IEICE Electronics Express, Vol.19, No.10, 1–6

LETTER

CASSANN-v2: A high-performance CNN accelerator architecture with on-chip memory self-adaptive tuning

Feng Liu[1, 2, 3], **Ruixiu Qiao**[1, 3], **Gang Chen**[1, 2, 3, 4], **Guoliang Gong**[1, 2, 3, 4, a)], **and Huaxiang Lu**[1, 2, 3, 4, 5]

Abstract This work proposes a high-performance reconfigurable CNN accelerator architecture, called CASSANN-v2, which can achieve 1TOPS peak performance at 1 GHz. CASSANN-v2 provides the function of on-chip SRAM memory real-time adaptive tuning by parameter configuration to reduce the intermediate output data transmission to further exploit the acceleration performance. The system simulation results show that CASSANN-v2 exhibits excellent performance on VGG-16 and ResNet-18 inference, with a throughput of 1009.54GOPS and 923.24GOPS at 1 GHz, which achieved 98.59% and 90.20% average processing element utilization, respectively. Compared with state-of-the-art accelerator works, CASSANN-v2 improves the resource utilization by 2.02× in VGG-16 and 2.35× in ResNet-18.

Keywords: SoC design, convolutional neural network (CNN) accelerator, high-performance accelerator, architecture optimization

Classification: Integrated circuits

1. Introduction

In the past decade, the extraordinary performance of Deep convolutional neural networks (DCNN) in machine vision, language processing has attracted much attention in various fields [1, 2, 3]. Nevertheless, DCNNs have high-throughput and high-energy consumption defects due to the computational density and the large storage requirements. It brings enormous restrictions to the further application of DCNN to practical tasks on power-constrained devices such as rescue robots [4], mobile medical auxiliary equipment [5], and automatic driving [6]. As a result, these difficulties attracted great interest in DCNN hardware accelerators design to quickly and efficiently complete CNN inference computations.

Many energy-efficient hardware accelerators have been proposed to reduce power consumption and improve the speed of DCNN computing in recent years. These accelerators based on application-specific integrated circuits (ASIC) [7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19] and field-programmable gate array (FPGA) [20, 21, 22, 23, 24, 25, 26] have achieved low latency and high efficiency on CNN computing. Two classic CNN of AlexNet and VGG have been demonstrated the excellent performance earlier, including UNPU [7], DSIP [12], Eyeriss [13], and DNPU [18]. Moreover, some works exploited the accelerators performance by reducing off-chip memory access [8, 27, 28].

However, most of the ASIC-based accelerators can only support one or several networks with similar structures. Some flexible CNN accelerators [9, 14, 20, 23, 24, 29] have been proposed to deal with it, which adapt to more advanced popular CNN models, such as ResNet and MobileNet. In general, the FPGA-based designs are more flexible than ASIC-based designs, mainly due to the natural advantages of FPGA flexibility. Nevertheless, these FPGA designs make an insufficient balance between hardware resources and performance because they only need to consider the upper limit of FPGA board resources. They used enormous data bus resources, the bus width up to 1024-bit, to achieve the fast data transfer between the off-chip and accelerator [23, 24]. However, such a wide bus is always unacceptable in expensive ASIC implementations.

From the existing successful accelerator studies, three instructive conclusions can be summarized. First, the power of off-chip data access such as DRAM is much higher than SRAM. But the speed is slower than SRAM, so reducing the external memory access is effective in saving energy and accelerating CNN's inference [8, 28]. Secondly, flexibility should be one of the essential features for expensive tailored design, which can expand the application scenarios of the accelerator [19, 23]. Finally, the bandwidth is the bottleneck of the CNN accelerator. Maximizing the bus bandwidth utilization can improve the performance of the accelerator [7, 9].

In this paper, we combine some existing valuable conclusions with our previous successes [27] to propose a high-performance and high-flexible CNN accelerator architecture based on ASIC, called CASSANN-v2. CASSANN-v2 uses a flexible memory structure and ingenious memory management method while optimizing the computing strategy on-chip for high resource utilization. Furthermore, we design an SoC architecture according to the CASSANN-V2; it is synthesized and simulated at 1GHz. The experimental results indicate that the architecture has achieved a 1TOPS peak performance and up to average resource utilization of 98.59%

[1] Institute of Semiconductors, Chinese Academy of Sciences, Beijing, 100083, China
[2] University of Chinese Academy of Sciences, Beijing, 100089, China
[3] Semiconductor Neural Network Intelligent Perception and Computing Technology Beijing Key Laboratory, Beijing,100083, China
[4] Materials and Optoelectronics Research Center, University of Chinese Academy of Sciences, Beijing, 100089, China
[5] College of Microelectronics, University of Chinese Academy of Sciences, Beijing, 100089, China
[a)] gongmianjie@semi.ac.cn

DOI: 10.1587/elex.19.20220124
Received March 10, 2022
Accepted March 29, 2022
Publicized April 7, 2022
Copyedited May 25, 2022

in VGG-16 inference. More specifying, CASSANN-v2 has the following features.
(1) Flexible storage structure: The on-chip SRAM blocks of the global memory have no fixed function. They are assigned the designated function by instruction configuration, which can make real-time adjustments for different cases.
(2) High bandwidth utilization: By co-scheduling the on-chip global memory and the computing units, CASSANN-V2 can reduce the intermediate data transmission between DRAM and the accelerator to exploit the performance.
(3) High flexibility and high efficiency: We optimize the computation of the residual block by an independent shortcut module. And combined with the memory scheduling method, it can provide high efficient computation for different CNNs.

2. Accelerator microarchitecture

The main challenge of CNN accelerator design is how to support the acceleration of different CNNs in high resource utilization by one microarchitecture. The proposed accelerator explores the parallel acceleration in different situations, especially in computing resource reorganization that can adapt with most of the current popular CNNs. We design all the configurations into parameterization, which allows the accelerator to switch flexibly according to instruction parameters during runtime. CASSANN-v2 needs to be reconfigured for each layer because of the layer-by-layer calculation strategy.

The proposed SoC architecture consists of a general processor, a DMA controller, an on-chip DRAM controller, a CNN accelerator, a video input and some necessary I/O. They are interconnected by AXI4 data bus and APB control bus, as shown in Fig. 1. Different from the traditional SoC design, the general processor is an auxiliary device in the CNN accelerator design. Which is specially used for registers configurations, an open-source RSIC-V processor [30] is sufficient to deal with control tasks.

2.1 Architecture overview

It is deserved to be mentioned that DMA is important in this SoC design. CNN accelerator has intensive data interaction with the memory. All the memory data access of off-chip DRAM is completed by DMA, which can improve bus utilization. The proposed architecture supports external DRAM storage up to 4 GB. Moreover, we placed a video input interface to support up to 1920 × 1080 image data transfer at 60 frames per second.

CASSANN v2 consists of the global control unit, global memory, data router, shortcut module, and computing block. The global control unit can be further divided into an instruction memory, a global controller, an interrupt generator. All the scheduling of CASSANN v2 is completed by the control instruction and the configuration instruction. Before the accelerator starts, the global controller will configure the local controller of the accelerator according to the configuration instructions, which will be specific to each sub-unit. Control instructions drive the active operation updates during the CNN calculation. When the next operation is required, the accelerator sends an interrupt request and updates the corresponding status register. Then the processor will send the new control instructions to the accelerator.

2.2 The configurable global buffer

Designers always hope to place as sizeable on-chip memory as possible for CNN accelerator designs to improve performance. However, SRAM is an expensive hardware resource, and its usage is invariably limited by area. We take a reconfigurable design for the global memory to minimize off-chip data access while maximizing the efficient utilization of this valuable storage. Different from the works that explicitly demarcate the data input buffer and data storage bank of feature map and weight in physical circuit level [8, 9]. The global memory of CASSANN v2 provides the function of data storage and data transfer through instruction configuration, which is realized at the logic level. It enables adaptive adjustment of the functional storage areas to different CNN layers. Moreover, CASSANN v2 can avoid part of the output data transfer by switching the memory address of the input and output functional area during calculation.

Another essential function of global memory is to mediate input and output data. We reorganized the m blocks of SRAM to form a cache similar to a ping-pong buffer structure. However, in fact, these N blocks of SRAM form an N/m levels cache structure. As shown in the red and orange dashed boxes in Fig. 2. While one buffer's data is being read, one of the idle buffers will be written and updated in time, which maximizes the utilization of SoC bus bandwidth. For example, the input buffer area of global memory is divided into four parts A, B, C, D, respectively. Buffer A and B are functionally equivalent to a traditional ping-pong buffer, and they will provide the input data caching for weight, bias, and configuration instructions. During the accelerator work, one of the idle buffer blocks will become

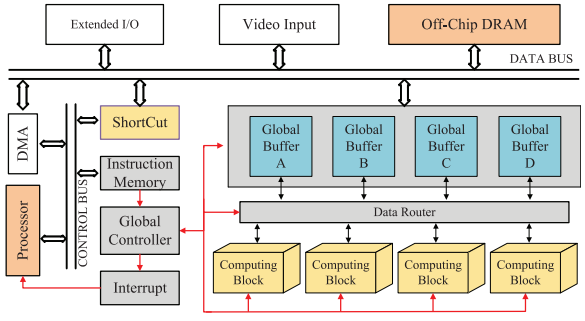

Fig. 1 Overview of microarchitecture.

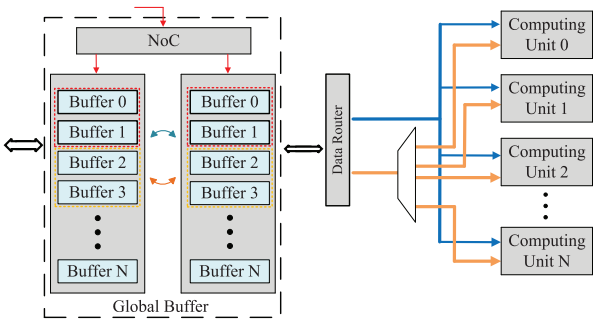

Fig. 2 The structure of the global memory.

the weight/bias/parameter input buffers, which is chosen by the on-chip state machine.

In addition, the feature map data will be stored in other buffers such as C and D. When the input buffer is switched to the output buffer, C and D will preferentially save part of the output feature map. If the output size is larger than the capacity of the output buffer, A and B will provide the off-chip DRAM data caching for the remaining output feature map data. Conversely, buffer A and B will be used as ordinary storage blocks to save the output data.

The bandwidth is one of the bottlenecks in the performance of CNN accelerators. Many conditions limit the data bus width of SoC. In this study, to avoid the input/output buffer dragging down the performance of the CASSANN v2, the number of the buffers m is required, as shown in Eq. (1).

$$m = \frac{WIDTH_{bus}}{WIDTH_{SRAM}} \tag{1}$$

Furthermore, the storage capacity of the buffer block needs to be at least larger than 4 KB to exploit the potential of the AXI bus. It is preferably a multiple of 4 KB. So the depth constraint of SRAM is shown in Eq. (2). i is the multiple factors of 4 KB.

$$DEPTH_{SRAM} = i \times \frac{4KB}{m \times WIDTH_{SRAM}} \tag{2}$$

The purpose of Eq. (1) is to align the ping-pong buffer with the SoC data bus, and Eq. (2) is to ensure bandwidth utilization by making the bulk data transmission and maintaining the AXI bus work in burst transmission mode. The capacity of the global memory is constrained by the number of blocks in the buffer, as shown in Eq. (3).

$$C_{glb} = 2^{k+1} \times m \times WIDTH_{SRAM} \times DEPTH_{SRAM} \tag{3}$$

In general, the capacity of the global memory is determined according to practical requirements because it is a flexible design. Larger global memory can obviously reduce the intermediate data transmission, thereby reducing the power consumption and time of the data access between the accelerator and off-chip memory. However, even if the capacity of global memory is very limited, the performance will not drop down significantly when the minimum storage requirement is met.

2.3 Computing block and calculation mode

Fig. 3 shows the computing block structure of CASSANN-v2. The proposed accelerator uses a united processing element (UPE) to compute the CNN efficiently. It is composed of 16 multipliers and an adder tree, as shown in Fig. 3 purple area. Each UPE is equipped with a feature map local buffer and a weight local buffer; it takes 16-group 8-bit weight and feature map as the input. The adder tree of UPE can summarize the output of multipliers and finish self-accumulation. Then the outputs of UPEs are connected to finish the accumulation and bias addition. The final sums precision is 32-bit.

The sums can be further processed by activation, pooling, and quantization module under the logic control. Finally, the results are sent to output FIFO. The computing unit (CU) is shown in the dotted box of Fig. 3, and it is composed of

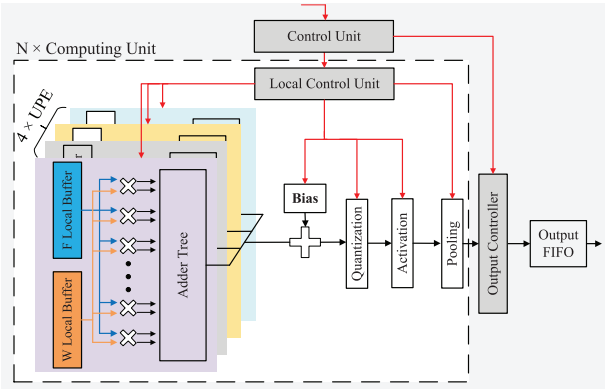

Fig. 3 The structure of computing block.

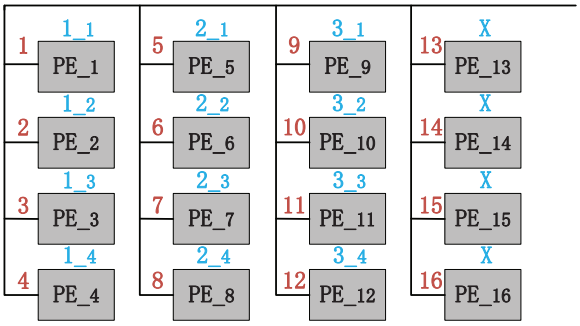

Fig. 4 The data flow of UPE.

4-UPE and a set of bias, quantization, activation, pooling modules. The additional functional modules are arranged in a pipeline structure. They are built into the CU to maximize computational efficiency and reduce bus workload. The computing block consists of N-group CU, which will send the results of the output FIFO to the designated address of the output buffer.

The input data of CASSANN v2 is the channel saved first, then the column, and finally the row direction. The data format is [row, column, channel], a prevailing data format in the CNN accelerator. CASSANN v2 utilizes two main inference modes: channel-oriented and channel-row mixed modes. We apply these two modes by reorganizing UPE to accommodate different computing requirements. It is why UPE is designed with 16-pair inputs. Because the channels number in CNN is usually a multiple of 16, such a setting can maintain 100% PE utilization when the CNN is computing in channel-oriented mode.

While the input channel $ch_i < 16$, CASSANN v2 works at row and channel mixed mode. For example, when $ch_i = 4$ and the kernel size is 3×3, the data order of UPE is the channel first and the column next, as shown in Fig. 4 blue sequence. UPE is independent working at the 1-group mode, the $PE_{13} - PE_{16}$ disabled. The data is output every three clock cycles. In addition, UPE can make adaptive adjustments according to the kernel size. If the kernel size is 5×5, UPE will work in the 2-group combined mode. The $PE_1 - PE_{20}$ are in normal computing, and the $PE_{21} - PE_{32}$ disabled. It is the same logic for other kernel sizes. The CU supports 4-group UPE working together at most, so the maximum kernel size must meet the following requirement:

$$Min(k_x, k_y) < \frac{64}{ch_i} \tag{4}$$

Moreover, the work mode of how many UPE can work together is as follows:

$$n = Round(\frac{ch_i \times k}{16}) \tag{5}$$

The ch_i is the number of input channels, k_x and k_y are the kernel sizes, k is the column number of kernel sizes. The most common case of the channel-row mixed mode is computing the first layer input. What needs to be explained is that we will fill zero to align the input channel to 4 when the $ch_i < 4$. Because the data needs to be aligned with the SoC bus, it can simplify the accelerator's state machine circuit logic.

When the input channel $ch_i \geq 16$, CASSANN v2 will use the channel-oriented calculation mode, as shown in Fig. 4. For instance, the $ch_i = 16$ and the kernel size is $k_x \times k_y$. The 16-channel data is sent to the UPE in parallel, and each UPE needs $k_x \times k_y$ cycles to obtain a final result. A CU works out the data of 4 output channels in parallel, and the UPE works independently in this case. When the $ch_i = 64$, each UPE undertakes 16-channel input data computing, and the 4-UPE in the CU work together to obtain a result after $k_x \times k_y$ clock cycles. UPE can work at 1-group, 2-group, 4-group synergistic modes. In other words, a CU can process 16, 32, 64 input channels data in parallel and output 4, 2, 1 channel results, respectively. It depends on the configuration of the accelerator.

The channel-oriented mode is the most frequent mode used for CNN inference of CASSANN v2. Another strategy to reduce the data transfer between the accelerator and off-chip memory is data reuse. The input buffer stores the feature map for global reuse, and the data is sent to each CU by broadcast. The data reuse strategy of the weight is local reuse, which makes the local buffer of weight need to store at least 16-channel convolutional kernel data. The minimum ping-pong buffer capacity of the local weight buffer is as follows:

$$C_{wlocal} = 16 \times k_x \times k_y \tag{6}$$

Considering the actual situation, the kernel size of popular CNN is usually less than 7×7 when the input channels are larger than 64. Moreover, the larger the local buffer for weights, the larger the data block that the SoC bus can send. It means the data transfer time can be shorter, which is beneficial to boost the accelerator's performance.

3. Experimental analysis and discussion

3.1 Experimental setup

We implement the SoC architecture of Fig. 1 in Verilog. The details of SoC are as follows. A 256-bit AXI4 bus is adapted to provide data interaction for the proposed SoC, and the bus work frequency is set as 800 MHz to match the operating frequency of the off-chip DRAM. In addition, we place 512 KB SRAM for global memory. There are four computing blocks in CASSANN-v2, and each computing block has 2-CUs that include 512-PEs in total. The local weight buffer of UPE is set to 2×2.4 KB, and the feature map local buffer is set to 2×1.2 KB. The size of these local ping-pong buffers is enough to support the accelerator to parallel complete 1024-channel data input and 8-channel data output where the convolution kernel size is 3×3. Furthermore, it should be noted that CU works in the state of self-circulation of the input channel, the time interval of data output are as follows:

$$\Delta t = \frac{ch_i \times k_x \times k_y \times clk}{64} \tag{7}$$

where the ch_i is input channels number, k_x and k_y is the convolution kernel size, the clk is clock period. CASSANN-v2 accelerator is work at 1 GHz. The proposed SoC has been synthesized in TSMC 28nm technology and simulated with GalaxSim of X-EPIC.

3.2 Experimental results analysis

Two classical CNNs, VGG-16 and ResNet-18, are used as benchmark networks to measure our proposed accelerator architecture. Fig. 5 shows the comparison between the intermediate feature map data generated by calculating and the temporary data that needs to be transmitted to the off-chip DRAM. It can be seen that the adaptive global memory of CASSANN-v2 has significantly reduced the feature maps off-chip data transfer of VGG-16 and Resnet-18. Especially in the ResNet-18, benefiting the output buffer is large enough to store the intermediate data, it reduced by 76.92%.

Table I provides a detailed comparison with state-of-the-art accelerators on running the VGG-16 convolutional layers at the COCO dataset. It can be seen that our work achieved a

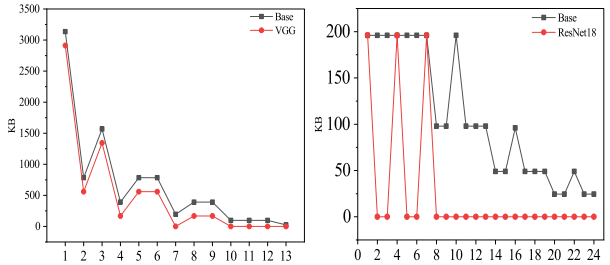

Fig. 5 (a) Output data amount of VGG-16. (b) Output data amount of ResNet-18. The red line is the amount of the actual feature map output data which needs to transfer to off-chip DRAM. The black line is the amount of generated output data in computing.

Table I Comparison of performance on VGG-16.

	ELEX'21 [25]	ITanji [26]	JSSC'21 [8]	CASSANN-X [27]	This Work
Platform	FPGA	FPGA	ASIC	ASIC	ASIC
Clock (MHz)	140	165	400	1000	1000
Precision (bit)	16	16	12	16 / 8	8
SRAM(KB)	NA	313	339.5	2100	742.4
MACs	576	256	324	512	512
Peak (GOPS)	161.28	84.48	259.2	1024	1024
Throughput (GOPS)	129.7	66.93	253.7	498.6	1009.54
Normalized Efficiency	80.42%	79.52%	97.88%	48.69%	98.59%

Table II Comparision of performance on ResNet.

	ELEX'21 [25]	APCCAS [29]	ISSCC'21 [19]	This Work
Platform	FPGA	FPGA	ASIC	ASIC
Clock (MHz)	140	199	470	1000
MACs	576	1518	864	512
Peak (GOPS)	161.28	604.16	812.16	1024
Throughput (GOPS)	92.16	318.4	331.28	923.24
Normalized Efficiency	57.14% ResNet-18	52.7% ResNet-18	38.34% ResNet-50	90.20% ResNet-18

throughput of 1009.54GOPS, which is far higher than other work. Part of that is because of the peak performance we can provide. The calculation method of peak performance is as follows:

$$Peak = MAC_{num} * 2 * Clock \tag{8}$$

Therefore, we normalized the results for a fair comparison of the performance between architectures. The normalized efficiency is calculated as:

$$Norm. = \frac{Actualthroughput}{peakperformance} \times 100\% \tag{9}$$

As we can see from the normalization result, CASSANN-v2 has a noticeable improvement in normalized efficiency compared with other accelerators. In particular, compared to CASSANN-X [27], we have improved performance by 2.02X and reduced SRAM consumption by 64.65%. Although we used 2.18× SRAM than the JSSC'21 [8] with only a slight improvement in normalized efficiency, CASSANN-v2 are more flexible than JSSC'21 [8], such as we can support the residual blocks computing directly.

Table II shows the ResNet-18 inference performance of CASSANN-v2. Our accelerator can still maintain a high normalized efficiency when CASSANN-v2 processes the residual blocks, which is a significant improvement compared to previous works, at least 33.06%. It benefits from the on-chip global memory, avoiding redundant off-chip data transfer. Moreover, we designed a shortcut module for the residual block to perform the computing with a data stream method. The time of residual block computing is hidden in the convolutional inference. The experiment results on different CNNs prove CASSANN-v2 is a high-performance and high-flexible architecture. It is achieved 98.59% efficiency of VGG-16 and 90.20% of ResNet-18, which is better than state-of-the-art works.

3.3 More discussion

There is no further analysis of power consumption in this paper. On the one hand, high-performance and low-power consumption are the two directions of architecture design. As a high-performance architecture, we have not completed the optimization for power consumption. On the other hand, the SoC architecture in this paper is designed at the whole system level, such as reducing the power consumption bu cutting down the off-chip memory data access. In the future, we will turn this SoC into an actual chip and apply it to a complete computing system to further analyze the power consumption droop by this architecture.

4. Conclusion

This paper proposes a high-performance CNN accelerator, CASSANN-v2, which uses a novel global memory structure and management method. CASSANN-v2 achieves the adaptive memory adjustment by the parameter configuration to reduce the intermediate output data transmission. In addition, the cooperation between the global memory and the local buffer can maximize the bus utilization rate to improve performance. The SoC system simulation results of VGG-16 and ResNet-18 show that CASSANN-v2 can significantly reduce the temporary feature map data transmission. When the accelerator worked at 1 GHz, it achieved 1009.54 GOPS and 923.24 GOPS throughput in VGG-16 and ResNet-18. Compared with other works, CASSANN-v2 has the highest normalized efficiency, which improved by 63.34% on ResNet at least. It is a remarkable improvement, proving that CASSANN-v2 is an excellent architecture for CNN acceleration.

Acknowledgments

The work was supported by National Natural Science Foundation of China (U19A2080, U1936106), in part by High-Technique Projects under Grant (31513070501, 1916312ZD00902201), in part by Chinese Academy of Sciences Strategic Leading Science and Technology Project (XDB44000000, XDA27040303, XDA18040400), and in part by the Beijing Academy of Artificial Intelligence (BAAI).

References

[1] Y. LeCun, *et al.*: "Deep learning," Nature **521** (2015) 436 (DOI: 10.1038/nature14539).
[2] Z.H. Wu, *et al.*: "A comprehensive survey on graph neural networks," IEEE Trans. Neural Netw. Learn. Syst. **32** (2021) C2 (DOI: 10.1109/tnnls.2020.2978386).
[3] V. Sze, *et al.*: "Efficient processing of deep neural networks: a tutorial and survey," Proc. IEEE **105** (2017) 2295 (DOI: 10.1109/jproc.2017.2761740).
[4] M.B. Bejiga, *et al.*: "A convolutional neural network approach for assisting avalanche search and rescue operations with UAV imagery," Remote Sensing **9** (2017) 100 (DOI: 10.3390/rs9020100).
[5] H.C. Shin, *et al.*: "Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning," IEEE Trans. Med. Imag. **35** (2016) 1285 (DOI: 10.1109/tmi.2016.2528162).
[6] Y.C. Tian, *et al.*: "DeepTest: automated testing of deep-neural-network-driven autonomous cars," ICSE (2018) 303 (DOI: 10.1145/3180155.3180220).
[7] J. Lee, *et al.*: "UNPU: an energy-efficient deep neural network accelerator with fully variable weight bit precision," IEEE J. Solid-State Circuits **54** (2019) 173 (DOI: 10.1109/jssc.2018.2865489).
[8] M. Kim and J.S. Seo: "An energy-efficient deep convolutional neural network accelerator featuring conditional computing and low external memory access," IEEE J. Solid-State Circuits **56** (2021) 803 (DOI: 10.1109/JSSC.2020.3029235).
[9] J. Sim, *et al.*: "An energy-efficient deep convolutional neural network inference processor with enhanced output stationary dataflow in 65-nm CMOS," IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **28**

(2019) 87 (DOI: 10.1109/tvlsi.2019.2935251).
[10] T.S. Chen, *et al.*: "DianNao: a small-footprint high-throughput accelerator for ubiquitous machine-learning," ACM Sigplan Notices **49** (2014) 269 (DOI: 10.1145/2541940.2541967).
[11] T. Luo, *et al.*: "DaDianNao: a neural network supercomputer," IEEE Trans. Comput. **66** (2017) 73 (DOI: 10.1109/tc.2016.2574353).
[12] J. Jo, *et al.*: "DSIP: a scalable inference accelerator for convolutional neural networks," IEEE J. Solid-State Circuits **53** (2018) 605 (DOI: 10.1109/jssc.2017.2764045).
[13] Y.-H. Chen, *et al.*: "Eyeriss: an energy-efficient reconfigurable accelerator for deep convolutional neural networks," IEEE J. Solid-State Circuits **52** (2017) 127 (DOI: 10.1109/jssc.2016.2616357).
[14] Y.-H. Chen, *et al.*: "Eyeriss v2: a flexible accelerator for emerging deep neural networks on mobile devices," IEEE J. Emerg. Sel. Topics Circuits Syst. **9** (2019) 292 (DOI: 10.1109/jetcas.2019.2910232).
[15] Y. Yamada, *et al.*: "A 20.5 TOPS multicore SoC with DNN accelerator and image signal processor for automotive applications," IEEE J. Solid-State Circuits **55** (2020) 120 (DOI: 10.1109/jssc.2019.2951391).
[16] B. Zimmer, *et al.*: "A 0.32–128 TOPS, scalable multi-chip-module-based deep neural network inference accelerator with ground-referenced signaling in 16 nm," IEEE J. Solid-State Circuits **55** (2020) 920 (DOI: 10.1109/JSSC.2019.2960488).
[17] J. Pei, *et al.*: "Towards artificial general intelligence with hybrid Tianjic chip architecture," Nature **572** (2019) 106 (DOI: 10.1038/s41586-019-1424-8).
[18] D. Shin, *et al.*: "14.2 DNPU: an 8.1 TOPS/W reconfigurable CNN-RNN processor for general-purpose deep neural networks," ISSCC (2017) 240 (DOI: 10.1109/ISSCC.2017.7870350)
[19] H. Mo, *et al.*: "9.2 A 28 nm 12.1 TOPS/W dual-mode CNN processor using effective-weight-based convolution and error-compensation-based prediction," ISSCC (2021) 146 (DOI: 10.1109/ISSCC42613.2021.9365943).
[20] S. Colleman and M. Verhelst: "High-utilization, high-flexibility depth-first CNN coprocessor for image pixel processing on FPGA," IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **29** (2021) 461 (DOI: 10.1109/tvlsi.2020.3046125).
[21] Y. Parmar and K. Sridharan: "A high-performance VLSI architecture for a self-feedback convolutional neural network," IEEE Trans. Circuits Syst. II, Exp. Briefs **68** (2021) 456 (DOI: 10.1109/tcsii.2020.3004616).
[22] Y. Sun, *et al.*: "An OpenCL-based hybrid CNN-RNN inference accelerator on FPGA," ICFPT (2019) 283 (DOI: 10.1109/ICFPT47387.2019.00048).
[23] Y.X. Yu, *et al.*: "OPU: an FPGA-based overlay processor for convolutional neural networks," IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **28** (2020) 35 (DOI: 10.1109/tvlsi.2019.2939726).
[24] X. Qu, *et al.*: "Cheetah: an accurate assessment mechanism and a high-throughput acceleration architecture oriented toward resource efficiency," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. **40** (2021) 878 (DOI: 10.1109/TCAD.2020.3011650).
[25] H. Jia, *et al.*: "An FPGA-based accelerator for deep neural network with novel reconfigurable architecture," IEICE Electron. Express **18** (2021) 20210012 (DOI: 10.1587/elex.18.20210012).
[26] H.Z. Zhu, *et al.*: "Tanji: a general-purpose neural network accelerator with unified crossbar architecture," IEEE Des. Test **37** (2020) 56 (DOI: 10.1109/mdat.2019.2952329).
[27] R. Qiao, *et al.*: "High performance reconfigurable accelerator for deep convolutional neural networks (in Chinese)," Journal of Xidian University **46** (2019) 130 (DOI: 10.19665/j.issn1001-2400.2019.03.020).
[28] R.V.W. Putra, *et al.*: "ROMANet: fine-grained reuse-driven off-chip memory access management and data organization for deep neural network accelerators," IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **29** (2021) 702 (DOI: 10.1109/tvlsi.2021.3060509).
[29] J. Wen, *et al.*: "An efficient FPGA accelerator optimized for high throughput sparse CNN inference," APCCAS (2020) 165 (DOI: 10.1109/APCCAS50809.2020.9301696).
[30] K. Asanovic, *et al.*: "The rocket chip generator," Dept. EECS, Univ. California, Berkeley, Tech. Rep. (2016) UCB/EECS-2016-17.