

A 1.15 TOPS/W Energy-Efficient Capsule Network Accelerator for Real-Time 3D Point Cloud Segmentation in Mobile Environment

Gwangtae Park¹, Graduate Student Member, IEEE, Dongseok Im¹, Student Member, IEEE,
Donghyeon Han¹, Graduate Student Member, IEEE, and Hoi-Jun Yoo¹, Fellow, IEEE

Abstract—An energy-efficient capsule network accelerator is proposed for real-time 3D point cloud segmentation in mobile devices. The proposed accelerator adopts the pipelined heterogeneous core architecture to achieve 1.55× throughput enhancement. Furthermore, the proposed dynamic route skipping controller predicts unimportant operations and skips them to reduce the external memory access by 39.1%. At last, the new squash activation function unit exploits the look-up table (LUT) based computing with L2-norm approximation to minimize the power and area overhead. The proposed architecture is implemented with the FPGA, Altera Cyclone V 5CEBA9F31C7 and we test capsule network-based 3D point cloud segmentation application. It consumes 2.15 W power and shows 0.05 TOPS/W energy-efficiency. Also, the architecture is simulated with the 65nm CMOS technology, showing 94.3mW power consumption and 1.15 TOPS/W energy-efficiency.

Index Terms—Capsule network, dynamic routing, 3D point cloud segmentation, L2-norm, look-up table, external memory access, energy-efficient accelerator.

I. INTRODUCTION

RECENTLY, the segmentation network achieved great success in autonomous driving [1], [2] and robotic assistance [3], [4], and so on. Nevertheless, CNN based 2D image segmentation cannot provide enough information such as 3D location [5]. Naturally, the application of CNN is extended to a 3D point cloud to resolve this problem [5], [6]. However, CNN for the 3D point cloud faces the performance degradation, especially when the input domain changes (e.g., rotation, deformation). Reference [8] found that the capsule network (CapsNet) can solve this problem by adopting a dynamic routing (DR) algorithm [7] and proved that the 3D-CapsNet has a better generalization ability compared with the conventional CNNs.

However, 3D-CapsNet acceleration faces both the computation and memory bottlenecks due to its heterogeneous characteristics among the layers. Fig. 1 shows the overall 3D-CapsNet architecture with the computation and external

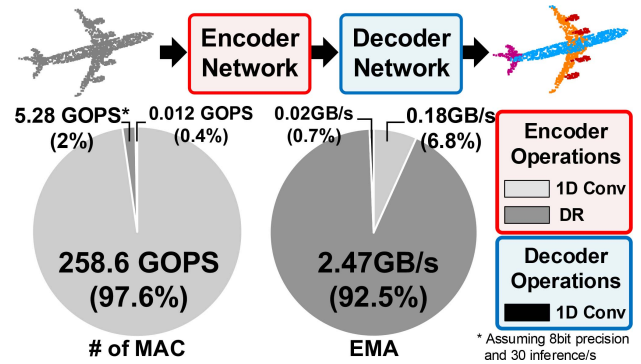


Fig. 1. Computation and EMA breakdown of the overall 3D-CapsNet.

memory access (EMA) breakdown. The 3D-CapsNet consists of the encoder and the decoder network. The convolution (CONV) operation in the encoder accounts for 97.6% of the overall multiply and accumulation (MAC) operations. In contrast, the DR requires 92.5% of EMA. To implement a real-time 3D point cloud segmentation through CapsNet in the mobile environment, both reducing the MAC operations of the CONV and the EMA of the DR are important.

3D-CapsNet demands a 16.5x larger EMA than the 2D image segmentation network called DeepLab v3+ [9] that uses MobileNetV2 [10] as a feature extractor. Therefore, recent CNN accelerators [11], [12] consume large power caused by heavy memory transactions from off-chip DRAM [13] when it accelerates the 3D-CapsNet. Previous 2D-CapsNet accelerator [14] dealt with the EMA problem by using the 8 MB on-chip buffer. However, 3D-CapsNet requires 68 MB cache which is much higher than the 2D-CapsNet.

The previous 2D-CapsNet accelerator [14] implements the squash activation function with look-up-table (LUT) based computing to reduce the latency. The approach suggested by [14] requires two kinds of LUTs – one for the square-root operation and the other for the final squash activation function output. However, the squash activation computing method suggested in [14] requires a large size of LUT and results in low energy and area efficiency.

In this brief, an energy-efficient CapsNet accelerator is proposed for the high-speed 3D point cloud segmentation in mobile devices. It has three key features: 1) a pipelined heterogeneous core architecture for throughput enhancement 2) dynamic route skipping unit for the EMA reduction, and 3) the energy-efficient squash activation function unit by combining L2-norm approximation and LUT based computing. As a result, the proposed accelerator achieves

Manuscript received June 6, 2020; accepted June 23, 2020. Date of publication June 26, 2020; date of current version September 3, 2020. This work was supported by the Samsung Electronics. This brief was recommended by Associate Editor Y. H. E. Bonizzoni. (Corresponding author: Hoi-Jun Yoo.)

The authors are with the School of Electrical Engineering, Korea Advanced Institute of Science and Technology, Daejeon 34141, South Korea (e-mail: gwangtaepark@kaist.ac.kr; dsim@kaist.ac.kr; hhd4797@kaist.ac.kr; hjyoo@kaist.ac.kr).

Color versions of one or more of the figures in this article are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TCSII.2020.3005191

1549-7747 © 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.

See <https://www.ieee.org/publications/rights/index.html> for more information.

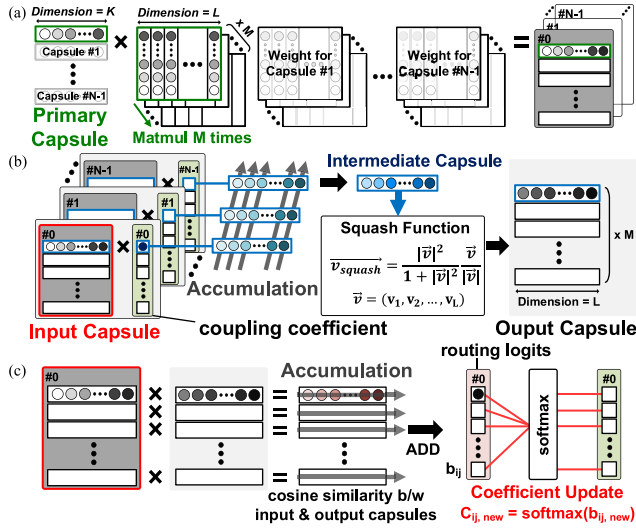


Fig. 2. Description of the dynamic routing (DR) process (a) FC stage (b) Feedforward (FF) stage (c) Feedback stage (FB).

1.15 TOPS/W energy-efficiency in the 3D point cloud segmentation application.

The rest of this brief is organized as follows. Section II briefly explains the DR algorithm and why it induces large EMA. After that, the overall architecture of the proposed CapsNet accelerator and its key features are introduced through Section III. Finally, this brief shows the measurement results with visual implementation in Section IV and concludes in Section V.

II. DYNAMIC ROUTING ALGORITHM

Dynamic Routing (DR) consists of three stages: fully-connected (FC), feedforward (FF), and the feedback (FB) in Fig. 2. First, the primary capsule is transformed through the FC stage. In the FF stage, input capsules are multiplied by the scalar coupling coefficient (c_{ij}) and summed up along each dimension to generate the intermediate capsules. After that, the squash activation function [7] makes the final output capsules. Coupling coefficients are updated in the FB stage. Initially, cosine similarity between the input capsules and output capsules is calculated. The cosine similarities are added to the routing logits (b_{ij}). Finally, the softmax function is applied to the routing logits and generates the new coupling coefficients. The FF and FB stages repeat multiple iterations to increase segmentation accuracy. Based on the experiment, the 3D-CapsNet shows the best accuracy if the DR iteration is 7.

The EMA of the DR occurs due to two main reasons. The first reason is a large number of weights in the FC stage. The other reason is the input capsule accessed in proportional to the DR iteration. The EMA caused by the FC stage can be decreased by quantizing the weight from 8bit to 4bit. Unlike the FC case where weight value is determined, input capsule value changes during the inference. Therefore, EMA caused by the input capsule cannot be solved by adopting the quantization. As a result, after applying the FC weight quantization, the input capsule occupies 54.2%, 68.4%, and 75.6% of the total EMA if DR iteration is 3, 5, and 7, respectively.

III. PROPOSED CAPSULE NETWORK ACCELERATOR

A. Overall Architecture

The overall architecture of the CapsNet accelerator is shown in Fig. 3. It consists of a CONV core and a DR core. Four

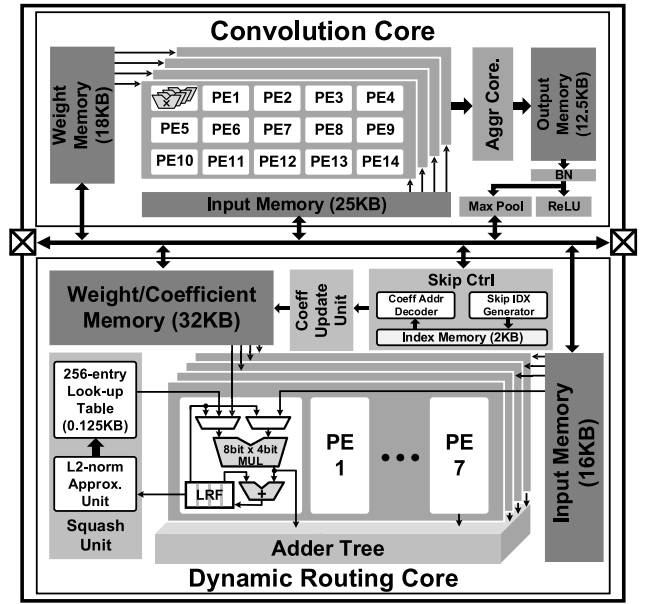


Fig. 3. Overall architecture of the proposed CapsNet accelerator.

processing elements (PE) in CONV core compute four different input channels. Each PE array consists of 3×5 PEs, processing five different points in parallel along three output channels. Each PE includes four $8\text{bit} \times 8\text{bit}$ multipliers, generating four different output channels. Multiplication results of the same output channel are summed up at the aggregation core.

The DR core includes reconfigurable datapaths to support four different operations of the DR. In the FC multiplication operation, every PE receives the unicasted input and weight from the memory. The adder tree sums up the multiplied results and generates the input capsule in Fig. 2(a). The input capsule is stored in the input memory. In the coefficient multiplication scenario, each component of the input capsule is unicasted from the input memory to each PE and multiplied with a broadcasted coupling coefficient from the weight/coefficient memory. In the squash activation function operation, coefficient multiplication results are sent to the L2-norm approximation unit which estimates the L2-norm. This output is used as an index for 256 LUT. Multiplication between the coefficient multiplication result and indexed LUT value generates the output capsule in Fig. 2(b). Cosine similarity between the input and output capsule is measured by using the output capsule element residing in each PE's local register file (LRF) as a weight. The coefficient update unit (CUU) receives the cosine similarity results and produces the new coupling coefficient through the softmax units and stores it into the weight/coefficient memory.

The CONV core includes input memory (IMEM), a double-buffered weight memory (WMEM), and the output memory (OMEM). In the DR core, IMEM, weight/coefficient memory (WCMEM), and index memory (IDXMEN) are all double buffered. The WCMEM is used for the weight memory at the FC stage and for the coefficient storage at the FF and FB stage.

B. Network Optimization and Channel Group Level Pipelining

The large computation cost of the 3D-CapsNet comes from the 1D CONV. The proposed accelerator utilizes the group

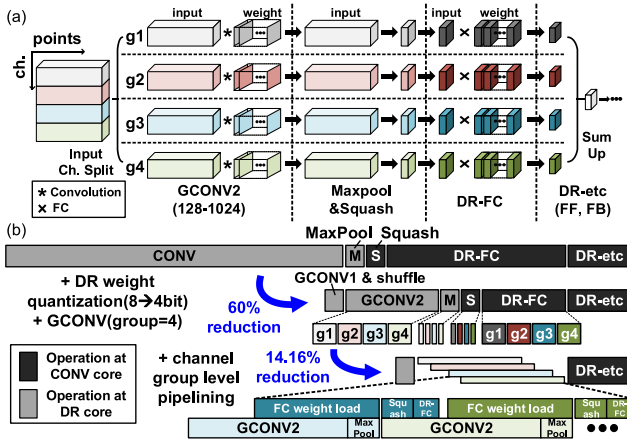


Fig. 4. (a) Network modification by applying group convolution (b) Latency reduction by channel group level pipelining.

convolution (GCONV) with channel shuffling [15] instead of the 1D CONV to increase the throughput. GCONV reduces the computation cost by removing connections between the input channel and the output channel of the different groups. However, the information loss due to the lost connections degrades the network performance. For 3D-CapsNet, the modified network degrades the accuracy by 1.3%. The shuffling layer compensates this by interchanging the channels among all the groups. As a result, the modified 3D-CapsNet maintains its performance with less computational cost.

Since the shuffling layer changes the order of the GCONV output channel, the proposed processor contains reordering logic in the aggregation core which is the sub-unit of the CONV core. It rearranges the channels of the GCONV result before it is transferred to the CONV core output memory. As a consequence, higher throughput enhancement can be achieved by supporting the GCONV with the small reordering logic.

Although the 3D-CapsNet latency is decreased by adopting the GCONV, it does not satisfy the real-time requirement. The main bottleneck is caused in the DR core. During the channel shuffling, the DR core is stalled due to the dependency among the channel groups. In this brief, the channel group level pipeline is constructed. In order to enable the pipeline, the 3D-CapsNet is optimized as follows. A 1D CONV layer which has 128-1024 input-output channels is replaced with three layers – the GCONV layer with 128-128 input-output channels (GCONV1), the shuffling layer, and the GCONV layer with 128-1024 input-output channels (GCONV2). Since there is no shuffling layer that follow the GCONV2, there is no data dependency among the channel groups of the GCONV2 as shown in Fig. 4(a). As a result, the GCONV2 and the DR-FC operation of each channel group can be pipelined.

The proposed accelerator constructs the channel group level pipeline as shown in Fig. 4 (b). In the beginning, CONV core computes GCONV2 of a single channel group. The latency of the FC weight load is hidden by computing the GCONV simultaneously. The GCONV result goes through max-pooling and are sent to the DR core to apply the squash activation function. After that, the DR core computes FC operation using the pre-fetched weight. By quantizing the FC weights and optimizing the 3D-CapsNet decreased the latency to 15.07% and 44.97% each. The pipelined architecture in the proposed accelerator achieves an additional 14.16% latency reduction.

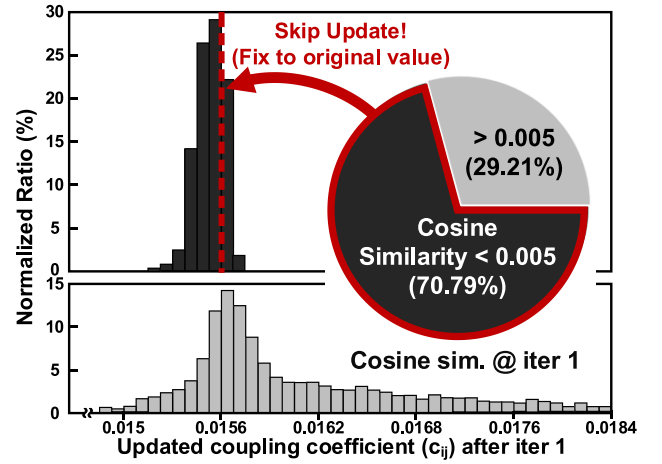


Fig. 5. Relative coupling coefficient change at first DR iteration.

C. Dynamic Route Skipping Unit

As shown in Fig. 5, 70.79% of the cosine similarity results in the DR are less than a threshold (0.005), only changing the routing logit values slightly. As a result, 81.24% of the coupling coefficients updated by those routing logits change within 1% even though the softmax function is applied. Therefore, skipping update operations on those coupling coefficients does not affect the feedforward step results significantly. The sign mismatch ratio between the output capsule calculated by coupling coefficients updated with skipping and that calculated without skipping is 0.07% on Shapenet-Part [17] dataset. Based on this analysis, the DR skipping unit supports omitting those operations and reduces the EMA.

Fig. 6. describes the detailed computation flows of the DR skipping method. After the first FF stage of the DR, the FB stage's update step starts where the cosine similarity is measured (Fig. 6 (a)). Here, input capsules from the input memory and the output capsule stored in the LRF are multiplied with the input capsule. These results are summed up at the adder tree, and the skip index generator compares the single aggregated result with the threshold value to determine whether the route can be skipped or not.

While waiting for the skipping decision, the DR core proceeds the FF step of the next iteration by assuming that all coupling coefficient updates will be skipped (Fig. 6(b)). Hence the DR core computes the FF step by reusing the input capsules and the coupling coefficients of the previous iteration. This result is either stored or omitted depending on the binary skip index made at the skip index generator. If the produced index is one, future FB steps which include applying softmax function are all skipped and the computed FF result is stored in the LRF. Future FF step is also skipped by using the stored FF result as a pre-computed partial sum. If the index is zero, the FF result is omitted and the DR core updates the corresponding coupling coefficient by resuming the FB stage. The CUU computes the softmax operation based on the LUT and stores the new coupling coefficient into the weight/coefficient memory. The updated coefficients are used for the future FF stage.

In the following iterations (Fig. 6(c)), the coefficient address decoder receives an 8bit skip index at each cycle and generates the coupling coefficient addresses by counting the number of zeros between the nonzero values. Only input capsule values corresponding to those addresses are loaded from the external memory. The multiplication result using the updated coupling

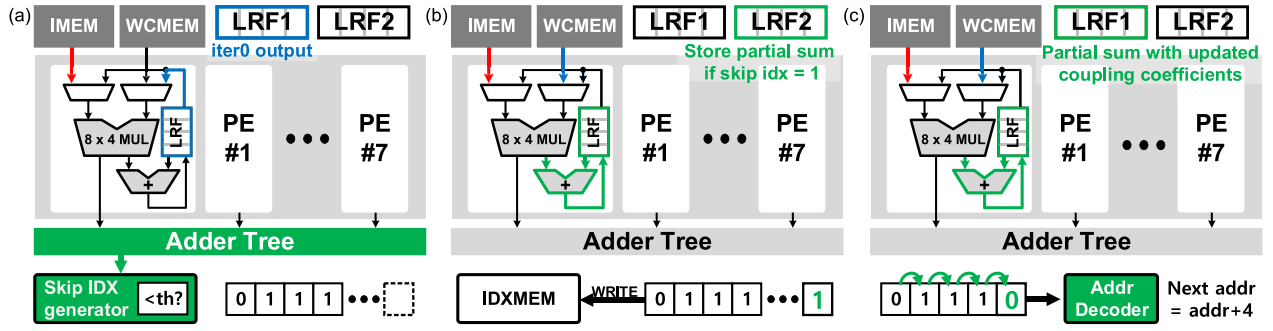


Fig. 6. Dynamic Route Skipping (a) Update at first iteration (b) coefficient multiplication during skip decision (c) coefficient multiplication after the first iteration.

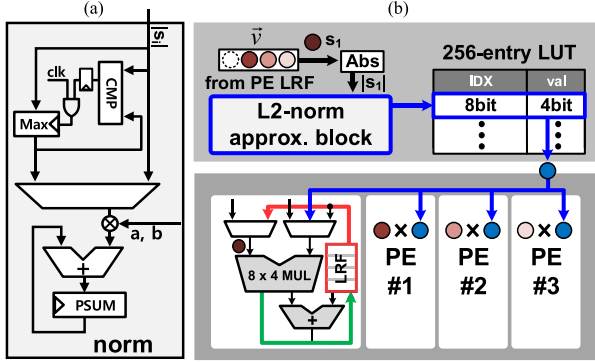


Fig. 7. (a) L2-norm approximation block (b) DR core Squash function datapath.

coefficient is stored in the LRF as shown in Fig. 6(c). The final result is generated by simply adding the partial sum stored in Fig. 6(b) and newly computed partial sum stored in Fig. 6(c).

When the threshold value is 0.005, 67.27% of the DR operations could be skipped after the first iteration. The DR skipping reduces the total EMA by 39.1%.

D. L2-Norm Approximation Based Squash Function Unit

Fig. 7(b) shows the datapath of the squash activation function applied to the input capsule \vec{v} . Input capsule elements are stored in LRF of each PE. Absolute values of the capsule elements are fed into the L2-norm approximation block in stream. After the L2-norm approximation calculation is finished, the result is used as an index of the squash LUT. The squash LUT consists of 256 entries and generates the 4bit output. The output of the squash LUT is broadcasted into each PE and it is multiplied with the input capsule elements.

The proposed L2-norm approximation block implements the approximation method introduced in [16]. We denote n -dimensional input capsule as $\vec{v} = (s_1, s_2, \dots, s_n)$, L1-norm, L2-norm and L-infinity norm as l_1 , l_2 , and l_∞ . L1-norm is always larger than L2-norm and L-infinity norm is always smaller than L2-norm. Therefore, L2-norm is approximated as a linear combination of two values, as shown in the below equation.

$$l_2 \approx a * l_1 + b * l_\infty$$

$$l_1 = |s_1| + |s_2| + \dots + |s_n|, l_\infty = \max(|s_1|, |s_2|, \dots, |s_n|) \quad (1)$$

Thanks to linear approximation [16], the L2-norm is simply obtained by the L2-norm approximation block. It estimates the L2-norm value with an adder, a comparator, and a multiplier.

TABLE I
FPGA UTILIZATION RESULT

	Used (DR Core)	Used (CONV Core)	Total on FPGA	Utilization (%)
ALM	13203	20903	113560	30
BRAM	64	221	1220	23
DSP	32	246	342	81
register	25305	17788	227120	19

TABLE II
COMPARISON TABLE

	DATE 2019 [14]	HPCA 2020 [19]	This Work	
Application	Image Classification	Image Classification	Point Segmentation	
Algorithm	2D Capsule Network	Dynamic Routing	3D Capsule Network	
Platform	ASIC	ASIC	ASIC	FPGA Cyclone V
Process	32 nm	24 nm	65 nm	28 nm
Voltage	1.05 V	-	1.2 V	1.1 V
Clock Frequency	250 MHz	312.5 MHz	200 MHz	200 MHz
Precision	8 bit fixed-point	32 bit floating-point	4/8 bit fixed-point	4/8 bit fixed-point
# of DSP	-	-	-	278
Total ALM	-	-	-	34.1 K
On-chip Memory	8 MB	8 GB	50 KB	50.18 KB
Area	2.9 mm ²	3.11 mm ²	4.56 mm ²	-
Peak Performance	128 GOPS	320 GFLOPS	96 / 12.8 GOPS*	96 / 12.8 GOPS*
Power	202 mW	2.24 W	72.7 / 21.6 mW*	0.91 / 1.23 W*
Energy Efficiency	0.64 TOPS/W	0.14 TFLOPS/W	1.15 TOPS/W	0.05 TOPS/W
Normalized Area Efficiency**	44.14 GOPS/mm ²	57.88 GFLOPS/mm ²	98.44 GOPS/mm ²	-

* Conv/DR core reported separately ** Normalized for 32nm Tech.

The approximation coefficient a and b value depends on data distribution and capsule length. Therefore, the processor adopts different coefficients for each squash layer. The coefficients are determined by sweeping them from 0 to 1 by an interval of 0.001. The relative error of the L2-norm approximation for the first squash layer was 4.36% when $a = 0.45$ and $b = 0.29$. For the second squash layer, the relative error was 5.21% when $a = 0.058$ and $b = 0.042$.

In summary, the proposed squash unit shows 1.18mW power consumption and 0.037mm² area occupation. The proposed activation function unit shows 5.14× higher area efficiency and 3.92× lower power consumption compared with [14].

IV. IMPLEMENTATION RESULT

The proposed capsule network accelerator was implemented on an FPGA Altera Cyclone V 5CEBA9F31C7 evaluation board. The FPGA utilization result is shown in Table I. The power consumption is 2.15W under 1.1V supply voltage and 200MHz operating frequency, showing 0.05 TOPS/W energy efficiency. The proposed accelerator was also implemented as



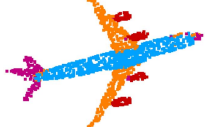


	Input Image	Ground Truth	Baseline	GCONV+QUANT	This work
Image					
Acc.	-	-	85.27%	84.80%	83.95%

Fig. 8. Part segmentation result on an airplane category in the Shapenet-Part dataset.

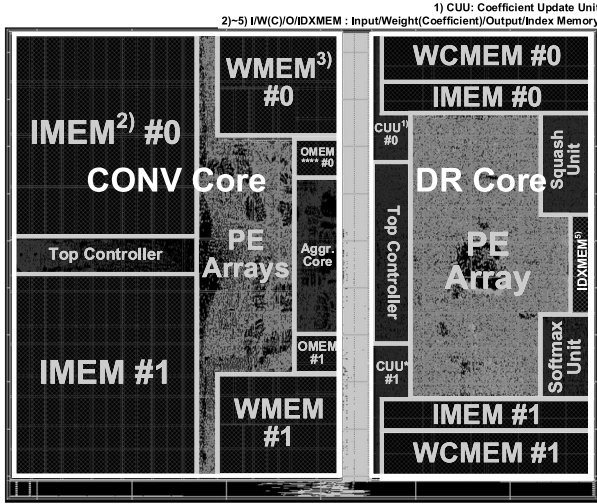


Fig. 9. Layout photograph.

an ASIC in 65nm CMOS technology, whose layout photograph is shown in Fig. 9. It occupies 4.56 mm² area and operates in 200 MHz clock frequency under 1.2V supply voltage. As shown in Table II, the power consumption of the ASIC is 94.3mW which is 2.14× smaller than [14].

In this brief, segmentation accuracy of the proposed 3D-CapsNet accelerator was measured as explained in [18] on the Shapenet-Part dataset [17]. The baseline network is trained as described in [8] with an 8bit dynamic fixed point precision. Compared with the baseline, accuracy degrades by 0.04%, 0.14%, 0.09%, and 0.67% respectively if FC weight quantization, GCONV, DR skipping, and L2-norm is applied. Fig. 8 shows one of the 3D point cloud segmentation results on the Shapenet-Part dataset. The proposed accelerator runs the target 3D-CapsNet at 32.48 FPS when the DR iteration is 3.

V. CONCLUSION

In conclusion, a 94.3mW low-power and 32.48 FPS real-time CapsNet accelerator are proposed for 3D point cloud segmentation. With the group-level pipelined core architecture, throughput is increased by 1.55×. The DR core supports skip operations thus reduce the EMA by 39.1%. At last, the squash activation function is implemented with an L2-norm approximation block, consuming 5.14× and 3.92× smaller area and power compared with the previous work.

The capsule network accelerator was the first verified on the Altera Cyclone V 5CEBA9F31C7 FPGA and run at a clock frequency of 200 MHz. The design was also implemented based on the 65nm process, showing 1.15 TOPS/W energy efficiency which is 1.8× higher than the state-of-the-art capsule network accelerator. Finally, the proposed capsule network accelerator was successfully demonstrated for the

real-time 3D point cloud segmentation application with the high energy-efficiency.

REFERENCES

- [1] M. Ren and R. S. Zemel, "End-to-end instance segmentation with recurrent attention," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, 2017, pp. 297–301.
- [2] M. Tremlet *et al.*, "Speeding up semantic segmentation for autonomous driving," in *Proc. Conf. Neural Inf. Process. Syst. (NIPS)*, 2016, pp. 1–7.
- [3] S. Chandra, S. Tsogkas, and I. Kokkinos, "Accurate human-limb segmentation in RGB-D images for intelligent mobility assistance robots," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV) Workshops*, Santiago, Chile, 2015, pp. 436–442.
- [4] T. Do, A. Nguyen, and I. Reid, "AffordanceNet: An end-to-end deep learning approach for object affordance detection," in *Proc. Int. Conf. Robot. Automat. (ICRA)*, Brisbane, QLD, Australia, 2018, pp. 1–5.
- [5] B. Yang, W. Luo, and R. Urtasun, "PIXOR: Real-time 3D object detection from point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Salt Lake City, UT, USA, 2018, pp. 7652–7660.
- [6] S. Shi, X. Wang, and H. Li, "PointRCNN: 3D object proposal generation and detection from point cloud," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Long Beach, CA, USA, 2019, pp. 770–779.
- [7] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Proc. Conf. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 3856–3866.
- [8] Y. Zhao, T. Birdal, H. Deng, and F. Tombari, "3D point capsule networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Long Beach, CA, USA, 2019, pp. 1009–1018.
- [9] L. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 833–851.
- [10] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Salt Lake City, UT, USA, 2018, pp. 4510–4520.
- [11] D. Shin, J. Lee, J. Lee, and H. Yoo, "14.2 DNPU: An 8.1TOPS/W reconfigurable CNN-RNN processor for general-purpose deep neural networks," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, San Francisco, CA, USA, 2017, pp. 240–241.
- [12] J. Lee, C. Kim, S. Kang, D. Shin, S. Kim, and H. Yoo, "UNPU: An energy-efficient deep neural network accelerator with fully variable weight bit precision," *IEEE J. Solid-State Circuits*, vol. 54, no. 1, pp. 173–185, Jan. 2019.
- [13] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers (ISSCC)*, San Francisco, CA, USA, 2014, pp. 10–14.
- [14] A. Marchisio, M. A. Hanif, and M. Shafique, "CapsAcc: An efficient hardware accelerator for CapsuleNets with data reuse," in *Proc. Design Automat. Test Eur. Conf. Exhibit. (DATE)*, Florence, Italy, 2019, pp. 964–967.
- [15] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Salt Lake City, UT, USA, 2018, pp. 6848–6856.
- [16] C. Seol and K. Cheun, "A low complexity Euclidean norm approximation," *IEEE Trans. Signal Process.*, vol. 56, no. 4, pp. 1721–1726, Apr. 2008.
- [17] L. Yi *et al.*, "A scalable active framework for region annotation in 3D shape collections," *ACM Trans. Graph.*, vol. 35, no. 6, pp. 1–12, 2016.
- [18] J. Li, B. M. Chen, and G. Lee, "SO-Net: Self-organizing network for point cloud analysis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Salt Lake City, UT, USA, 2018, pp. 9397–9406.
- [19] X. Zhang, S. L. Song, C. Xie, J. Wang, W. Zhang, and X. Fu, "Enabling highly efficient capsule networks processing through a PIM-based architecture design," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, San Diego, CA, USA, 2020, pp. 542–555.