Design Automation for Binarized Neural Networks: A Quantum Leap Opportunity?

Design Automation for Binarized Neural Networks: A Quantum Leap Opportunity?

Manuele Rusci*, Lukas Cavigelli†, Luca Benini†*
* Energy-Efficient Embedded Systems Laboratory, University of Bologna, Italy – manuele.rusci@unibo.it
†Integrated Systems Laboratory, ETH Zurich, Switzerland – {cavigelli, benini}@iis.ee.ethz.ch

***Abstract*—Design automation in general, and in particular logic synthesis, can play a key role in enabling the design of application-specific Binarized Neural Networks (BNN). This paper presents the hardware design and synthesis of a purely combinational BNN for ultra-low power near-sensor processing. We leverage the major opportunities raised by BNN models, which consist mostly of logical bit-wise operations and integer counting and comparisons, for pushing ultra-low power deep learning circuits close to the sensor and coupling them with binarized mixed-signal image sensor data. We analyze area, power and energy metrics of BNNs synthesized as combinational networks. Our synthesis results in GlobalFoundries 22 nm SOI technology shows a silicon area of 2.61 mm^2 for implementing a combinational BNN with 32×32 binary input sensor receptive field and weight parameters fixed at design time. This is 2.2× smaller than a synthesized network with re-configurable parameters. With respect to other comparable techniques for deep learning near-sensor processing, our approach features a 10× higher energy efficiency.**

I. Introduction

Bringing intelligence close to the sensors is an effective strategy to meet the energy requirements of battery-powered devices for always-ON applications [1]. Power-optimized architectures for near-sensor processing target the reduction of the amount of data to be dispatched out from the sensors, along with significant energy savings, by applying local data processing over raw sensed data [2]. To this aim, in the context of visual sensing, novel computer vision chips include in-sensor computational modules for the early extraction of mid- and low- level visual features, which are transferred to a digital processing unit for further computation or used to feed a first stage classifier [3]. Thanks to the integration of analog processing circuits on the focal-plane, the amount of data crossing the costly analog-to-digital border is reduced [4] and, if compared with a camera-based system featuring a traditional imaging technology, the system energy consumption is lower because of (a) a reduced sensor-to-processor bandwidth and (b) a lower demand for digital computation [5]. Relevant examples of mixed-signal computer vision capabilities include the extraction of spatial and temporal features, such as edges or frame-difference maps, or a combination of them [6]. Because of the employed highly optimized architectures, the power consumption of these smart visual chips results to be more than one order of magnitude lower than off-the-shelf traditional image sensors [7].

However, to favor the meeting between smart ultra-low power sensing and deep learning, which is nowadays the leading technique for data analytics, a further step is required. At present, the high computational and memory requirement of deep learning inference models have prevented a full integration of these approaches close to the sensor at an ultra low power cost [4], [8]. A big opportunity for pushing deep learning into low-power sensing come from recently proposed Binarized Neural Networks (BNNs) [9], [10]. When looking at the inference task, a BNN consists of logical XNOR operations, binary popcounts and integer thresholding. Therefore, major opportunities arise for hardware implementation of these models as part of the smart sensing pipeline [11].

In this paper, we explore the feasibility of deploying BNNs as a front-end for an ultra-low power smart vision chip. The combination of mixed-signal processing and hardware BNN implementation represents an extremely energy-efficient and powerful solution for always-ON sensing, serving as an early detector of interesting events. Therefore, we design and synthesize a purely combinational hard-wired BNN, which is fed with the binary data produced by a mixed-signal ultra-low power imager [7]. The main contributions of this paper are:

- The hardware design and logic synthesis of a combinational BNN architecture for always-ON near-sensor processing.
- The area and energy evaluation of the proposed approach, for varying network models and configurations.

We evaluate two BNN models with 16×16 and 32×32 binary input size, either with fixed or variable parameters. In case of a combinational BNN with 32×32 input data and hardwired parameters, our synthesis results in GlobalFoundries 22 nm SOI technology shows an area occupancy of 2.61 mm^2, which is 2.2× smaller than the model with variable parameters, and features a 10× higher energy efficiency with respect to comparable techniques for deep learning-based near-sensor processing. Moreover, our study paves the way for exploring a new generation of logic synthesis tools—aimed at aggressively optimizing deep binarized networks and enabling focal-plane processing of images with higher resolution.

II. Related Work

Besides the mixed-signal processing circuits for extracting basic spatial and temporal visual features in-the-sensor [6], [12], [13], recent approaches tried to push deep learning circuits into the analog side to exploit the energy benefits of focal-plane processing [3]. The work presented in [14] makes use of angle-sensitive pixels, integrating diffraction gratings on the focal plane. Based on the different orientations of the pixel-level filters, multiple feature maps are locally computed as the first layer of a convolutional network. RedEye [4] embeds column-wise processing pipelines in the analog domain to perform 3D convolutions before of the digital conversion. The chip is implemented in 0.18 μm technology and needs 1.4 mJ to process the initial 5 layers of GoogLeNet, leading to an energy efficiency of about 2 TOp/s/W. A sensing front-end supporting analog multiplication is proposed in [15] to operate

This project was supported in part by the EU's H2020 programme under grant no. 732631 (OPRECOMP) and by the Swiss National Science Foundation under grant 162524 (MicroLearn).

on analog sensed data. The authors introduce a MAC unit composed of only passive switches and capacitors to realize a switched-capacitor matrix multiplier, which achieves an energy efficiency of 8.7 TOp/s/W when running convolution operations. With respect to these analog approaches, we leverage the potentiality of BNNs to deploy a digital and optimized purely combinational network to notably increase the energy efficiency of near-sensor processing circuits.

On the near-sensor digital side, many neural network accelerators have been reported in the literature, most of them with an energy efficiency in the range of few TOp/s/W [16]–[18]. Several recent approaches have focused on quantizing the weights down to binarization in order to gain a significant advantage in memory usage and energy efficiency [8], [18], pushing it up to around 60 TOp/s/W while advances in training methods have achieved accuracy losses of less than 1% for this setup. A new approach has been to quantize also the activations down to binary with initial accuracy losses of up to 30% on the ILSVRC dataset, these have improved to around 11% over the last two years and even less for smaller networks on datasets such as CIFAR-10 and SVHN [9], [10], [18]. During this time, some VLSI implementations have been published, most of them targeting FPGAs such as the FINN framework [11], [19]. Only few ASIC implementations exist [19]–[21], of which XNOR-POP uses in-memory processing and reports the highest energy efficiency of 22.2 TOp/s/W and thus less than the best binary-weight-only implementation.

III. Combinational Hardware BNN Design

A BNN design represents both the networks weights and the activation layers with single-bit precision, leading to an intrinsic 32× memory footprint reduction with respect to a baseline full-precision model. When applying the binarization scheme to a Convolutional Neural Network (CNN), the resulting BNN features a stacked architecture of binary convolutional layers, where every layer transforms IF binary input feature maps into OF binary output feature maps through the well-known convolution operation. Because of the binary domain, denoted as {0,1}, of both the input data and the weight filters, the convolution kernel can be rewritten as

$$\varphi(m,x,y) = \text{popcount}(\textit{weights(m)} \text{ xnor } \textit{recField(x,y)}), \quad (1)$$

where $\varphi(m,x,y)$ is the result of the convolution, *weights(m)* is the array of binary filter weights and *recField(x,y)* is the receptive field of the output neuron located at position (x,y) of the m-th output feature map. The $\text{popcount}(\cdot)$ function returns the numbers of asserted bits of the argument. Note that the convolution output $\varphi(m,x,y)$ is an integer value. As presented by [9], the popcount result is binarized after a batch normalization layer. However, the normalization operation can be reduced to a comparison with an integer threshold,

$$outMap(m,x,y) = \begin{cases} \varphi(m,x,y) \geq thresh(m) & \text{if } \gamma > 0 \\ \varphi(m,x,y) \leq thresh(m) & \text{if } \gamma < 0 \\ 1 \quad \text{if } \gamma = 0 \text{ and } \beta \geq 0 & \\ 0 \quad \text{if } \gamma = 0 \text{ and } \beta < 0 & \end{cases}, \quad (2)$$

where $thresh(m)$ is the integer threshold that depends on the convolution bias b and on the parameters learned by the batch normalization layer μ, σ, γ and β. After training the network, the $thresh(m)$ parameters are computed offline as $\lfloor \mu - b - \beta \cdot \sigma/\gamma \rfloor$ if $\gamma > 0$ or $\lceil \mu - b - \beta \cdot \sigma/\gamma \rceil$ if $\gamma < 0$.

Fig. 1 graphically schematizes the binary convolution operation. The *BinConv* module applies (1) and (2) over the

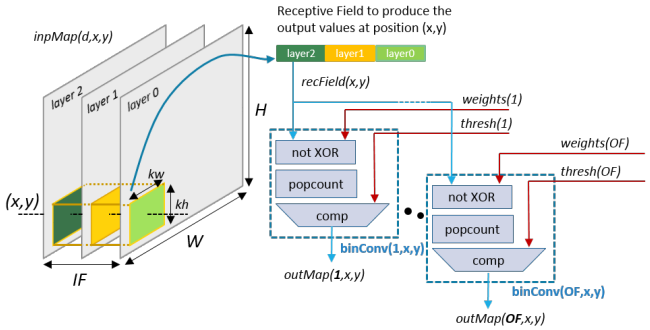

Fig. 1. Computational flow of a binary convolutional layer. For each of the OF output feature maps, the binary value at position (x,y) is produced by overlapping the m-th weight filter to the array of the receptive field of the input feature map centered at the spatial position (x,y).

receptive field values associated to any of the output neurons $outMap(m,x,y)$. Hence, to build a convolutional layer, the *BinConv* is replicated for every output neuron. The hardware architecture of a *BinConv* element is shown in Fig. 2. In the figure, the input signals *recField(x,y)*, *weights(m)* and *thresh(m)* and the output signal *outMap(m,x,y)* of the block refer to (1) and (2). Additionally, the *sign(m)* signal drives the selection of the correct output neuron's value depending on the batch normalization parameters (eq. (2)). The network parameters, *weights, thresh* and *sign*, highlighted in red, can be stored in a memory block, to allow online reconfiguration, or can be fixed at design time. In the first case, the memory footprint required to store the parameters of a convolutional layer is $OF \cdot (IF \cdot kw \cdot kh + \lfloor \log_2(IF \cdot kw \cdot kh) \rfloor + 3)$ bits. On the contrary, if hard-wiring the binary weights, the circuit implementation lacks flexibility but benefits in terms of silicon occupation.

To explore the feasibility of deep combinational BNNs, we focus on VGG-like network topologies as in [9]. These networks include convolutional layers with a small filter size (typically $kw = kh = 3$) and an increasing feature dimension going deeper into the network. The spatial dimension tends to decrease by means of strided pooling operations placed after the binary convolution of (1). Following the intuition of [11], a MaxPooling layer can be moved behind the binarization by replacing the MAX with an OR operation among the binary values of the activations. The network frontend is composed by multiple fully-connected layers. Their hardware implementation is similar to the *binConv* module of Fig. 2, where the convolutional receptive field contains all the input neurons of the layer. The last fully-connected layer generates a confidence score for every class. Differently from the original BNN scheme, our network architecture is fed with a binary

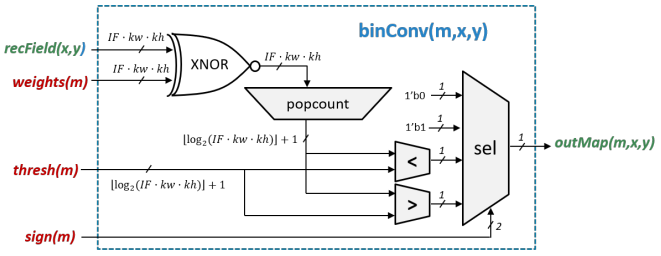

Fig. 2. Hardware architecture of the combinational building block for computing binary convolutions. Every *binConv(m,x,y)* module instantiated within a convolutional layer produces the binary value of the output neuron at location (x,y) of the m-th output feature map.

TABLE I
VGG-LIKE BNN MODELS[1]

layer	Model with a 16×16 input map	Model with a 32×32 input map
1	bConvLyr3x3(1,16)+MaxP2x2	bConvLyr3x3(1,16)+MaxP2x2
2	bConvLyr3x3(16,32)+MaxP2x2	bConvLyr3x3(16,32)+MaxP2x2
3	bConvLyr3x3(32,48)+MaxP2x2	bConvLyr3x3(32,48)+MaxP2x2
4	bFcLyr(192,64)	bConvLyr3x3(48,64)+MaxP2x2
5	bFcLyr(64, 4)	bFcLyr(256,64)
6		bFcLyr(64, 4)

single-layer signal coming from a mixed-signal imager [7]. However, the presented approach also holds for multi-channel imagers.

A. Estimating Area

Before looking at synthesis results, we estimate the area of a binary convolutional layer. For each output value (output pixel and feature map, $N_{\text{out}} = H \cdot W \cdot OF$), we have a receptive field of size $N_{\text{RF}} = IF \cdot kw \cdot kh$ and thus need a total of $N_{\text{out}} N_{\text{RF}}$ XNOR gates. These are followed by popcount units—adder trees summing over all N_{RF} values in the receptive field. The resulting full-precision adder trees require $\sum_{i=1}^{\log_2(N_{\text{RF}})} N_{\text{RF}} 2^{-i} = N_{\text{RF}} - 1$ half-adders and $\sum_{i=1}^{\log_2(N_{\text{RF}})} (i-1) N_{\text{RF}} 2^{-i} = N_{\text{RF}} - \log_2(N_{\text{RF}}) - 1$ full-adders (FAs) each, and are replicated for every output value. The subsequent threshold/compare unit is insignificant for the total area.

To provide an example, we look at the first layer of the network for 16×16 pixel images with 1 input and 16 output feature maps and a 3×3 filter ($N_{\text{RF}} = 9, N_{\text{out}} = 4096$). Evaluating this for the GF22 technology with $A_{\text{XNOR}} = 0.73\,\mu\text{m}^2$, $A_{\text{HA}} = 1.06\,\mu\text{m}^2$ and $A_{\text{FA}} = 1.60\,\mu\text{m}^2$, we obtain an area of $A_{\text{XNOR,tot}} = 0.027\,\text{mm}^2$, $A_{\text{HA,tot}} = 0.033\,\text{mm}^2$ and $A_{\text{FA,tot}} = 0.029\,\text{mm}^2$—a total of $0.089\,\text{mm}^2$. Note that this implies that the area scales faster than linearly with respect to the size of the receptive field N_{RF} since the word width in the adder tree increases rapidly. This is not accounted for in the widely used GOp/img complexity measure for NNs, as it is becoming only an issue in this very low word-width regime.

IV. EXPERIMENTAL RESULTS

A. BNN Training

The experimental analysis focuses on two VGG-like network topologies described in Tbl. I to investigate the impact of different input and network size. As a case-study, we trained the networks with labelled patches from the MIO-TCD dataset [22] belonging to one of the following classes: car, pedestrian, cyclist and background. The images from the dataset are resized to fit the input dimension before applying a non-linear binarization, which simulates the mixed-signal preprocessing of the sensor [7]. By training the BNNs with ADAM over a training set of about 10ksamples/class (original images are augmented by random rotation), the classification accuracy against the test-set achieves 64.7% in case of the model with 32×32 input data, while a 50% is measured for the 16×16 model because of the smaller input size and network. Since this work focuses on hardware synthesis issues of BNN inference engines, we do not explore advanced training approaches for NNs with non-traditional input data, which have been discussed in the literature [23].

[1]bConvLyr3x3(x,y) indicates a binary convolutional layer with a 3×3 filter, x input and y output feature maps, MaxP2x2 is a max pooling layer of size 2×2, bFcLyr(x, y) is a binary fully connected layer with x binary input y binary output binary neurons.

TABLE II
SYNTHESIS AND POWER RESULTS FOR DIFFERENT CONFIGURATIONS

		area		time/img		E/img	leak.	E-eff.
netw.	type	[mm²]	[MGE]†	[ns]	[FO4]‡	[nJ]	[µW]	[TOp/J]
16×16	var.	1.17	5.87	12.82	560	2.40	945	470.8
16×16	fixed	0.46	2.32	12.40	541	1.68	331	672.6
32×32	var.	5.80	29.14	17.27	754	11.14	4810	479.4
32×32	fixed	2.61	13.13	21.02	918	11.67	1830	457.6

† Two-input NAND-gate size equivalent: $1\,\text{GE} = 0.199\,\mu\text{m}^2$
‡ Fanout-4 delay: $1\,\text{FO4} = 22.89\,\text{ps}$

TABLE III
AREA BREAKDOWN FOR THE 16×16 NETWORK

layer	compute [kOp/img]	area estim. [mm²]	var. weights area [mm²]	fixed weights area [mm²]
1	74 (6.5%)	0.093	0.077 (6.6%)	0.008 (1.7%)
2	590 (52.2%)	0.971	0.647 (55.4%)	0.204 (44.3%)
3	442 (39.1%)	0.738	0.417 (35.8%)	0.241 (52.3%)
4	25 (2.2%)	0.041	0.026 (2.2%)	0.008 (1.7%)

B. Synthesis Results

We analyze both aforementioned networks for two configurations, with weights fixed at synthesis time and with variable weights (excl. storage, modeled as inputs). The fixed weights are taken from the aforementioned trained models.

We provide an overview of synthesis results for different configurations in Tbl. II. We synthesized both networks listed in Tbl. I in GlobalFoundries 22 nm SOI technology with LVT cells in the typical case corner at 0.65 V and 25°C. The configuration with variable weights scales with the computational effort associated with the network (1.13 MOp/img and 5.34 MOp/img for the 16×16 and 32×32 networks) with 0.97 and 0.92 MOp/cycle/mm², respectively. The variable parameters/weights configuration does not include the storage of the parameters themselves, which would add $1.60\,\mu\text{m}^2$ (8.0 GE) per FF. This weight memory could be loaded through a scan-chain without additional logic cells (from some flash memory elsewhere on the device). Alternatively, non-volatile memory cells could be used to store them. The number of parameters is 33 and 65 kbit and thus $0.05\,\text{mm}^2$ (264 kGE) and $0.10\,\text{mm}^2$ (520 kGE) for the 16×16 and 32×32 network, respectively.

Looking at the more detailed area breakdown in Tbl. III, we can see that there is a massive reduction when fixing the weights before synthesis. Clearly, this eliminates all the XNOR operations which become either inverters or wires, and even some of the inverters can now be shared among all units having this particular input value in their receptive field. However, based on the estimates described in Sec. III-A, this cannot explain all the savings. Additional cells can be saved through the reuse of identical partial results, which not only can occur randomly but must occur frequently. For example, consider 16 parallel popcount units summing over 8 values each. We can

TABLE IV
ENERGY AND LEAKAGE BREAKDOWN FOR THE 16×16 NETWORK

	var. weights		fixed weights	
layer	energy/img [pJ]	leakage	energy/img [pJ]	leakage
1	38 (1.6%)	68 µW	9 (0.5%)	8 µW
2	806 (33.7%)	547 µW	478 (28.5%)	152 µW
3	1440 (60.2%)	310 µW	1037 (61.9%)	163 µW
4	107 (4.5%)	20 µW	151 (9.0%)	7 µW

TABLE V
COMPARISON WITH STATE-OF-THE-ART APPROACHES FOR DEEP-LEARNING NEAR SENSOR PROCESSING

Approach	[15]	[19]	XNOR-POP [21]	YodaNN [8]	This Work
Description	Analog switched-capacitor matrix-mult	Cluster of 256 BNN Digital Engines	In-memory XNOR Engines	Binary-weight digital accelerator	Combinational BNN with fixed weights
Technology	40 nm	14 nm	32 nm (logic)	65 nm	22 nm SOI
Peak Performance	2 GOp/s	16 TOp/s	5.2 TOp/s	1.5 TOp/s	91.12 TOp/s
Power Consumption	228 µW @ 1 V	–	237 mW	25 mW @ 0.6 V	135 mW (dyn) @ 0.65 V
Peak Energy-efficiency	8.77 TOp/s/W	–	22.2 TOp/s/W	61.2 TOp/s/W	672.6 TOp/s/W
Area	–	–	2.24 mm^2	1.9 mm^2(1.33 MGE)	0.46 mm^2(2.32 MGE)

split the value into 4 groups with 2 values each. Two binary values can generate $2^2 = 4$ output combinations. Since we have 16 units of which each will need one of the combinations, they will on average be reused 4 times. This is only possible with fixed weights, otherwise the values to reuse would have to be multiplexed, thereby loosing all the savings.

Generally, we can observe that these already small networks for low-resolution images require a sizable amount of area, such that more advanced ad-hoc synthesis tools exploiting the sharing of weights and intermediate results are needed.

C. Energy Efficiency Evaluations

We have performed post-synthesis power simulations using 100 randomly selected real images from the dataset as stimuli. The results are also reported in Tbl. II while a detailed per-layer breakdown is shown in Tbl. IV. We see that the 32×32 model has lower energy-efficiency and higher latency with fixed weights, opposed to the smaller model. We attribute this to the optimization towards minimal area without a timing constraint, which favors weak drivers and high-fanout nets with accordingly long transition times. We have observed that particularly many cells are driven by signals corresponding to the feature maps between layers, and that fixed-weight circuits are more affected by this due to the significantly higher input capacitance of FAs relative to XOR gates.

When heavily duty-cycling a device, leakage can become a problem. In this case, we see 945 µW and 331 µW of leakage power, which might be significant enough in case of low utilization to require mitigation through power-gating or using HVT cells. Generally, voltage scaling can also be applied, not only reducing leakage, but also active power dissipation. The throughput we observe in the range of 50 Mframe/s is far in excess of what is meaningful for most applications. Thus aggressive voltage scaling, power gating and the reverse body biasing available in this FD-SOI technology should be optimally combined to reach the minimum energy point where leakage and dynamic power are equal while the supply is ON.

A comparison with state-of-the-art approaches is reported in Tbl. V. The energy efficiency numbers of our approach are in the order of 10× higher than those of the next competitor YodaNN [8]. However, they are fundamentally different in the sense that YodaNN (a) runs the more complex binary weight networks, (b) requires additional off-chip memory for the weights and intermediate results, (c) can run large networks with a fixed-size accelerator, and (d) is in an older technology but doing aggressive voltage scaling.

We expect the energy values to be highly dependent on the input data, since energy is consumed only when values toggle. While a single pixel toggling at the input might affect many values later in the network, it has been shown that rather the opposite effect can be seen: changes at the input tend to vanish deeper into the network [24]. A purely combinational implementation fully leverages this and BNNs naturally have a threshold that keeps small changes from propagating and might thus perform even better for many real-world applications.

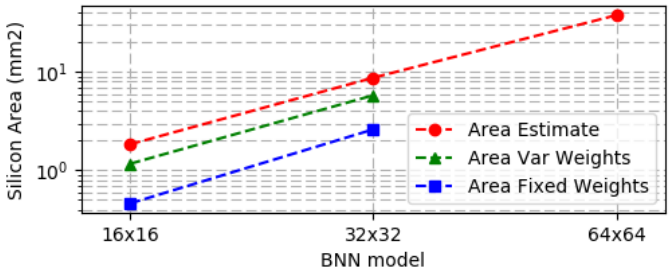

Fig. 3. Silicon area estimation (in red) and measurements with variable (green) and fixed (blue) weights of three BNNs featuring a model complexity which scales depending on the imager resolution. The area occupation of the 64×64 model is not reported because the synthesis tool is not able to handle such a complex and large design.

D. Scaling to Larger Networks

Our results show an area requirement in the range of 2.05 to 2.46 GE/Op and an average 1.9 fJ/Op. Scaling this up to 0.5 cm^2 (250 MGE) of silicon and an energy consumption of only 210 nJ/img, we could map networks of around 110 MOp/img—this is already more than optimized high-quality ImageNet classification networks such as ShuffleNets require [25].

Fig. 3 shows the estimation and measurements of the silicon area corresponding to the synthesized BNNs for fixed and variable weights. We also consider a model with a larger 64×64 input imager receptive field and a higher complexity (5 convolutional and 2 fully-connected layers, 23.05 GOp/img). Such a model presents is more accurate on the considered classification task (73.6%) but current synthesis tool cannot handle the high complexity of the design, using in excess of 256 GB of memory. When estimating the area occupancy, the 64×64 BNNs result to be 4.3× larger than the area estimated for the 32×32 model. A direct optimization of such large designs is out of scope of today's EDA tools, clearly showing the need for specialized design automation tools for BNNs.

V. CONCLUSION

We have presented a purely combinational design and synthesis of BNNs for near-sensor processing. Our results demonstrate the suitability and the energy efficiency benefits of the proposed solution, fitting on a silicon area of 2.61 mm^2 when considering a BNN model with 32×32 binary input data and weight parameters fixed at design time. Our study also highlighted the need for novel synthesis tools able to deal with very large and complex network designs, that are not easily handled by current tools.

REFERENCES

[1] M. Alioto, *Enabling the Internet of Things: From Integrated Circuits to Integrated Systems*. Springer, 2017.

[2] M. Rusci, D. Rossi *et al.*, "An event-driven ultra-low-power smart visual sensor," *IEEE Sensors Journal*, vol. 16, no. 13, pp. 5344–5353, 2016.

[3] Á. Rodríguez-Vázquez, R. Carmona-Galán *et al.*, "In the quest of vision-sensors-on-chip: Pre-processing sensors for data reduction," *Electronic Imaging*, vol. 2017, no. 11, pp. 96–101, 2017.

[4] R. LiKamWa, Y. Hou *et al.*, "Redeye: analog convnet image sensor architecture for continuous mobile vision," in *Proc. IEEE ISCA*, 2016, pp. 255–266.

[5] S. Zhang, M. Kang *et al.*, "Reducing the energy cost of inference via in-sensor information processing," *arXiv:1607.00667*, 2016.

[6] J. Fernández-Berni, R. Carmona-Galán *et al.*, "Focal-plane sensing-processing: A power-efficient approach for the implementation of privacy-aware networked visual sensors," *Sensors*, vol. 14, no. 8, pp. 15 203–15 226, 2014.

[7] M. Gottardi, N. Massari, and S. A. Jawed, "A 100μ w 128$\times$64 pixels contrast-based asynchronous binary vision sensor for sensor networks applications," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 5, pp. 1582–1592, 2009.

[8] R. Andri, L. Cavigelli *et al.*, "Yodann: An architecture for ultra-low power binary-weight cnn acceleration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2017.

[9] M. Courbariaux, I. Hubara *et al.*, "Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1," *arXiv:1602.02830*, 2016.

[10] M. Rastegari, V. Ordonez *et al.*, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *Proc. ECCV*. Springer, 2016, pp. 525–542.

[11] Y. Umuroglu, N. J. Fraser *et al.*, "Finn: A framework for fast, scalable binarized neural network inference," in *Proc. ACM/SIGDA FPGA*, 2017, pp. 65–74.

[12] J. Choi, S. Park *et al.*, "A 3.4-μw object-adaptive cmos image sensor with embedded feature extraction algorithm for motion-triggered object-of-interest imaging," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 1, pp. 289–300, 2014.

[13] G. Kim, M. Barangi *et al.*, "A 467nw cmos visual motion sensor with temporal averaging and pixel aggregation," in *Proc. IEEE ISSCC*, 2013, pp. 480–481.

[14] H. G. Chen, S. Jayasuriya *et al.*, "Asp vision: Optically computing the first layer of convolutional neural networks using angle sensitive pixels," in *Proc. IEEE CVPR*, 2016, pp. 903–912.

[15] E. H. Lee and S. S. Wong, "Analysis and design of a passive switched-capacitor matrix multiplier for approximate computing," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 261–271, 2017.

[16] Z. Du, R. Fasthuber *et al.*, "Shidiannao: Shifting vision processing closer to the sensor," in *ACM SIGARCH Computer Architecture News*, vol. 43, no. 3, 2015, pp. 92–104.

[17] L. Cavigelli and L. Benini, "Origami: A 803-gop/s/w convolutional network accelerator," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 11, pp. 2461–2475, 2017.

[18] V. Sze, Y.-H. Chen *et al.*, "Efficient processing of deep neural networks: A tutorial and survey," *arXiv:1703.09039*, 2017.

[19] E. Nurvitadhi, D. Sheffield *et al.*, "Accelerating binarized neural networks: Comparison of fpga, cpu, gpu, and asic," in *Proc. FPT*, 2016, pp. 77–84.

[20] K. Ando, K. Ueyoshi *et al.*, "Brein memory: A 13-layer 4.2 k neuron/0.8 m synapse binary/ternary reconfigurable in-memory deep neural network accelerator in 65 nm cmos," in *Proc. VLSI Symposium*, 2017.

[21] L. Jiang, M. Kim *et al.*, "Xnor-pop: A processing-in-memory architecture for binary convolutional neural networks in wide-io2 drams," in *Proc. IEEE/ACM ISLPED*, 2017.

[22] "The traffic surveillance workshop and challenge 2017 (tswc- 2017)," 2017, MIO-TCD: MIOvision Traffic Camera Dataset. [Online]. Available: http://podoce.dinf.usherbrooke.ca

[23] S. Jayasuriya, O. Gallo *et al.*, "Deep learning with energy-efficient binary gradient cameras," *arXiv:1612.00986*, 2016.

[24] L. Cavigelli, P. Degen, and L. Benini, "Cbinfer: Change-based inference for convolutional neural networks on video data," *arXiv:1704.04313*, 2017.

[25] X. Zhang, X. Zhou *et al.*, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," *arXiv:1707.01083*, 2017.