A 34-FPS 698-GOP/s/W Binarized Deep Neural Network-based Natural Scene Text Interpretation Accelerator for Mobile Edge Computing

Yixing Li, Zichuan Liu, *Student Member*, Wenye Liu, *Student Member*, Yu Jiang, *Member, IEEE*, Yongliang Wang, Wang Ling Goh, *Senior Member*, Hao Yu, *Senior Member*, Fengbo Ren, *Member*

***Abstract*—The scene text interpretation is a critical part of natural scene interpretation. Currently, most of the existing work is based on high-end GPU implementation, which is commonly used on the server side. However, in IoT application scenarios, the communication overhead from the edge device to the server is quite large, which sometimes even dominates the total processing time. Hence, the edge-computing oriented design is needed to solve this problem. In this paper, we present an architectural design and implementation of a natural scene text interpretation (NSTI) accelerator, which can classify and localize the text region on pixel-level efficiently in real-time on mobile devices. To target the real-time and low-latency processing, the Binary Convolutional Encoder-decoder Network (B-CEDNet) is adopted as the core architecture to enable massive parallelism due to its binary feature. Massively parallelized computations and a highly pipelined data flow control enhance its latency and throughput performance. In addition, all the binarized intermediate results and parameters are stored on chip to eliminate the power consumption and latency overhead of the off-chip communication. The NSTI accelerator is implemented in a 40nm CMOS technology, which can process scene text images (size of 128×32) at 34 fps and latency of 40 ms for pixelwise interpretation with the pixelwise classification accuracy over 90% on ICDAR-03 and ICDAR-13 dataset. The real energy-efficiency is 698 GOP/s/W and the peak energy-efficiency can get up to 7825 GOP/s/W. The proposed accelerator is 7× more energy efficient than its optimized GPU-based implementation counterpart, while maintaining a real-time throughput with latency of 40 ms.**

***Index Terms*—Application specific integrated circuits, Mobile applications, Neural network hardware, Real-time systems**

Manuscript received April 30, 2018; revised August 28, 2018; accepted September 21, 2018. Arizona State University's work is supported by NSF grant IIS/CPS-1652038.

Y. Li and F. Ren are with School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, Tempe, AZ 85281, USA (e-mail: yixingli/renfengbo@asu.edu).

Z. Liu, W. Liu, Y. Jiang and W. Goh are with School of Electrical Electronic Engineering, Nanyang Technological University, Singapore, 639798. (e-mail: zliu016/wliu015/yjiang017@e.ntu.edu.sg, ewlgoh@ntu.edu.sg).

Y. Wang is with Verisilicon Corp., Shanghai, 201203, China. (e-mail: lianger_wang@sina.com)

H. Yu is with the Department of Electrical and Electronic Engineering, Southern University of Science and Technology of China, Guangdong, 518055, China. (e-mail: yuh3@sustc.edu.cn)

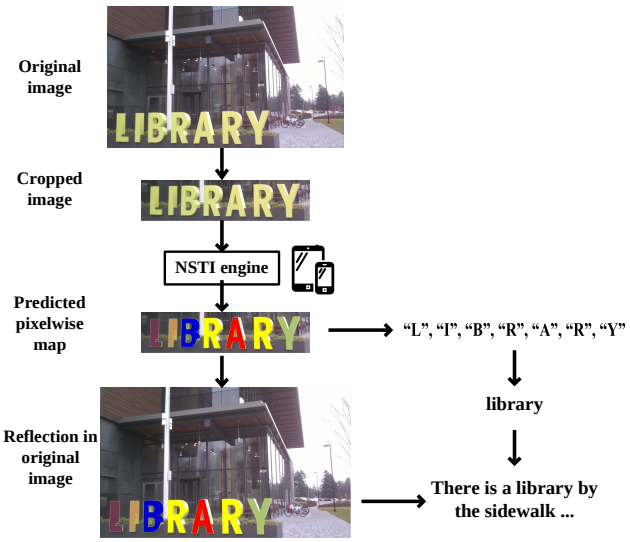


Fig. 1. Natural scene text interpretation system.

I. Introduction

THE scene text interpretation is a critical part of natural scene interpretation, since the text probably contains more explicit information than the natural object. For instance, in a driving scenario, besides the standard road signs, text-based guide signs are essential in predicting the demand of lane changing. In a walking scenario, the name of the building or the store can help to make precise and reliable localization.

Conventionally, text recognition has been vastly investigated for document images [1]. However, in the natural scene, the background is much more complicated than that of the document images, which makes the scene text recognition become a more challenging task. With the recent development in neural networks and deep learning [2] [3], the accuracy of natural scene text recognition has outperformed the traditional feature selection methods by using features selected automatically [4] [5]. The related work can be categorized as character-level based and word-level based solutions. The character-level based solutions [6] [7] detect and recognize character

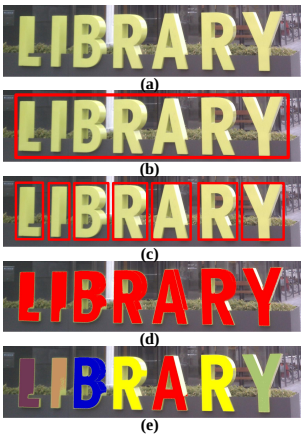


Fig. 2. Comparison of different levels of natural scene text processing.

one at a time. Its front-end is a sliding window approach for character proposals, which makes it suffer from the processing time. The word-level based solution [8] requests large fully-connected layer to generate the probability for thousands of word classes, which place a heavy burden on memory access. The shared limitation of either character-level [6] [7] or word-level based [8] solutions is that their architecture is not capable of achieving a low-latency performance. In [9], it performs one-shot text interpretation with a binary convolutional encoder-decoder network (B-CEDNet). Since most of the computation in B-CEDNet are bitwise operations, it opens a new opportunity for hardware acceleration.

However, all the previous work mentioned above is implemented by high-end GPUs (such as Nvidia Titan X). The power-hungry high-end GPUs are not able to be deployed on energy-constrained mobile devices. If GPUs are deployed on the server side, the communication overhead from a client to a cloud server is quite large, which sometimes even dominates the total processing time. However, long latency is not tolerant in augmented reality (AR) applications. If one chooses to use low-power oriented GPUs, such as Nvidia Tegra X1, on the power constrained edge devices, it will get 20× performance (in terms of Flops) degradation compared with the Nvidia Titan X GPU [10]. Considering the performance degradation factor, the frame rate in [9] will drop from 200 fps to 20 fps when it is mapped onto a Tegra X1. As such, it cannot maintain a real-time throughput on a lower-power GPU. In addition, the power consumption of a Tegra X1 is 6W [10], which is still too power hungry for a smartphone. Hence, an edge-computing oriented design is needed to solve this problem.

In this paper, in order to target a low-latency and real-time processor for energy-efficient natural scene text processing on mobile devices, we propose an ASIC B-CEDNet-based natural scene text interpretation (NSTI) accelerator. As shown in Fig. 1, the processor takes the cropped natural scene image as the input and outputs a map of pixelwise classification results with the same size as input. In comparison with generating a bounding box for each character or the entire word (as shown in Fig. 2 (b) and (c)), the pixelwise classification output (in Fig. 2 (a)) shows morphological boundary, which is much more user-friendly in AR applications. Compared with binary classification results for the text and non-text regions in Fig. 2 (d), the proposed processor can identify different characters in a one-shot prediction. In addition, with the localization, morphological and categorized information, it largely alleviates the workload for the back-end word-level prediction and even scene description as shown in Fig. 1. The bitwise operation dominated computation in B-CEDNet enables massive parallelism of multiply-add operations (MACs) in the proposed processor. The binarized parameters and intermediate results are fully mapped on chip to eliminate the communication cost (regarding power consumption) instead of loading them from off-chip memory.

The rest of the paper is organized as follows: Section II discusses the Convolutional encoder-decoder network (CEDNet) and its binary counterpart B-CEDNet from the algorithm perspective. Then Section III presents the hierarchical design of the ASIC NSTI accelerator. Section IV illustrates the implementation details of NSTI accelerator. Section V discusses the performance results of the accelerator. Finally, the last Section VI concludes the paper.

II. Preliminary

In this section, Section A discusses the CEDNet architecture for pixelwise interpretation from the algorithm perspective. Then Section B introduces its binary counterpart, the B-CEDNet architecture. Section B emphasizes the differences between two architectures and explains how its binary feature brings new opportunity for the hardware acceleration.

A. Convolutional encoder-decoder network (CEDNet)

Conventionally convolutional neural networks (CNNs) are used for image classification tasks [11]–[13]. Generally, they are composed of convolutional layers, pooling layers, and fully-connected layers [16]. To perform image classification, the network only generates one prediction for the entire image. Therefore, CNNs cannot be directly deployed for the pixelwise interpretation of images. In Fig. 3, the convolutional encoder-decoder network (CEDNet) is proposed in [14] for the multi-class pixelwise classification. A CEDNet takes the scene text images as input. The body of the network can be divided into the encoder part and decoder part. The output of the CEDNet is a salience map $S \in R^{W_I \times H_I \times C}$, which contains the probability information of each pixel over C categories (including one background class), where C is 27 in our case (characters are case insensitive). The encoder part is a stack of encoder blocks, while the decoder is a stack of decoder blocks. Each encoder block contains a convolutional (Conv) layer, a pooling layer (PL), a batch normalization (BN) layer and a rectified linear unit (ReLU) layer. The convolutional layer applies convolutional operations on input feature map $a_{k-1} \in R^{W_{k-1} \times H_{k-1} \times D_{k-1}}$ with trainable weight matrix $w_k \in R^{w_k \times h_k \times D_k \times D_k}$, where the subscript k indicates the k^{th} block. The convolutional operations can be formulated as

$$s_k(x,y,z) = \sum_{i=1}^{w_k} \sum_{j=1}^{h_k} \sum_{l=1}^{D_{k-1}} w_k(i,j,l,z) * a_{k-1}(i+x-1, j+y-1, l), \quad (1)$$

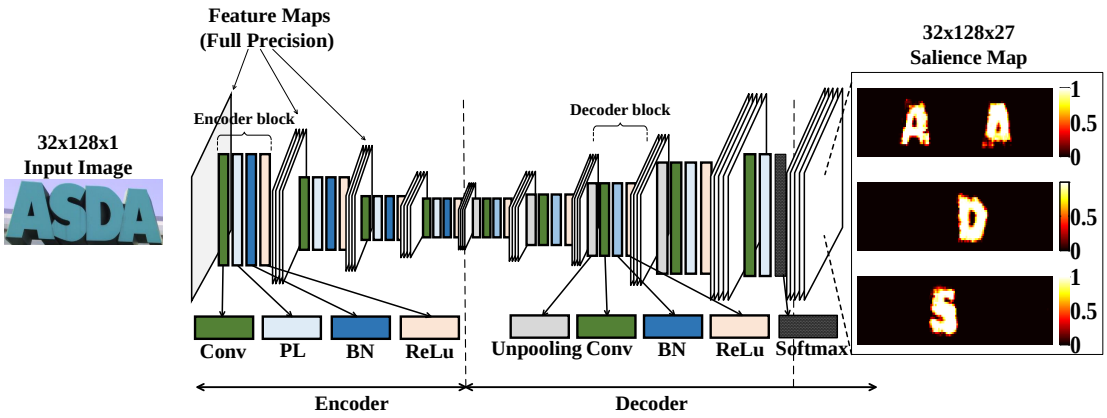


Fig. 3. Architecture of the convolutional encoder-decoder network (CEDNet).

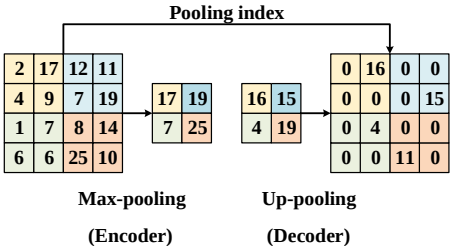


Fig. 4. Pooling and up-pooling layers.

where $s_k \in R^{W_k \times H_k \times D_k}$ is the output of k^{th} Conv layer. Equation (1) shows that the computation of s_k along three dimensions has no data dependence, which can be highly paralleled in an ASIC implementation. The Conv layer is intended to extract high-level features, which are critical for the pixelwise classification. In the PL layer, it pools out the critical information and eliminates the non-critical one. The PL layer can perform either max pooling or average pooling [11]. A max-pooling layer is shown in Fig. 4, it pools out the maximum value in each 2×2 window. By introducing the pooling layer, the size of the feature map is shrinking as the network goes deeper. The BN layer is mainly used for accelerating training process [15]. So in the inference stage, the BN layer is also applied to match the training process forming a stable distribution of the activations (a_k). The output of k^{th} BN layer is represented as follows:

$$a_k(x,y,z) = \frac{s_k(x,y,z) - \mu(x,y,z)}{\sqrt{\sigma^2(x,y,z) + \epsilon}} \gamma(x,y,z) + \beta(x,y,z), \quad (2)$$

where μ and σ^2 is the mean and variance over the mini-batch training data, while γ and β are trainable scaling factors. The activation function is a nonlinear transformation. The most commonly used activation function [11], ReLU function is represented as

$$a_k(x,y,z) = \begin{cases} 0, a_k(x,y,z) \leq 0 \\ a_k(x,y,z), a_k(x,y,z) \geq 0. \end{cases} \quad (3)$$

The entire encoder part is similar to a CNN without fully-connected layers.

Since the output salience map is desired to be the same size as the input, in the decoder part, each decoder block substitutes the pooling layer with the up-pooling layer. As shown in Fig. 4, the up-pooling (UPL) layer pools back the maximum value to the same index in corresponding max-pooling layer. As such, the output salience map can represent the same localized information as the input. In order to predict the pixelwise character appearance probability, the output block replaces the ReLU function with softmax function. As shown in the rightmost part of Fig. 3, it only plots the salient map slices for character "A", "D" and "S". The lighter color code means higher confidence level and vice versa. The CEDNet architecture can enable highly parallelized MAC computing inside every encoder or decoder block. It eliminates both the run-time bottle stage in sliding window-based proposal and the computation-intensive fully-connected layer.

B. Binary convolutional encoder-decoder network (B-CEDNet)

Even though the mobile devices are getting more and more computing power, it is still hard to deploy full-precision CNNs for efficient computing on mobile edge devices. Since the CNN architecture is proved to have huge redundancy [16], different methods [17]–[21] have been proposed to reduce the computation complexity and/or alleviate the memory access issues. Some approaches [20] focus on minimizing total number of parameters, which mainly alleviate the memory access issues. While other approaches [17]–[19], [21] reduce the precision of weights and activations, which can both reduce the computation complexity and alleviate the memory access issues. Among these approaches, binarization [18], [19], [22] can push the weights and activations to be represented in binary format $w_k^b \in \{0,1\}^{W_k \times H_k \times D_k \times D_k}$, and $a_{k-1}^b \in \{0,1\}^{W_{k-1} \times H_{k-1} \times D_{k-1}}$. It can achieve up to $32\times$ memory saving and converting the convolution operations to bitwise XNOR operations for much more efficient computing. It has been proved in [9], binarization approach can be adopted in CEDNet to build a binary convolutional encoder-decoder

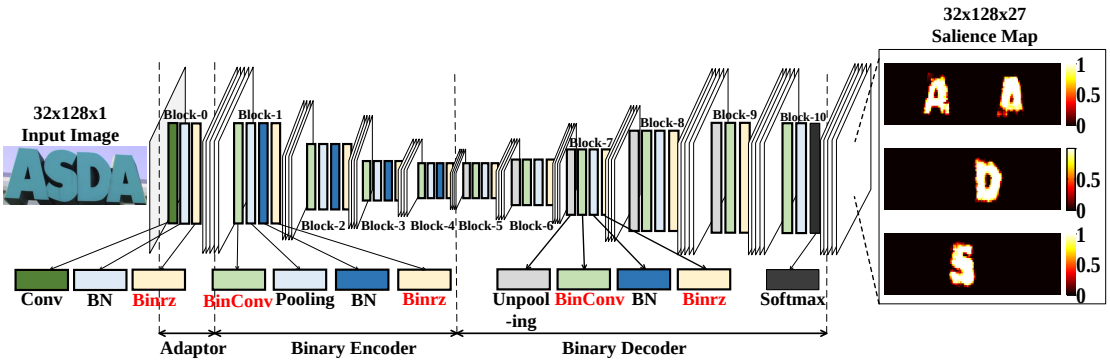


Fig. 5. Architecture of the binary convolutional encoder-decoder network (B-CEDNet).

network (B-CEDNet as shown in Fig. 5) for pixelwise text classification with merely no accuracy drop.

In the B-CEDNet, it replaces the Conv layer and ReLU layer with the binary convolutional layer (BinConv layer) and Binarization layer (Binrz layer), respectively. The equation for the BinConv layer and Binrz layer is shown in (4) and (5), respectively.

$$s_k(x,y,z) = \sum_{i=1}^{w_k}\sum_{j=1}^{h_k}\sum_{l=1}^{D_{k-1}} \sim (w_k^b(i,j,l,z) \oplus a_{k-1}^b(i+x-1, j+y-1, l)) \tag{4}$$

$$a_k^b(x,y,z) = \begin{cases} -1, a_k(x,y,z) \le 0 \\ +1, a_k(x,y,z) \ge 0 \end{cases} \tag{5}$$

The most costly computation, full-precision multiplication, is now converted into the hardware-friendly bitwise XNOR operation. For GPU implementation, one MAC module can process 32-bit bitwise XNOR instead of one 32-bit multiply-add operation. For FPGA implementation, the BinConv layer is no longer needed to be implemented in DSP slices. Massive LUTs can be used for efficiently implementing bitwise operations. For ASIC implementation, it is flexible enough to build tailored computing units for a BinConv layer with tree-like bitwise XNOR and bit-count logics. With simplified basic computing units, it is able to map massive computing units to target a high system throughput.

B-CEDNet has brought new opportunity in energy-efficient edge-computing applications. Compared with power-hungry GPU-based solutions and overhead of routing in FPGA-based solutions, a tailored ASIC solution for B-CEDNet can be the most energy efficient solution with high throughput performance. It is able to satisfy the need for real-time and low-latency processing in power-constrained edge-computing device for scene text interpretation.

III. Architecture Design

Most existing ASIC/FPGA-based CNN accelerators are only compatible with encoder blocks (down-sampling trend) for image classification, recognition and detection tasks [23], [24]. While some optimize designs for decoder blocks (up-sampling trend) for super resolution applications [25]. The proposed architecture is customized for the convolutional encoder-decoder network. The Fig. 6 shows the ASIC architecture of the proposed Natural Scene Text Interpretation (NSTI) accelerator. The NSTI accelerator takes the scene text image from the off-chip DRAM as the input. Then it is processed through computing blocks in a streaming manner. The computing blocks, Block-0 to Block-10, are corresponding to 11 blocks in Fig. 5. Each computing block is built upon a processing element (PE) array, as shown in the right half of the Fig. 6. Each PE performs the operations of convolution, max-pooling/un-pooling, activation function and batch normalization. The spatial parallelism of the NSTI accelerator is reflected on the block level, PE level and sub-PE level. The temporal parallelism is reflected in highly pipelined steaming data flow. Both massive spatial parallelism and temporal parallelism enable high throughput performance of the proposed NSTI accelerator. Reduction in computation complexity to bit-level operations benefits in power saving. Storing all the weights (w^b) and intermediate results (a^b) on chip to minimize off-chip communication gives extra credits to energy saving. In this section, Section A illustrates the details in computing blocks hierarchically. Section B demonstrates the design consideration for the memory. The dataflow control is then presented in Section C.

A. Processing elements (PEs)

Each computing block in Fig. 6 performs the computation corresponding to Fig. 5. Therefore, Block-1 to Block-4 and Block-5 to Block-8 are identical, respectively. Although the functions vary among these blocks, the structure inside each block is the same as shown in Fig. 6. In each block, PE arrays take the feature map a_{k-1}^b from previous layers and weight w_k^b values from its local memory (ROM) as the inputs, and output the feature map a_k^b of the current layer. All the PEs in the same block work simultaneously. The differences among these blocks exist in their processing elements (PEs).

The PEs of encoder and decoder are shown in Fig. 7. The PE of the encoder in Fig. 7(a) has 4 BinConv kernels, a PL kernel and a BN-Binrz kernel, while the PE of the decoder in Fig. 7(b) has a BinConv kernel, an UPL (un-pooling) kernel and a BN-Binrz kernel. For the convenience of the ASIC implementation, we group the UPL layer in block $k+1$ to

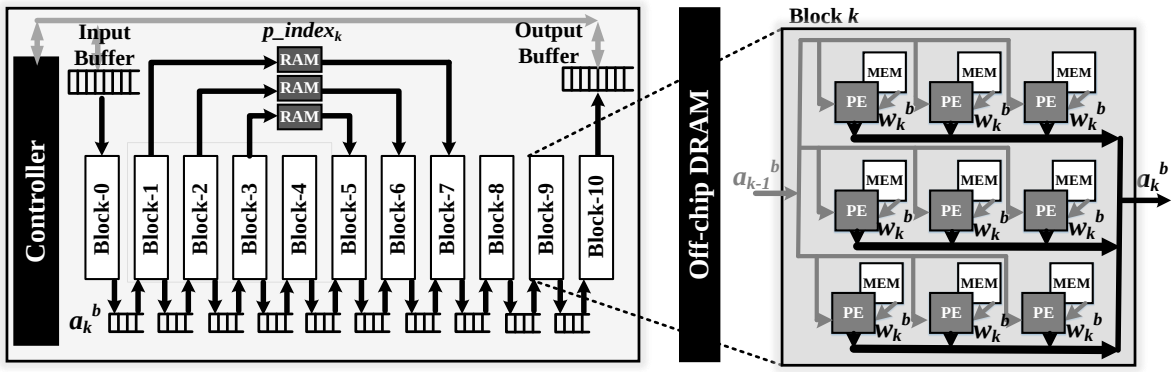
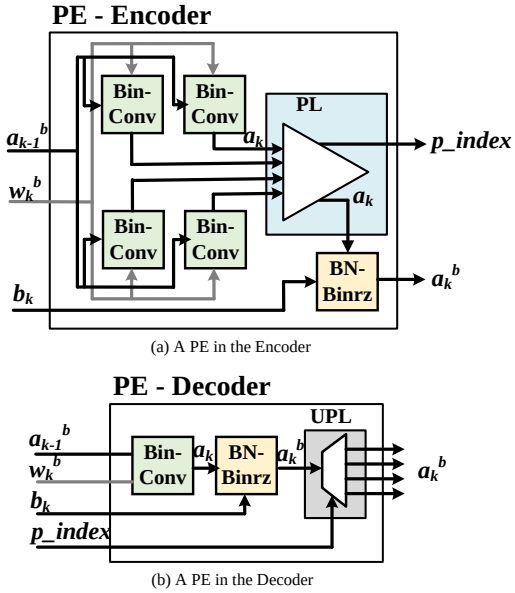


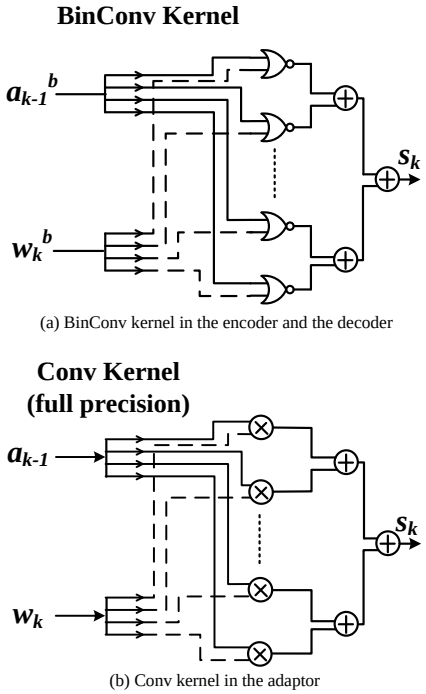
Fig. 6. Architecture of the convolutional encoder-decoder network (CEDNet).



(a) A PE in the Encoder

(b) A PE in the Decoder

Fig. 7. Processing elements (PEs).



(a) BinConv kernel in the encoder and the decoder

(b) Conv kernel in the adaptor

Fig. 8. BinConv kernel and Conv kernel.

block k in building the computing block. Therefore, in each decoder PE, it starts with a BinConv kernel and ends with an UPL kernel. If BinConv kernels in an encoder PE are substituted with Conv kernels, it becomes a PE for the adapter. BinConv kernels in Fig. 7(a) and Conv kernels of the adapter are both implemented in a tree-like structure as shown in Fig. 8. A Conv kernel has a floating-point operation on each node, while a BinConv kernel performs bit-level XNOR and bit-count. They both are implemented by pure combinational logics. In each Conv/BinConv kernel, it computes one $s_k(x, y, z)$ at a time, that is to say, the parallelism factor in terms of number of operations is $w_k \times h_k \times D_k$. The computation of the BN and Binrz layer can be simplified as a threshold function [26], which can be implemented by a single 2-input comparator, denoted as BN-Binrz kernel in Fig. 7. The PL kernel is implemented with a 4-input comparator, which also encodes the index of the maximum value in pooling region. The pooled out value and its index are stored in buffer. Then feed them into the UPL kernel in its symmetric decoder block as shown in Fig. 6. The DEMUX in the UPL kernel of Fig. 7(b) writes back the (pooled maximum) value with the index information to the right location in the RAM. The up-pooled data in the buffer serves as the input of next decoder block.

B. Memory design

In DL-based ASIC designs [27]–[29], the communication to the off-chip DRAM is very power-intensive. The binary feature of B-CEDNet enable us to store all the weights (w^b) and intermediate results (a^b) on chip to minimize off-chip communication for energy saving. As shown in Table 1, the first and second column indicates the memory size of weight values in the non-binary case (CEDNet) and binary case (B-CEDNet), respectively. The total memory size of weights in the B-CEDNet has 30× saving, comparing with the non-binary one. The ideal memory saving results from converting full-precision network (32 bits) to binarized-weight network should be 32×. Since the first layer still has non-binary weights, the real compression ratio is a little bit less than the ideal case. 2,144 KB distributed ROMs are built to store all weight values,

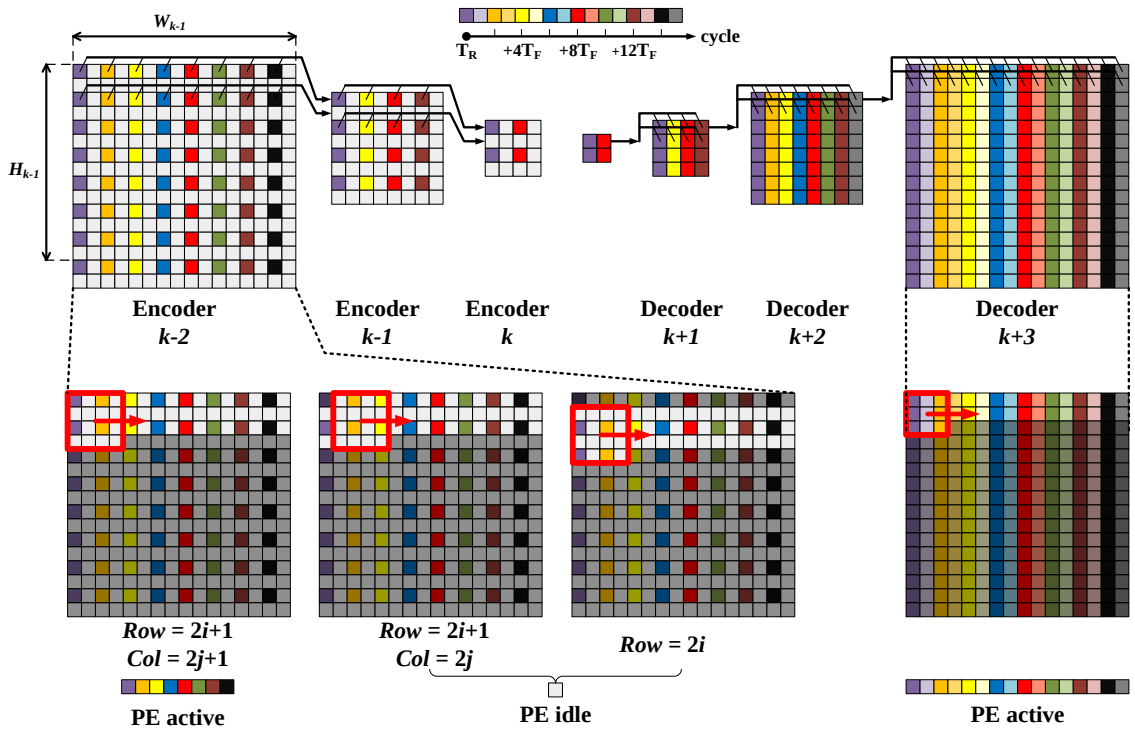


Fig. 9. Data flow control between blocks.

as shown in Fig. 6. In a PE array, each PE has its local ROM attached. This can alleviate the routing issue in the bottom-up design flow. There are total 423 KB binarized intermediate results a^b between blocks. Since the size of a^b is relatively small, synthesized shift registers are used to buffer a^b. This can enable global voltage scaling with the core computing part to get a more energy efficient point in chip testing stage. 172 KB Block SRAMs (hard macros) are used between the encoder block (Block-1, -2 and -3) and its symmetric decoder block (Block-5, -6 and -7) to buffer the pooling index. For the innermost block, Block-4, we directly up-pool the outputs of the max-pooling layer. As such, there is no need to store the pooling index of block 4.

C. Dataflow control

In Fig. 9, it shows the data flow control across different blocks. Since all the layers share the same depth, we can simplify it as a 2-D B-CEDNet in the following discussion. In a BinConv layer, the filter size of the weight matrix is 3×3 and the stride is equal to 1. While in a PL layer, the filter size is 2×2 and the stride is equal to 2. In an encoder block, since we have group 4 BinConv kernels and its corresponding PL kernel into one PE (as shown in Fig. 7(a)), the size of sliding window should be 4×4. In the decoder case, the size of the sliding window is 3×3 with only one BinConv PE.

The color code is in Fig. 9 indicates the location of the sliding window regarding to the clock cycle. Each sliding window is located by the pixel of its upper-left corner. In the first pixel of each row, the reference time is defined as T_R, where R is the row index. We keep reusing the same feature map region (where the red sliding window is) at time T_F and process it with different weight values. In order to maximize the data reuse, we buffer $(F-1)\times W_{k-1}+F$ pixels at a time, where the size of the sliding window is $F\times F$. The buffered data in Fig. 9 are in relatively high brightness. Feature map reuse helps to reduce the frequency of fetching new (feature map) data, which will result in energy saving. After T_F cycles, the window slides to the right with the stride equal to 1. All the pixels with non-white color codes indicate the PEs are active. In the active mode, the PEs read in new data from the previous block, execute the computation and write the processed data to the next block. Since PEs are implemented by combinational logics, once the buffered data is ready, the current block produces valid results simultaneously. All the other pixels in the white color code indicate the PEs are idle, where the PEs only reads new data into the buffer. As shown in Fig. 9, the PEs are active in 1/2, 1/4 and 1/8 of total time in the encoder block $k-2$, $k-1$ and k, respectively. In order to maximize the utilization of PEs (active time ratio of PEs), we assign 4×, 2× and 1× number of PEs, accordingly. Similarly, in the decoder block $k+1$, $k+2$ and $k+3$, number of PEs increases as 1×, 2× and 4×. Therefore, the proposed data flow control makes all the computing blocks work in a highly pipelined fashion, which enhances the throughput performance of the NSTI accelerator.

IV. Chip Implementation

The configuration of the B-CEDNet is the same as [9]. The chip summary is shown in Table II. The NSTI accelerator is implemented in a 40nm 1p10m process using a standard-cell-based design flow. The RTL code is synthesized in Synopsys Design Compiler (DC). To achieve the target throughput, a clock period of 33.33 ns (30 MHz) evaluated at the worst-case process, voltage, and temperature (PVT) corner is targeted

TABLE I
MEMORY SUMMARY (UNIT:KB)

Block	w	w^b	a^b	p_index
Block-0	22	22	$<<1$	N/A
Block-1	2,008	71	50	131
Block-2	9,008	289	16	33
Block-3	9,008	289	8	8
Block-4	9,008	289	4	N/A
Block-5	9,008	289	1	N/A
Block-6	9,008	289	8	N/A
Block-7	9,008	289	16	N/A
Block-8	9,008	289	16	N/A
Block-9	9,008	289	16	N/A
Block-10	54	9	N/A	N/A
Total	66,126	2,144	423	172

TABLE II
CHIP SUMMARY

Symbol	Quantity
Technology	40nm 1p10m CMOS
Transistor flavor	HVT 92.8%, SVT 7.2%
Gate count	2811 kGates
I/Os	Digital: 13/27, Power: 33
Core VDD	0.9 V
I/O VDD	1.8 V
Core size	12.7 mm^2

throughout the chip implementation. Taking into account the overhead to be introduced by the subsequent physical design, a 40% timing slack is used during the synthesis. Specifically, the NSTI accelerator is synthesized with a target clock frequency of 30/(1-40%) = 50 MHz. To reduce leakage power, the NSTI accelerator is first synthesized using high-threshold (HVT) standard cells only. Then, standard-threshold (SVT) standard cells are selectively inserted into the critical paths for timing improvement. This is carried out by switching on the leakage optimization tool in DC. Overall, the chip occupies a core area of 12.7 mm^2 with an aspect ratio of 0.52 and integrates 2811 kGates. The layout of the accelerator is shown in Fig. 10. The computing blocks and the buffers for intermediate results are colored in red. The pooling indexes (shown as RAM in Fig. 6) have consumed a memory size of 172 KB. To reduce area cost, RAMs are realized by dual-port SRAM hard macros, which are in blue. The 2,144 KB local ROMs for weights are distributed in each block, colored in yellow. For the leakage reduction purpose, HVT devices are used in 92.8% of the logic cells. The chip has 13 digital inputs, 27 digital outputs, and 33 power pads supply core and I/O power domain. The I/O domain has a constant supply voltage of 1.8 V, and the logic and memory domain both have a normal supply voltage of 0.9 V. The post-layout simulation is performed to verify the functionality. The die photo is shown in Fig. 11.

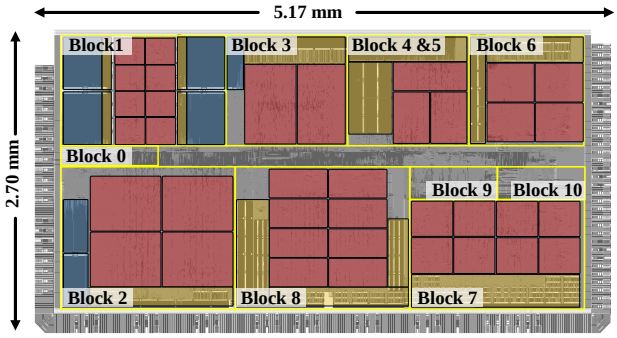


Fig. 10. The layout of NSTI accelerator.

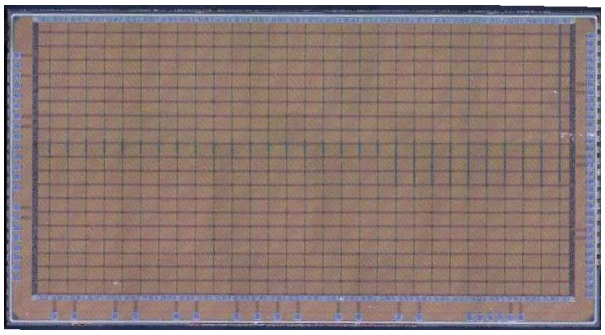


Fig. 11. The die photo of NSTI accelerator

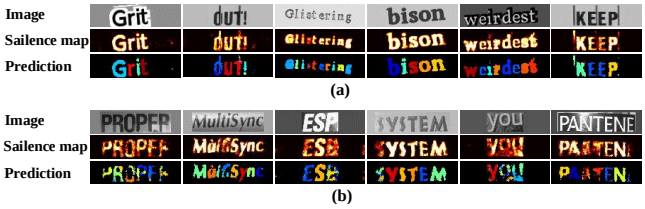


Fig. 12. Visualization of NSTI accelerator output

V. EXPERIMENT RESULTS

As shown in Fig. 12, the first row is the input images of the NSTI accelerator. The accelerator takes the cropped text region and outputs the prediction of each pixel as shown the third row. In the second row, it merges the 3-D salience map into a 2-D salience map, showing the confidence level of each pixel to the ground true class. In Fig. 12 (a), it shows some good prediction examples with high confidence level and clean prediction boundary. In Fig. 12 (b), some bad predictions with low confidence level are shown, which have uneven illumination or low contrast input images. By evaluating the pixelwise classification accuracy, the NSTI accelerator achieves an accuracy of 90% and 91% testing on two public datasets, ICDAR-03 and ICDAR-13, respectively.

The implementation results are summarized in Table III. In this highly paralleled architecture, we are able to map 46 PEs, which contains 193 MMACs (Mega multiply-add operations) in total onto our chip. The total number of operations (MAC operation is counted as 2 operations) in B-CEDNet is 39 G. The NSTI accelerator can work at a frame rate of 34 fps (1,326

TABLE III
EXPERIMENT RESULTS AND COMPARISON

	ASIC/ GPU	Binary	Process (nm)	Core VDD (V)	Power (W)	Freq (MHz)	Frame rate (fps)	Latency (ms)	Num. of OP	GOP/s		Area (mm^2)	Area efficiency (GOP/s/mm2)	Energy efficiency (GOP/s/W)	
										Peak	Real			Peak	Real
This work	ASIC	Yes	40	0.9	1.9	30	34	40	39 G	14,868	1,326	12.7	893	7825	698
[9]	GPU	Yes	28	N/A	80	1000	200	5	39 G	N/A	7,800	601	N/A	N/A	97
$[30]^1$	ASIC	Yes	65	0.6	N/A	400	N/A	64	1.2 G	N/A	19	1.33 MGE	14 GOP/s/MGE	N/A	56700
$[30]^2$	ASIC	Yes	65	1.2	N/A	N/A	435	N/A	1.2 G	N/A	525	1.33 MGE	395 GOP/s/MGE	N/A	8600
[31]	ASIC	No	45	1	0.6	400	N/A	N/A	N/A	320	294	12.5	25.6	533	490
[27]	ASIC	No	65	0.82-1.17	0.278	100-250	35 fps	115	2.7 G	42	23	12.25	3.43	151	83
[32]	ASIC	No	28	0.575-1.1	0.039	200-1175	58	17	1.3 G	676	78	35.28	600	2930	N/A

∗ For the reference ASIC designs, if the design is testing with different neural network architecture, we pick the one with best-recorded performance.
∗ For all the reference ASIC designs, the power and energy efficiency results do not include the off-chip DRAM. This work include all the memory needed for this application.
∗ $[30]^1$ and $[30]^2$ shows the result for the best energy efficiency and GOP/s, respectively.

GOP/s) with the peak energy efficiency of 7825 GOP/s/W and the real energy efficiency of 698 GOP/s/W. The total power of the NSTI accelerator is 1.9 W with the core consuming 0.8 W. The dynamic power is estimated based upon simulation waveform of the test cases.

The first two rows in Table III compares exactly the same architecture (B-CEDNet) by GPU and our ASIC implementation. It should be noted that the GPU-based implementation for B-CEDNet (binary) already delivered 8× better throughput than that of the CEDNet (non-binary) [9]. Compared with its optimal GPU-based implementation counterpart [9], this work provides 7× better energy efficiency while still maintaining a real-time frame rate with less than 2 W power consumption. Therefore, the proposed accelerator can enable real-time scene text interpretation on the power-constrained mobile devices.

We also compare our work with other ASIC designs for convolutional neural network acceleration (CNN). All of [27], [30]–[32] are general CNN accelerators rather than task-specific ones. Reference [30] is built upon the binary weight CNN, while [27], [31], [32] are built upon fixed-point CNNs. Compared with the throughput-optimal test set in [30], the proposed accelerator achieve 2.5× better throughput in terms of GOP/s. (Since the total number of operations in different network vary a lot, GOP/s is a better reflection of throughput rather than the frame rate.) Compared with fixed-point CNN ASIC designs [27], [31], [32], the proposed accelerator can delivery 6×-58× better throughput. In terms of the latency, even if the number of operations of our network is 14×-32× larger than [30^1, 27], the proposed accelerator achieves the best latency among them. Only when compared with [32], its latency is 2.4× better than ours, due to 30× less number of operations in its network. Among all these ASIC designs, our accelerator is the only one that can process the B-CEDNet with 39 Giga operations in a real-time manner and low latency of 40 ms. Binary feature of B-CEDNet enable us to map 46 PEs containing 193 MMACs for massive spatial parallelism. Highly pipelined data flow control enable more temporal parallelism. Both spatial and temporal parallelism contribute to optimize the throughput and latency in our design. 12× energy efficiency gap between our work and $[30]^2$ can be explained by following points. First, we trade it off for better throughput, since our primary task is to guarantee a real-time throughput. Second, they have designed customized on-chip memory for a low-power design. Additionally, they store the intermediate results (between blocks/layers) and parameters in off-chip DRAM, which are excluded in power consumption reports. We do consider the power consumption for the entire application rather than just for the computation core. Reduction in computation complexity to bit-level operations benefits in power saving. Store all the weights and intermediate results on chip eliminating off-chip communication for the sake of extending battery life.

VI. CONCLUSION

In this paper, we present an ASIC accelerator for real-time and low-latency natural scene text interpretation on power-constrained mobile devices. The NSTI accelerator takes the cropped scene text image as input and output a salience map for pixelwise classification result. To target a real-time throughput and low latency, a B-CEDNet is adopted as the core architecture to enable massive spatial parallelism. A highly pipelined data flow control is applied to enable temporal parallelism. Moreover, all the binarized intermediate results and parameters are stored on chip to eliminate the power consumption and latency overhead of off-chip commutation. This NSTI accelerator is implemented in a 40nm CMOS technology, which can process 128×32 scene text images at 34 fps with an latency of 40 ms for pixelwise interpretation with accuracy no less than 90%. Its real energy-efficiency is 698 GOP/s/W and its peak energy-efficiency can get up to 7825 GOP/s/W. In the IoT applications, the proposed accelerator can be used in power-constrained edge devices to enable real-time augment reality applications for natural scene understanding.

REFERENCES

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[2] M. Rahmani, A. Ghanbari, and M. M. Ettefagh, "Hybrid neural network fraction integral terminal sliding mode control of an inchworm robot manipulator," *Mechanical Systems and Signal Processing*, vol. 80, pp. 117–136, 2016.
[3] M. Rahmani, A. Ghanbari, and M. M. Ettefagh, "A novel adaptive neural network integral sliding-mode control of a biped robot using bat algorithm," *Journal of Vibration and Control*, vol. 24, no. 10, pp. 2045–2060, 2018.

[4] Z. Liu, G. Lin, S. Yang, J. Feng, W. Lin, and W. Ling Goh, "Learning markov clustering networks for scene text detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6936–6944, 2018.

[5] K. Xu, D. Li, N. Cassimatis, and X. Wang, "Lcanet: End-to-end lipreading with cascaded attention-ctc," in *Automatic Face & Gesture Recognition (FG 2018), 2018 13th IEEE International Conference on*, pp. 548–555. IEEE, 2018.

[6] A. Bissacco, M. Cummins, Y. Netzer, and H. Neven, "Photoocr: Reading text in uncontrolled conditions," in *Computer Vision (ICCV), 2013 IEEE International Conference on*, pp. 785–792. IEEE, 2013.

[7] T. Wang, D. J. Wu, A. Coates, and A. Y. Ng, "End-to-end text recognition with convolutional neural networks," in *Pattern Recognition (ICPR), 2012 21st International Conference on*, pp. 3304–3308. IEEE, 2012.

[8] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Deep features for text spotting," in *European conference on computer vision*, pp. 512–528. Springer, 2014.

[9] Z. Liu, Y. Li, F. Ren, H. Yu, and W. Goh, "Squeezedtext: A real-time scene text recognition by binary convolutional encoder-decoder network," 2018.

[10] G.-B. D. L. Inference, "A performance and power analysis," *Whitepaper, November*, 2015.

[11] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[13] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*, vol. 1. MIT press Cambridge, 2016.

[14] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[15] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[16] Y. Cheng, F. X. Yu, R. S. Feris, S. Kumar, A. Choudhary, and S.-F. Chang, "An exploration of parameter redundancy in deep networks with circulant projections," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2857–2865, 2015.

[17] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Advances in neural information processing systems*, pp. 3123–3131, 2015.

[18] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Advances in neural information processing systems*, pp. 4107–4115, 2016.

[19] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European Conference on Computer Vision*, pp. 525–542. Springer, 2016.

[20] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in neural information processing systems*, pp. 1135–1143, 2015.

[21] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2704–2713.

[22] X. Lin, C. Zhao, and W. Pan, "Towards accurate binary convolutional neural network," in *Advances in Neural Information Processing Systems*, pp. 345–353, 2017.

[23] B. Moons and M. Verhelst, "A 0.3–2.6 tops/w precision-scalable processor for real-time large-scale convnets," in *VLSI Circuits (VLSI-Circuits), 2016 IEEE Symposium on*, pp. 1–2. IEEE, 2016.

[24] F. Tu, S. Yin, P. Ouyang, S. Tang, L. Liu, and S. Wei, "Deep convolutional neural network architecture with reconfigurable computation patterns," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 8, pp. 2220–2233, 2017.

[25] X. Zhang, S. Das, O. Neopane, and K. Kreutz-Delgado, "A design methodology for efficient implementation of deconvolutional neural networks on an fpga," *arXiv preprint arXiv:1705.02583*, 2017.

[26] Y. Li, Z. Liu, K. Xu, H. Yu, and F. Ren, "A 7.663-tops 8.2-w energy-efficient fpga accelerator for binary convolutional neural networks," in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 290–291. ACM, 2017.

[27] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, 2017.

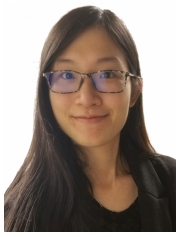
[28] G. Desoli, N. Chawla, T. Boesch, S.-p. Singh, E. Guidetti, F. De Ambroggi, T. Majo, P. Zambotti, M. Ayodhyawasi, H. Singh *et al.*, "14.1 a 2.9 tops/w deep convolutional neural network soc in fd-soi 28nm for intelligent embedded systems," in *Solid-State Circuits Conference (ISSCC), 2017 IEEE International*, pp. 238–239. IEEE, 2017.

[29] K. Bong, S. Choi, C. Kim, S. Kang, Y. Kim, and H.-J. Yoo, "14.6 a 0.62 mw ultra-low-power convolutional-neural-network face-recognition processor and a cis integrated with always-on haar-like face detector," in *Solid-State Circuits Conference (ISSCC), 2017 IEEE International*, pp. 248–249. IEEE, 2017.

[30] R. Andri, L. Cavigelli, D. Rossi, and L. Benini, "Yodann: An architecture for ultralow power binary-weight cnn acceleration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 1, pp. 48–60, 2018.

[31] P.-H. Pham, D. Jelaca, C. Farabet, B. Martini, Y. LeCun, and E. Culurciello, "Neuflow: Dataflow vision processing system-on-a-chip," in *Circuits and Systems (MWSCAS), 2012 IEEE 55th International Midwest Symposium on*, pp. 1044–1047. IEEE, 2012.

[32] G. Desoli, N. Chawla, T. Boesch, S.-p. Singh, E. Guidetti, F. De Ambroggi, T. Majo, P. Zambotti, M. Ayodhyawasi, H. Singh *et al.*, "14.1 a 2.9 tops/w deep convolutional neural network soc in fd-soi 28nm for intelligent embedded systems," in *Solid-State Circuits Conference (ISSCC), 2017 IEEE International*, pp. 238–239. IEEE, 2017.

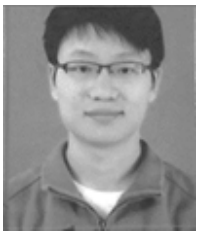


Yixing Li received B.Eng and M.S. degrees in microelectronics from South China University of Technology, Guangzhou, China, in 2012 and 2015 respectively, and is currently working towards her Ph.D. degree in School of Computing, Informatics, and Decision Systems Engineering at Arizona State University. Her current research interests focus on hardware acceleration for data analytics, hardware-friendly algorithm and computer vision applications.

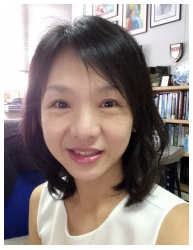
Zichuan Liu (S'17) received the B.Eng. degree communication engineering from the Jilin University, China, in 2014. He is currently pursuing the Ph.D degree with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. His research interests include deep learning, computer vision and neural network acceleration.

Wenye Liu (S'17) received the B.S. degree in microelectronics from Shenzhen University, China, in 2014, the B.S. degree in physics from Umea University, Sweden, in 2014, and the M.S. degree in IC design engineering from Hong Kong University of Science and Technology. He is currently pursuing the Ph.D degree with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. His current research interests include hardware accelerator and deep learning.

Yu Jiang (M'15) received the B.Eng. degree in electronics and information engineering from Anhui University, Anhui, China, and the M.S degree in electronics science and technology from Fudan University, Shanghai, China, in 2011 and 2014, respectively. She joined VIRTUS and CINTRA Lab as a Ph.D. student at Nanyang Technological University, Singapore. Her research interests are miniaturized sensors design for biomedical applications.



Yongliang Wang received the B.Eng. degree in electronics and information engineering from Anhui University, Anhui, China, and the M.S. degree in electronics science and technology from Anhui University, Anhui, China, in 2011 and 2014, respectively. He joined the digital design implementation department as a senior engineer at Verisilicon Corp., Shanghai. His research interests are complex SOC static timing analysis and signoff at advance technology node.



Wang Ling Goh (S'91-M'06-SM'09) received the B.Eng. degree in electrical and electronic engineering and Ph.D. degree in microelectronics from Queen's University of Belfast, U.K. in 1990 and 1995, respectively. She was a Research Engineer at the Northern Ireland Semiconductor Research Centre while working toward the Ph.D. degree. She joined the School of Electrical and Electronic engineering, Nanyang Technological University, Singapore, as a Lecturer in 1996, and became an Associate Professor in 2004. Her research interests include digital/mixed-signal IC design, telemetry circuits, neural recording ICs, and 3-D IC.

Hao Yu (M'06-SM'14) obtained Ph. D degree from electrical engineering department at UCLA in 2007. He is now with Southern University of Science and Technology (SUSTech), China. His primary research interest is energy-efficient data links, sensors and analysis . He has about 250 peer-reviewed IEEE/ACM publications, 6 books, 1 best paper award of ACM Transaction, 3 keynote talks, 3 best paper award nominations (DAC06, ICCAD06, ASP-DAC12), 3 student paper competition (advisor) finalists (IMS15, RFIC13, SiRF13), 1 inventor award from semiconductor research cooperation (SRC), and 20 granted patents. He is associate editor (Nature-SciReports, IEEE TBioCAS, ACM TECS, Elsevier Microelectronics etc.) and technical program committee member (IEEE-CICC, IEEE-ASSCC, ACM-DAC, ACM-DATE etc.) of many IEEE/ACM international journals and conferences. He is a senior member of IEEE and member of ACM.

Fengbo Ren (S'10-M'15) received the B.Eng. degree from Zhejiang University, Hangzhou, China, in 2008 and the M.S. and Ph.D. degrees from University of California, Los Angeles, in 2010 and 2014, respectively, all in electrical engineering.

In 2015, he joined the faculty of the School of Computing, Informatics, and Decision Systems Engineering at Arizona State University (ASU). His Ph.D. research has involved in designing energy-efficient VLSI systems, accelerating compressive sensing signal reconstruction, and developing emerging memory technology. His current research interests are focused on hardware acceleration and parallel computing solutions for data analytics and information processing, with emphasis on bringing energy efficiency and signal intelligence into a wide spectrum of todayâs computing infrastructures, from data center server systems to wearable and Internet-of-things devices. He is a member of the Digital Signal Processing Technical Committee and VLSI Systems Applications Technical Committee of the IEEE Circuits and Systems Society.

He received the Broadcom Fellowship in 2012, the National Science Foundation (NSF) Faculty Early Career Development (CAREER) Award in 2017, and the Google Faculty Research Award in 2018.