begin{cases} Sign(B_1) \oplus Sign(A_i) & B_1 \neq 0 \text{ and } A_i \neq 0 \\ 0 & \text{otherwise} \end{cases}, \quad (1)$$

where $Sign(x)$ indicates the sign of the data x and $\oplus$ means xor. In this way, we just need to slice P into n parts to have the product of each low precision multiplier.

For each computation lane in **MP-OPU**, the number of DSPs is set to be fixed. Since each DSP can be decomposed to multiple low precision multipliers, the number of outputs for the MPM array varies according to the number of multipliers. Therefore, the following adder tree is also designed to be programmable by the parameter registers, as shown in Fig. 2(c). In order to use unified bit width for the adders in the adder tree, we first do slicing and sign extension for each input in the slicer. The aligned data are then fed into the 4-stage adder tree. Afterwards, the sum of two 4-stage adder trees are either summed up or concatenated according to the computation mode. Moreover, some stages of the adder tree can also be bypassed in order to support depth-wise convolution.

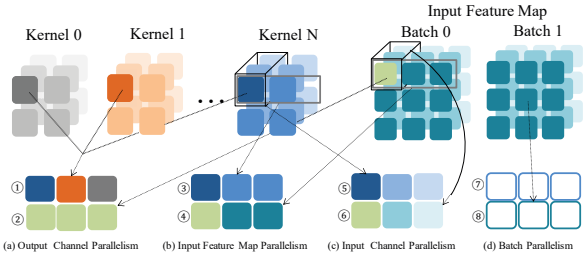

Fig. 3: Four levels of parallelism in **MP-OPU**.

2) Parallelism Exploration: In order to speed up the calculation of the CNN, we use four levels of parallelism in the **MP-OPU**, as shown in Fig. 3. For the output channel parallelism, we use one activation to multiply different kernels to generate different activations of different output channels, as shown in Fig. 3(a). In the input feature map parallelism (shown in Fig. 3(b)), we will compute the convolution of different activations in the same input feature map with the corresponding kernels to produce one activation of one output feature map. For the input channel parallelism (shown in Fig. 3(c)), multiplication of activations in different input channels and corresponding kernels are computed in parallel and then summed up to form a result (or intermediate result) of one output channel. The batch parallelism will calculate activations from different input images in parallel. These four levels of parallelism are selected according to different types and parameters of the layers. For example, for the conventional convolutional layer, we will use input channel parallelism in each computation lane, and use output channel parallelism among different computation lanes in each PE. Since each output feature map channel is generated by convolving one kernel and one input feature map channel in the depth-wise convolutional layer, we use the input feature map parallelism in each computation lane and output channel parallelism among different computation lanes in each PE. To simplify the design, we set all the PEs to be independent and only do batch parallelism among different PEs.

B. Data Pre-fetch and Placement for on-chip Memory

Since the maximal bandwidth of the external memory is fixed, we use ping-pong architecture and data pre-fetch module to increase the run-time bandwidth of the external memory. With ping-pong architecture, the communication time can be hid under the computation time. The data pre-fetch module manages to load as many data with different precision as possible. As different parallelism levels require activations and weights to be placed in different orders, we arrange and store the data accordingly in the memory system in advance. On the

TABLE I: Comparison with customized FPGA accelerators/processors on conventional CNNs.

	OPU1024 [2]	[13]	[14]	**MP-OPU**	
Year	2019	2019	2020	**2021**	
Device	XC7K325T	2 XC7VX690T	XC7VX485T	**XC7VX690T**	
Network	VGG16	VGG16	Tiny-Yolo-V3	**VGG16**	**Tiny-Yolo-V3**
Bit width	8	mixed [1]	18	**mixed**	
Frequency (MHz)	200	156	200	**200**	
DSP Used	516	-	2304	**3072** [2]	
Inference latency (ms)	88.7	200.9	-	**11.2**	**2.91**
Throughput/DSP (GOPS)	0.68	-	0.2	**0.90**	**0.62**

[1] Weights are mixed precision while activations are 8-bit.
[2] 2048 DSPs are used for low precision multipliers while 1024 DSPs are used for adders.

TABLE II: Resource Utilization of **MP-OPU** on XC7VX690T.

Resource	LUT	LUTRAM	FF	BRAM	DSP
Used	278548	42853	324033	912	3072
Available	433200	174200	866400	1470	3600
Utilization	64.3%	24.6%	37.4%	62.0%	85.3%

other hand, the feature maps are arranged during run-time to fit different parallelism levels. As the output feature map of one layer will be the input of another layer, we add an extra data rearrangement logic to change the data arrangement between row major and channel major according to the parallelism levels. Only when the parallelism level changes between two adjacent layers, will the data rearrangement logic be enabled.

IV. Experiments

A. Experimental Setup

The proposed **MP-OPU** is implemented on the Xilinx VC709 evaluation board with an XC7VX690T FPGA. The design is described in Verilog-HDL, synthesized and implemented with Vivado 2020.1. For the network benchmarks, we use both conventional and lightweight CNNs for a comprehensive comparison to show the effectiveness of **MP-OPU**. These CNNs cover different kernel sizes (1×1 and 3×3), strides (1×1 and 2×2), and convolutional layer types (conventional convolution and depth-wise convolution). In addition, irregular layer operations such as residual and concatenation are also included.

B. Hardware Implementation

In this work, 8 PEs are implemented. To balance the usage of DSPs and LUTs, we use 256 DSPs to implement mixed precision multipliers and 128 DSPs to implement mixed precision adders in each PE. All the 256 DSPs for multipliers are configured by the same parameter registers to support multipliers with bit width varying from 2 to 8 bits. The processor is designed to meet 200MHz timing constraints and the detailed resource utilization is listed in Table II.

C. Comparison with Customized FPGA Accelerators

We further compare **MP-OPU** with 5 FPGA accelerators/processors on both conventional and lightweight CNNs, and the results are shown in Table I and Table III. As the quantization method is not included in this paper, we use the bit width generated by HAQ [5]. We use the **inference latency** to evaluate the performance of running the network on each FPGA processor/accelerator. **Throughput/DSP**, which is defined as the throughput conducted by each DSP during run-time, is utilized to indicate the efficiency of each design. The image size is $416 \times 416 \times 3$ for Tiny-Yolo-V3 and $224 \times 224 \times 3$ for the other networks.

As shown in Table I, **MP-OPU** achieves $7.9\times$ and $17.9\times$ inference latency reduction on VGG16 compared with OPU1024 and the accelerator in [13], respectively. In addition, **MP-OPU** outperforms OPU1024 on throughput/DSP by $1.3\times$. As for Tiny-Yolo-V3, we have $3.1\times$ better throughput/DSP than the prior accelerator. The approach in [13] does not report the DSP utilization, but they use two XC7VX690T FPGAs while we only use one. For MobileNet-V1/V2, **MP-OPU** performs $6.0\times$ and $10.1\times$ reduction on latency on average compared with existing works, respectively. Although we use more DSPs than others, we still have $2.4\times$ and $3.5\times$ better throughput/DSP on average, respectively. Better throughput/DSP indicates higher computation efficiency during run-time, and this is one main reason why we have lower inference latency.

D. Discussion

To further demonstrate the effectiveness of **MP-OPU**, we evaluate the inference latency with respect to different bit width combinations. We take Tiny-Yolo-V3 as an example and the results are shown in Fig. 4. In general, the latency increases as the bit width increases. The observation follows the rule that one DSP can be decomposed to more multipliers when the target bit width is small. However, the latency remains unchanged when the bit width changes from 7-bit to 8-bit because one DSP can only be decomposed into two multipliers for both 7-bit and 8-bit. Furthermore, compared with the 8-bit model, the inference latency of the 2-bit model reduces to about $3.9\times$, although one DSP can only be decomposed to $3\times$ as many 2×2 multipliers as 8×8 multipliers. The extra benefits come from the data pre-fetch module, where we manage to fetch $4\times$ as much 2-bit data as 8-bit data under the same external memory bandwidth. For the 8-bit case, the bandwidth constraints is still severe as we cannot hide all the external communication time under the computation time, especially for the layers with 1×1 kernel size. Therefore, the percentage of the DSPs being effective during run-time is less than 100%. The problem of bandwidth constraints can be alleviated when the activations and weights decreased to 2-bit.

TABLE III: Comparison with customized FPGA accelerators/processors on lightweight CNNs.

	[15]	[16]	[13]	Light-OPU		**MP-OPU**	
Year	2018	2018	2019	2020		**2021**	
Device	Stratix V 5SGSD8	Arria 10 SoC	2 Stratix 10	XC7K325T		**XC7VX690T**	
Network	MobileNet-V1	MobileNet-V2	MobileNet-V1	MobileNet-V1	MobileNet-V2	**MobileNet-V1**	**MobieNet-V2**
Bit width	16	8	mixed [1]	8		**mixed**	
Frequency (MHz)	133	150	156	200		**200**	
DSP Used	1641	1278	-	704		**3072** [2]	
Inference latency (ms)	4.33	3.76	0.32	3.78	3.07	**0.47**	**0.34**
Throughput/DSP (GOPS)	0.13	0.06	-	0.21	0.14	**0.38**	**0.29**

[1] Weights are mixed precision while activations are 8-bit.
[2] 2048 DSPs are used for low precision multipliers while 1024 DSPs are used for adders.

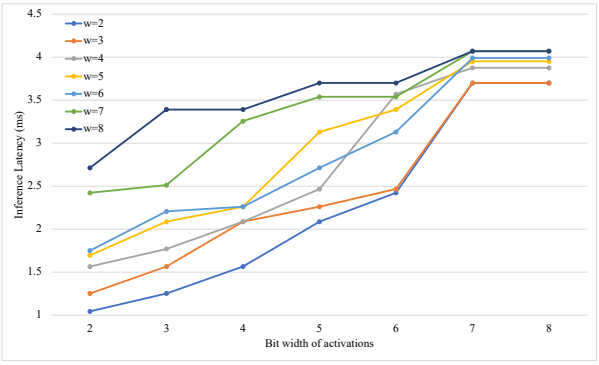

Fig. 4: Inference latency of Tiny-Yolo-V3 with respect to different combinations of bit width. In the legends, "w=2" means the bit width of weights is 2.

V. Conclusions

In this paper, we propose a Mixed Precision Processor on FPGA (**MP-OPU**) to leverage the advantages of mixed precision CNNs. We reuse part of the instructions and compilation flow in Light-OPU and redesign the hardware processor. To support mixed precision CNNs, the computation core is designed to be run-time re-configurable to have different number of multipliers according to the given precision. Meanwhile, the ping-pong architecture, pre-fetch and data rearrangement logic in the memory system make fully utilization of the bandwidth of the external memory. By mapping on Xilinx VC709, **MP-OPU** can reach 4.92 TOPS peak throughput when configuring to only support 2-bit. Our experimental results show that **MP-OPU** manages to reduce inference latency by 12.9× and increase throughput/DSP by 2.2× for conventional CNNs on average, respectively. Also, the average latency reduction is 7.6× and the throughput/DSP increment is 2.9× for lightweight CNNs.

Acknowledgment

This work is supported in part by the Natural Science Foundation for Distinguished Young Scholars of Jiangsu Province, China under Grant BK20200038.

References

[1] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[2] Y. Yu, C. Wu, T. Zhao, K. Wang, and L. He, "Opu: An fpga-based overlay processor for convolutional neural networks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 1, pp. 35–47, 2019.

[3] C. Wu, M. Wang, X. Chu, K. Wang, and L. He, "Low precision floating-point arithmetic for high performance fpga-based cnn acceleration," *arXiv preprint arXiv:2003.03852*, 2020.

[4] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization," *arXiv preprint arXiv:1612.01064*, 2016.

[5] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han, "Haq: Hardware-aware automated quantization with mixed precision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019, pp. 8612–8620.

[6] S. Markidis, S. W. Der Chien, E. Laure, I. B. Peng, and J. S. Vetter, "Nvidia tensor core programmability, performance & precision," in *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE, 2018, pp. 522–531.

[7] H. Sharma, J. Park, N. Suda, L. Lai, B. Chau, V. Chandra, and H. Esmaeilzadeh, "Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural network," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2018, pp. 764–775.

[8] Y. Umuroglu, L. Rasnayake, and M. Själander, "Bismo: A scalable bit-serial matrix multiplication overlay for reconfigurable computing," in *2018 28th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2018, pp. 307–3077.

[9] Y. Yu, T. Zhao, K. Wang, and L. He, "Light-opu: An fpga-based overlay processor for lightweight convolutional neural networks," in *The 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2020, pp. 122–132.

[10] A. Maki, D. Miyashita, K. Nakata, F. Tachibana, T. Suzuki, and J. Deguchi, "Fpga-based cnn processor with filter-wise-optimized bit precision," in *2018 IEEE Asian Solid-State Circuits Conference (A-SSCC)*. IEEE, 2018, pp. 47–50.

[11] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1," *arXiv preprint arXiv:1602.02830*, 2016.

[12] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer, "Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 734–10 742.

[13] Y. Zhao, X. Gao, X. Guo, J. Liu, E. Wang, R. Mullins, P. Y. Cheung, G. Constantinides, and C.-Z. Xu, "Automatic generation of multi-precision multi-arithmetic cnn accelerators for fpgas," in *2019 International Conference on Field-Programmable Technology (ICFPT)*. IEEE, 2019, pp. 45–53.

[14] A. Ahmad, M. A. Pasha, and G. J. Raza, "Accelerating tiny yolov3 using fpga-based hardware/software co-design," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2020, pp. 1–5.

[15] R. Zhao, X. Niu, and W. Luk, "Automatic optimising cnn with depthwise separable convolution on fpga: (abstact only)," in *Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2018, pp. 285–285.

[16] L. Bai, Y. Zhao, and X. Huang, "A cnn accelerator on fpga using depthwise separable convolution," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, no. 10, pp. 1415–1419, 2018.