A Scalable Pipelined Dataflow Accelerator for Object Region Proposals on FPGA Platform *

Wenzhi Fu [1,3], Jianlei Yang [1,3], Pengcheng Dai [2,3], Yiran Chen [4] and Weisheng Zhao [2,3]

[1] School of Computer Science and Engineering, Beihang University, Beijing, 100191, China.

[2] School of Electronic and Information Engineering, Beihang University, Beijing, 100191, China.

[3] Fert Beijing Research Institute, BDBC, Beihang University, Beijing, 100191, China.

[4] Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708, USA.

{jianlei, weisheng.zhao}@buaa.edu.cn

ABSTRACT

Region proposal is critical for object detection while it usually poses a bottleneck in improving the computation efficiency on traditional control-flow architectures. We have observed region proposal tasks are potentially suitable for performing pipelined parallelism by exploiting dataflow driven acceleration. In this paper, a scalable pipelined dataflow accelerator is proposed for efficient region proposals on FPGA platform. The accelerator processes image data by a streaming manner with three sequential stages: resizing, kernel computing and sorting. First, Ping-Pong cache strategy is adopted for rotation loading in resize module to guarantee continuous output streaming. Then, a multiple pipelines architecture with tiered memory is utilized in kernel computing module to complete the main computation tasks. Finally, a bubble-pushing heap sort method is exploited in sorting module to find the top-k largest candidates efficiently. Our design is implemented with high level synthesis on FPGA platforms, and experimental results on VOC2007 datasets show that it could achieve about 3.67X speedups than traditional desktop CPU platform and >250X energy efficiency improvement than embedded ARM platform.

Keywords

Scalable pipeline, Dataflow accelerator, Region proposal, FPGA platform, Streaming processing

1. INTRODUCTION

Recent years, deep neural networks have made a great success in image classification tasks of single object [1]. However, it can not be directly applied to multi-object detection, which is much more practical in real-world applications [2, 3]. Object detection first decides the interested regions which potentially include objects (so-called region proposal) and then performs image classification on these proposed regions. Efficient real-time object detection is a prerequisite for many kinds of unmanned systems since the energy efficiency of intensive computation is critical for them. There have been many research works focused on hardware acceleration for image classification [4] but no one designed for region proposals, which have posed a bottleneck in efficient object detection.

Recently several computationally intensive approaches have shown a better performance [5], however, it is not practical for them running on embedded platform with limited resources and restricted power consumption. On the other hand, binarized normed gradients (BING) method is proposed as an effective approach for region proposal generation [6, 7, 8, 9], which achieves state-of-the-art detection rate. However, even though there have already been several techniques to optimize the implementation of BING algorithm on CPU[6], due to the control-flow processing style of Von Neumann architecture, it is still difficult for BING to run efficiently especially in embedded platform.

In this work, a scalable dataflow accelerator is proposed for the BING algorithm who has been reformed to a dataflow-driven manner. With the memory hierarchy, this efficiency of the streaming processing can be highly improved which overcomes the shortages of the traditional platform. Furthermore, this accelerator has a good scalability to be scaled to a larger parallelism efficiently without the problem of synchronization.

The main contributions of this work are listed as following:

(1) The BING algorithm is reformed as a dataflow-driven manner to obtain significant benefits from dataflow processing.

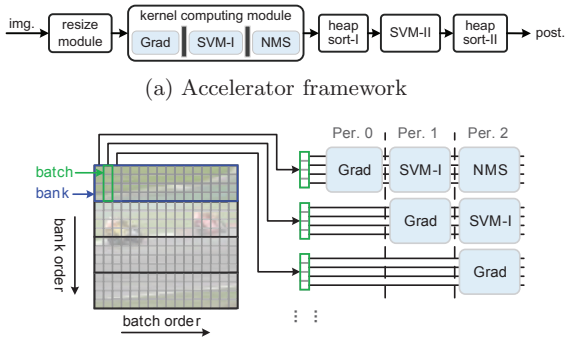
(2) A dataflow architecture is proposed for the BING algorithm, which generate a continues input stream then process it in a streaming manner to fully deployed the locality and minimized the intermediate data amount.

2. REGION PROPOSAL WITH BING

The aim of region proposal is to find the interested regions which potentially contain objects with minimized windows, which is important in multi-object detection tasks. Previous works [6, 7] have shown that a simple 8×8 feature by computing the normed gradients could be adopted as an efficient region proposal. After binarizing the normed gradients feature, BING could achieve efficient objectness estimation. The original image is first resized to different sizes with different preset resizing ratios, therefore, the region proposal candidates with different resolution in the original image can be represented with 8×8 window uniformly in the resized images. Then, the SVM (Support Vector Machine) stage I is adopted to evaluate the confidence for each region proposal.

*This work was supported in part by the National Natural Science Foundation of China (61602022, 61501013, 61571023, 61521091 and 1157040329), National Key Technology Program of China (No. 2017ZX01032101) and the 111 Talent Program B16001.

978-1-7281-0214-6/18/$31.00 ©2018 IEEE
DOI 10.1109/FPT.2018.00070



(a) Accelerator framework

(b) Pipelined kernel computing driven by resized images

Figure 1: Demonstration of our proposed accelerator.

Following that, the NMS (non-maximum suppression) stage is utilized to reduce the overlap among the proposals. After that, top-n largest candidates are selected from the region proposals corresponding to each resized image and the SVM stage II is performed to evaluate the confidence among all of the resized images. Finally, the top-k largest candidates could be obtained as final proposals by a sorting stage. Since the BING algorithm has few branch or jump operations, it could be abstracted as a data-driven algorithm which is ideal to perform dataflow driven acceleration.

3. PROPOSED DATAFLOW ACCELERATOR

3.1 Accelerator Framework

Corresponding to the computation process of BING algorithm, the framework of the proposed dataflow accelerator shown in Fig. 1(a) can also be divided into resizing module, kernel computing module and sorting module. The function of each module is same as the corresponding computation in the algorithm but can be completed in a different manner. As shown in Fig. 1(a), the resizing module first resize the original image with preset ratios then partition the resized image into a series of batch, which represent for four neighbor pixels vertically. The following kernel computing module works in the granularity of batch and can be divided into three stages: CalcGrad, SVM and NMS operations, which can process the continuously batch streaming with a parallel pipelines architecture and exports a stream of candidates. The sorting module is deigned to obtain the top-k or top-n candidates by performing bubble-pushing heap sort strategy to satisfy the throughput requirements. Finally, the region proposals could be obtained after post processing. Without loss of generality, the kernel computing module is demonstrated by four pipelines in this paper which could be extended as more pipelines as following. And the bubble-pushing heap sort model is very similar to [10] which is too trivial to be listed here.

3.2 Resizing Module

A naive resizing approach is carried out in [11] but it cannot satisfy our requirements for streaming processing. The kernel computing module requires a continuous batch streaming output from resizing module to make sure the multiple pipelines are fully loaded. In this work, the original image is partitioned into four blocks uniformly as shown in Fig. 2. Only one port of the configured BRAMs is assigned for each block

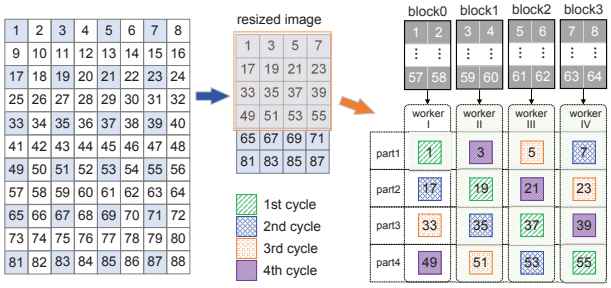


Figure 2: Illustration of resizing module.

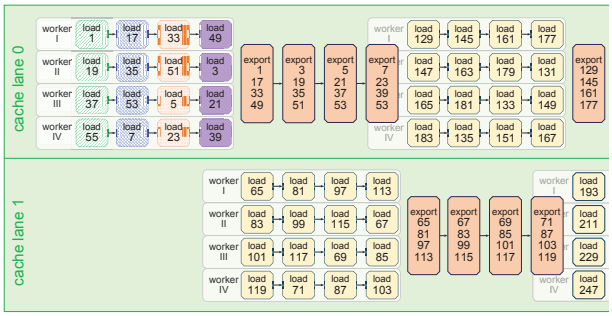


Figure 3: Continuous batch streaming with Ping-Pong cache.

while two dual-port or four single-port BRAM are required for processing each image. The pixels from four blocks are fetched in parallel as processed by four workers. Since the fetched pixels from different blocks are discontinuous, a Ping-Pong cache, which consists of two cache lanes, is adopted here for buffering. Meanwhile, the cache is also partitioned as four parts which correspond to the four BRAM ports. The fetching procedure loads the pixels in different blocks by a rotation style and feeds them to different part of cache. As shown in Fig. 3, two groups of workers on two cache lanes could alternately provide continuous batch streaming with Ping-Pong cache strategy.

3.3 Kernel Computing Module

The kernel computing module includes CalcGrad, SVM-I and NMS stage. First, we define a distance in RGB color space between pixel P_a and P_b as

$$D\left(P_a, P_b\right) = \max_{q \in \{R,G,B\}} \left|P_a\left(q\right) - P_b\left(q\right)\right|$$

The gradients on vertical direction and horizon direction are defined as $I_x\left(i,j\right)$ and $I_y\left(i,j\right)$, respectively, where i and j represents the pixel location. They could be obtained by calculating the normed gradient between neighbor pixels

$$I_x\left(i,j\right) = D\left(P_{(i-1,j)}, P_{(i+1,j)}\right)$$

$$I_y\left(i,j\right) = D\left(P_{(i,j-1)}, P_{(i,j+1)}\right)$$

and the gradient of each pixel $G\left(i,j\right)$ could be calculated by

$$G\left(i,j\right) = \min\left\{I_x\left(i,j\right) + I_y\left(i,j\right), 255\right\}$$

The obtained normed gradient of each pixel is reformulated as a two-dimension array.

In the SVM-I stage, the normed gradients $G_{8\times8}$ of every 8×8 window are formed by the gradients $G_{1\times8}$ of each row and reshaped as a 64-dimension feature with a row-wise manner.

Then the trained SVM weights W_{SVM} are adopted to perform the classification and determine the evaluation scores of each window

$$s = G_{8\times8} \cdot W_{SVM}$$

And all the scores s compose a two-dimensional array S. During the NMS stage, the max score $\max_{5\times5}$ for each 5×5 block of S is determined by finding the max score $\max_{1\times5}$ for each row first and then maximum of them. For all 5×5 blocks, only the window corresponding to $\max_{5\times5}$ will be selected for windows sequence output.

A rough approach to perform CalcGrad, SVM-I and NMS tasks usually requires a temporal two-dimensional array for intermediate data buffering which can result in waste of resources and computation cycles. In our design, these stages are reformulated and delivered on multiple pipelines for streaming processing as shown in Fig. 4. Each of the three stages has its own workspace with its own pipelines which performs operations in a streaming manner, and can be connected serially for processing batch streaming continuously. Meanwhile, the data locality in each workspace is exploited by the tiered cache, which is built with memory window and line buffer[12], by caching all the required data for batch pass synchronization.

Since the pipelines behave similarly, the SVM-I stage is taken for pipelines illustration as an example without loss of generality. As shown in Fig. 4, the input batch streaming can be processed continuously by the pipelines and consequently generate a streaming output for the following stage. During the processing of each workspace, the data are continuously loaded from tiered cache for the calculation of $G_{8\times8}$, while the calculation results are restored to the tiered cache for the next operation simultaneously.

At the end of the kernel computing module, the NMS operation usually results in non-continuous output streaming. In this work, a FIFO structure is adopted as streaming buffer to make sure the pipelines run smoothly which could improve the total efficiency of the proposed accelerator.

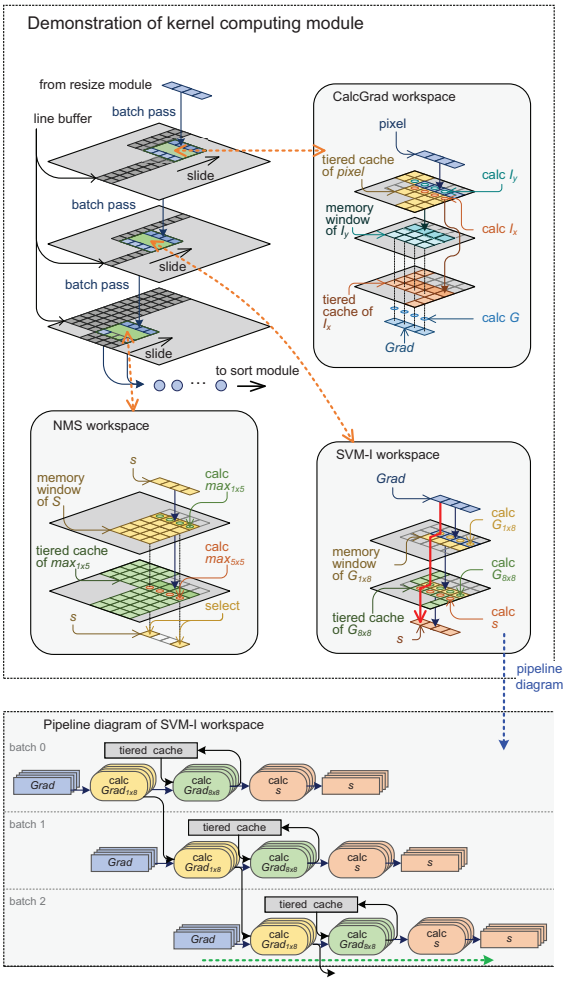


Figure 4: Demonstration of kernel computing module with the diagram of SVM-I pipeline.

4. EXPERIMENTAL RESULTS

4.1 Experiment Setup

The proposed accelerator is implemented by C language and synthesized with Xilinx Vivado HLS 2017 [13] on two target chips: Artix-7 (low voltage version) @ 3.3MHz for always-on & low power application, and Kintex UltraScale+ @ 100MHz for real-time & high performance application. A carefully quantization strategy is adopted to specify various bit-width for different data storage purpose. The synthesized resources utilizations are illustrated in Table 1. The power consumption and system latency is obtained by C-RTL co-simulation in Vivado. The VOC2007 datasets [14] are adopted to evaluate the quality of proposed windows on the metrics by detection rate (DR v.s. #WIN), and mean average best overlap (MABO v.s. #WIN), where #WIN is the number of given proposals. The metric DR v.s. #WIN means the detection rate (DR) for the given #WIN proposals [6]. And MABO v.s. #WIN means the mean average best overlap for the given #WIN proposals [7]. Hence, a larger DR or MABO value means a better proposal quality.

4.2 Performance Evaluation

A defined IoU (intersection-over-union) parameter in the previous works [7] is also adopted in our evaluation. It is a scoring function to measure the affinity of two bounding

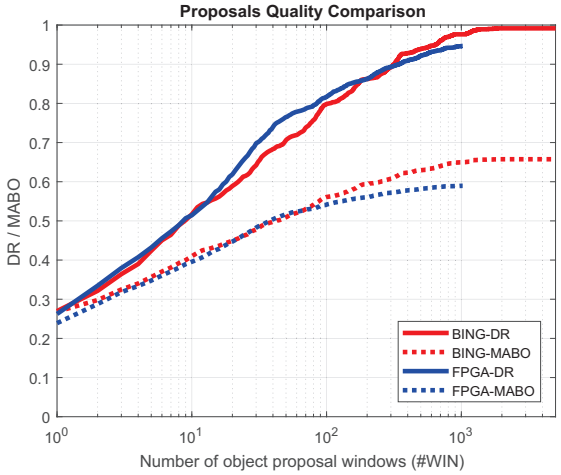


Figure 5: Quality evaluation of windows proposals by comparing BING [6] and our proposed FPGA accelerator.

boxes, defined by the intersection area of two bounding boxes divided by their union. The DR and MABO metrics are measured by varying the IoU overlap threshold which is set as 0.4 as default for correct detection.

Table 1: The FPGA resources utilizations comparison between two target devices: Artix-7 and Kintex UltraScale+.

Resources	Artix-7 Low Volt.† @ 3.3MHz		Kintex UltraScale+* @ 100MHz	
	Available	Utilized	Available	Utilized
LUT	63400	54453	162720	56504
LUT-RAM	19000	4166	99840	3157
FF	126800	48611	325440	50079
BRAM	135	135	360	146
DSP	240	25	1368	25
BUF-G	-	-	256	8

† Targeted on Artix-7 (low voltage) xc7a100tlftg256-2L.
* Targeted on Kintex UltraScale+ xcku3p-ffva676-3-e.

The detection accuracy of our proposed accelerator is compared with BING [6] as shown in Fig. 5. The BING algorithm generates 5000 object windows to evaluate the accuracy. However, we have observed it only achieves less than 3% accuracy improvement compared with only 1000 object windows generated. Hence, only 1000 object windows are proposed in our design when considering the significantly increasing requirements on hardware resources. For evaluating 1000 proposals, the detection rate of our proposed approach (denoted as FPGA-DR) is about 94.72% while BING is about 97.63%.

Table 2: Speedup and power efficiency compared with Intel i7 and ARM platforms.

	Kintex UltraScale+		Artix-7 Low Volt.	
	Speedup	Power efficiency	Speedup	Power efficiency
Intel i7	3.67X	>220X	0.12X	66X
ARM A53	68X	>250X	2.2X	>60X

Table 3: Performance evaluation between targeted on Artix-7 and Kintex UltraScale+, where P_{tot} is the total power consumption which includes static power and dynamic power consumption, and P_{dyn} is the dynamic power consumption.

Artix-7 Low Volt. @ 3.3MHz			Kintex UltraScale+ @ 100MHz		
P_{tot}	P_{dyn}	Speed	P_{tot}	P_{dyn}	Speed
$97mW$	$15mW$	$35fps$	$821mW$	$350mW$	$1100fps$

The implemented FPGA accelerators are compared with two conventional CPU platforms. The BING algorithm is well-optimized and could achieve a proposal speed by $300fps$ on Intel i7-3940XM CPU (TDP: 55W) platform with multi-threaded programming and subword parallelism techniques [6]. We also evaluate BING on a Raspberry-Pi 3B platform with 64-bit ARM A53 processor, which could achieve $16fps$ speed and 3W∼4W [15] power consumption. However, the proposed accelerator on Kintex UltraScale+ target could achieve a proposal speed by $1100fps$ while the power consumption is only $821mW$ when running at 100MHz, which is applicable for real-time processing of multi-camera sensor fusion applications. Hence, it could achieve about 3.67X speedups and >220X energy efficiency improvement compared with BING on Intel i7 CPU. Furthermore, the proposed accelerator targeted on low voltage Artix-7 could achieve $35fps$ with an extremely low power consumption $97mW$ when running at 3.3MHz, which is attractive for ultra-low power applications with always-on working mode whose speed is also sufficient for most of the applications. The detailed comparison results of our proposed FPGA accelerators against the two CPU platforms are shown in Table 2.

5. CONCLUSIONS

Efficient region proposal generation is a critical task for object detection. A scalable dataflow accelerator is proposed in this paper for efficient region proposals on FPGA platform. The proposal procedures in BING algorithm are reformulated as a dataflow driven manner and implemented on multiple pipelines architecture. All of the modules in the accelerator are organized by streaming processing mechanisms so that these streaming data could guarantee the pipelines are fully loaded. Furthermore, a tiered cache system is utilized to improve the bandwidth of data synchronization between different processing stages. Evaluations on VOC2007 datasets show that our proposed accelerator achieves a very large scale of speedups and energy efficiency improvement with a little detection rate degradation.

6. REFERENCES

[1] Alex Krizhevsky et al. Imagenet classification with deep convolutional neural networks. In *Advances in NIPS*, pages 1097–1105, 2012.

[2] Ziming Zhang et al. Group membership prediction. In *Proceedings of the IEEE ICCV*, pages 3916–3924, 2015.

[3] Gregory Castañón et al. Efficient activity retrieval through semantic graph queries. In *Proceedings of the 23rd ACM Multimedia*, pages 391–400, 2015.

[4] Joel Emer et al. Tutorial on hardware architectures for deep neural networks. *http://eyeriss.mit.edu/tutorial.html*, 2016.

[5] Wei Liu et al. Ssd: Single shot multibox detector. In *Proceedings of ECCV*, pages 21–37, 2016.

[6] Ming-Ming Cheng et al. BING: Binarized normed gradients for objectness estimation at 300fps. In *Proceedings of the IEEE CVPR*, pages 3286–3293, 2014.

[7] Ziming Zhang et al. Sequential optimization for efficient high-quality object proposal generation. *IEEE TPAMI*, 2017.

[8] Yunchao Wei et al. Hcp: A flexible cnn framework for multi-label image classification. *IEEE TPAMI*, 38(9):1901–1907, 2016.

[9] Shengxin Zha et al. Exploiting image-trained cnn architectures for unconstrained video classification. *BMVC*, 2015.

[10] Wojciech M Zabołotny. Dual port memory based heapsort implementation for FPGA. *Proceedings of SPIE*, 2011.

[11] Nitish Kumar Srivastava et al. Accelerating Face Detection on Programmable SoC Using C-Based Synthesis. In *Proceedings of ACM/SIGDA FPGA*, pages 195–200, 2017.

[12] Fernando Martinez Vallina. Implementing memory structures for video processing in the Vivado HLS tool. *XAPP793 (v1. 0), September*, 20, 2012.

[13] Xilinx Inc. Vivado HLx 2017. *https://www.xilinx.com/products/design-tools/vivado.html*.

[14] Mark Everingham et al. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. *http://www.pascalnetwork.org/challenges/VOC/voc2007/workshop/index.html*.

[15] Raspberry Pi Power Consumption Benchmarks . *https://www.pidramble.com/wiki/benchmarks/power-consumption*.