Area-Time Efficient Streaming Architecture for FAST and BRIEF Detector

Siew-Kei Lam, *Member, IEEE*, Guiyuan Jiang, Meiqing Wu, and Bin Cao

***Abstract*—The combination of FAST corners and BRIEF descriptors provide highly robust image features. We present a novel detector for computing the FAST-BRIEF features from streaming images. To reduce the complexity of the BRIEF descriptor, we employ an optimized adder tree to perform summation by accumulation on streaming pixels for the smoothing operation. Since the window buffer used in existing designs for computing the BRIEF point-pairs are often poorly utilized, we propose an efficient sampling scheme that exploits register reuse to minimize the number of registers. Synthesis results based on 65-nm CMOS technology show that the proposed FAST-BRIEF core achieves over 40% reduction in area-delay product compared to the baseline design. In addition, we show that the proposed architecture can achieve 1.4x higher throughput than the baseline architecture with slightly lower energy consumption.**

***Index Terms*—Feature descriptor, embedded vision, VLSI, hardware acceleration**

I. Introduction

COMPUTER vision algorithms such as Simultaneous Localization and Mapping (SLAM), object detection, object matching and image correspondence, rely on image features [1]. The features are used for object representation, or directly matched and tracked across multiple frames. Due to the real-time nature of these applications [2], the computational complexity for detecting the features must be kept low.

The Features from an Accelerated Segment Test (FAST) algorithm was introduced to detect reasonable corners at high speed. The Binary Robust Independent Elementary Feature (BRIEF) descriptor is computed by performing binary tests of *n* point-pairs on a square image patch. The combination of FAST and BRIEF (FAST-BRIEF) [3][4] has been shown to outperform other image features in many applications [5]. FAST-BRIEF features provide a good trade-off between robustness and compute efficiency. However, it still contributes to a significant portion of the overall application runtime. For example, the FAST-BRIEF detector contributes to over a third of the total runtime of a real-time tracking process [5]. This has motivated several hardware implementations for the FAST-BRIEF detector. Existing hardware realizations can be categorized into non-stream and stream processing.

Non-stream processing architectures such as [6]-[9] assumes the availability of a frame buffer that stores the image frame. This enables the BRIEF descriptors to be computed only for patches that are centered on the detected FAST corners. The work in [6][7] presented a FPGA implementation of an object recognition system that relies on FAST-BRIEF features, which are detected at multiple scales in parallel [6]. The 256-bit BRIEF descriptor of each corner is computed by performing sequential binary test on the corresponding 256 point-pairs. The FAST-BRIEF implementation in [6] achieves a frame rate of about 55 FPS for a 640x480 image at a clock operating frequency of 100MHz. A FAST architecture using a string matching algorithm was proposed in [8]. This implementation requires an external memory for frame buffering and a mechanism for sequencing the input data in the form of a 1D text for string matching. Non-stream processing architectures such as [6]-[9] suffer from the need of large memory resources.

On the contrary, stream processing architectures (e.g. [10]-[13]) are extremely attractive as they do not require external memories for storing input video frames and can achieve high throughput (since the memory fetch and store latencies are avoided). However, in stream processing architectures where the incoming pixels are processed on-the-fly without being stored in frame buffers, the BRIEF descriptors need to be computed for all pixel locations. This is performed in parallel with the FAST corner detection, and only the BRIEF descriptors corresponding to detected corners are used as outputs. Stream-based architectures for computing the BRIEF descriptor need many parallel processing elements to keep up with the rate of incoming pixel streams. For example, the work in [10] employs a 33x33 register file for storing the patch and utilizes 256 parallel comparators to compute the 256-bit descriptor of each patch in a single clock cycle. The FPGA implementation of the BRIEF descriptor in [10] achieves 325 FPS for a 640x480 image at a clock frequency of 100MHz. The stream architectures in [11]-[13] also utilizes parallel comparators for computing the BRIEF descriptors as in [10].

II. Main Contributions

Our work focuses on stream processing architecture for FAST-BRIEF detector. Existing stream processing architectures rely on large computational resources for parallel processing of the smoothing operations and binary tests. We present two novel hardware design strategies that reduces the hardware complexity for computing the BRIEF descriptor:

S.-K. Lam, G. Jiang, M. Wu, and B. Cao are with School of Computer Science and Engineering, Nanyang Technological University, 50 Nanyang Avenue, Singapore (e-mail: siewkei_lam@pmail.ntu.edu.sg).

This research project is funded by the National Research Foundation Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme.

- An optimized adder tree for the smoothing operation that significantly reduces the number of adders through summation by accumulation of continuous input pixels.
- An optimized sampling scheme that takes advantage of register reuse for computing the BRIEF point-pairs, which led to significant reduction in the number of registers.

III. Algorithm and Baseline Implementation

A. FAST-BRIEF Algorithm

The FAST corner detector uses an efficient method to classify a pixel as a corner by considering pixels on the surrounding circle. The BRIEF descriptor is computed on an image patch P (centered at a FAST corner) of size $m \times m$ with n (where n is typically 128, 256, or 512) number of (s, d) point pairs, where a binary test is performed on each of the point-pairs. The binary test τ is shown in Eq. (1), where $P(s)$ is the pixel intensity in P at location $s = (x, y)^T$. The resulting BRIEF descriptor for patch P is a n-dimensional bit-string as shown in (2).

$$\tau(P; s, d) = \begin{cases} 1 & if\ P(s) < P(d) \\ 0 & otherwise \end{cases} \qquad (1)$$

$$B(P) = \sum_{1 \le i \le n} 2^{i-1} \cdot \tau(P; s_i, d_i) \qquad (2)$$

$$A(s) = \frac{\left(\sum_{i=-\lfloor r/2 \rfloor}^{\lfloor r/2 \rfloor} \sum_{j=-\lfloor r/2 \rfloor}^{\lfloor r/2 \rfloor} I(x+i, y+j)\right)}{r^2} \qquad (3)$$

The choice of the n binary test locations in Eq. (2) (i.e. (s_i, d_i)) can be customized or learned from the image scenes of the application. Fig. 1 shows four approaches (GI to GIV) for choosing the test locations [3]. Since intensity comparison is sensitive to noise, a smoothing step is performed on the image patch P prior to computing the BRIEF descriptor. The smoothing operation is typically achieved using a box filter of size $r \times r$. Eq. (3) shows the smoothing (averaging) operation at location $s = (x, y)^T$ in image I.

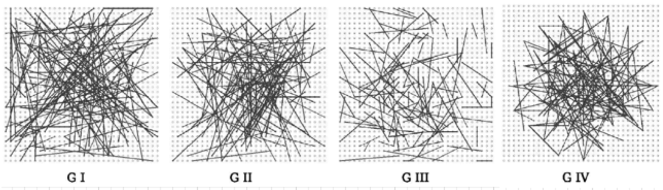

Fig. 1. Different approaches for choosing the test locations [3].

B. Baseline Implementation

We have adapted the FAST-BRIEF design in [10] as our baseline since it is the only stream processing architecture in the literature that provided sufficient implementation details. The baseline implementation relies on a patch size of 33x33 (i.e. m = 33) to compute a 256-bit BRIEF descriptor (i.e. n = 256). A 7x7 box filter (i.e. r = 7) is used to smooth the pixel intensities in the patch prior to performing the binary tests. We have relied on the more commonly-used box filter for the smoothing operation (which is consistent with the original BRIEF algorithm) while [10] employs a circular averaging filter.

Fig. 2 shows an overview of the FAST-BRIEF architecture of the baseline and proposed implementation, which consists of the FAST-BRIEF core and row buffers. Under the assumption that the image is read sequentially using a raster scan mode at a rate of one pixel per clock cycle, the incoming pixels need to be cached locally using a set of row buffers. 6 row buffers are concatenated in the form of FIFO (First-In, First-Out) delay buffers to cache the incoming pixels. The size of each row buffer is equivalent to the horizontal resolution of the image L (for example L = 640 for 640x480 image), and hence each row buffer effectively delays the input by one row.

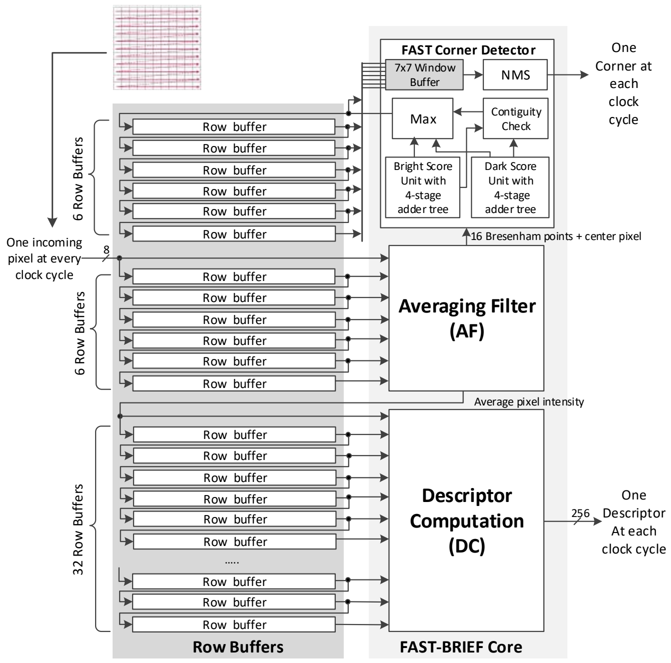
FAST Corner Detector
7x7 Window Buffer
NMS
Max
Contiguity Check
Bright Score Unit with 4-stage adder tree
Dark Score Unit with 4-stage adder tree
One Corner at each clock cycle
6 Row Buffers
Row buffer
One incoming pixel at every clock cycle
8
16 Bresenham points + center pixel
Averaging Filter (AF)
Average pixel intensity
32 Row Buffers
Descriptor Computation (DC)
256
One Descriptor At each clock cycle
Row Buffers
FAST-BRIEF Core

Fig. 2. Overview of FAST-BRIEF architecture.

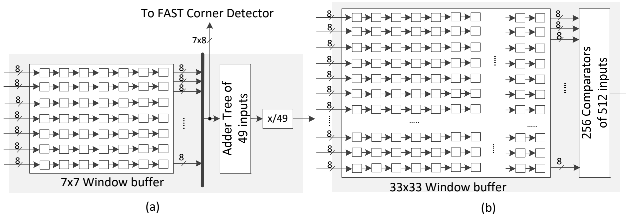

Fig. 3(a). Averaging filter (AF) of the baseline implementation, (b) Descriptor computation (DC) of the baseline implementation.

In the baseline implementation, the k-bit (k = 8 for grayscale images) pixels at the tail end of each row buffer are shifted into a set of registers, called the window buffer (see Fig. 3(a)). These pixels will be simultaneously processed by the FAST corner detector and averaging filter (AF). The hardware implementation of the FAST corner detector is same as [10]. Note that the FAST corner detector requires another 6 row buffers to perform the Non-Maximal Suppression (NMS).

The AF performs the smoothing operation, which requires a 6-level adder tree ($\lceil log_2(49) \rceil$) and a divider. Note that [10] did not provide the detail implementation of AF and hence, we have used the box filter design in [14]. The average value is then shifted into another set of row buffers as shown in Fig. 2. Since the BRIEF descriptor is computed on a 33x33 patch, we require another 32 FIFO delay buffers to cache the average pixel values. The tail end of these buffers is shifted into the 33x33 window buffer which enables the BRIEF Descriptor Computation (DC) module to perform the 256 binary tests in parallel using 256 comparators (see Fig. 3(b)). The operations in the architecture are pipelined to maintain maximum throughput. The outputs of the FAST, AF and DC modules are registered. A new FAST corner and a 256-bit BRIEF descriptor will be produced at the rate of the incoming pixel. The critical path delay of the baseline BRIEF core lies in the divider in AF.

IV. Proposed Architecture

The proposed architecture adopts the same computational blocks and row buffers as shown in Fig. 2. Like the baseline implementation, a new FAST corner and a 256-bit BRIEF descriptor will be produced at the rate of the incoming pixel. The FAST implementation is the same as the baseline. Next, we describe the proposed designs for AF and DC.

A. *Proposed Averaging Filter (AF)*

The AF in the baseline implementation performs a 7x7 box filter operation that requires the accumulation of 49-pixel intensities before they are divided by a constant. Instead of relying on a large adder tree, the proposed AF uses minimal resources to accumulate continuous pixel stream in the 7x7 window patch in two steps as shown in Fig. 4. First, a pipelined 4-level adder tree is employed to accumulate the row pixel values in the input register (*Row Add*). Due to the continuous input pixel stream, a new accumulated row pixel value is produced in each clock cycle. In the second step, the accumulated row pixel values are summed using a single adder (*Column Add*). Since the final accumulated value of 7 columns are required, a FIFO is used to delay the values that will be subtracted from the current accumulated row pixel value at each clock cycle. The proposed AF leads to 2x lesser registers and more than 4x lesser adders that the implementation in [14].

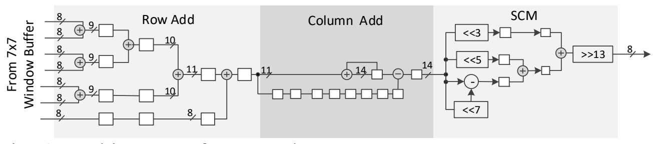

Fig. 4. Architecture of proposed AF.

Finally, the proposed AF utilizes a pipelined Single Constant Multiplication (SCM) that multiplies the final accumulated value by 1/49 to compute the average value of 7x7 pixels. The SCM design is based on [15] with 15 fractional bit precision. The critical path delay of the proposed AF is only one adder delay. Compared to the baseline AF, the proposed design leads to lower number of hardware resources and critical path delay.

B. *Proposed Descriptor Computation (DC)*

The baseline DC employs a 33x33 window buffer to enable 256 binary tests in parallel. However at any one time, only 512 of the 1089 (33x33) elements of the window buffer are used for computations which results in high redundancy. The proposed sampling scheme takes advantage of the pre-determined BRIEF sample patterns to customize the storage, computational resources and interconnectivity in DC, such that maximal utilization is achieved while maintaining the required throughput. We will describe the proposed DC based on example in Fig. 5(a) with an 8x8 image patch and 6 BRIEF point-pairs. Let's denote the 6 BRIEF point-pairs in Fig. 5(a) as *N1*, *N2*, …, *N6*.

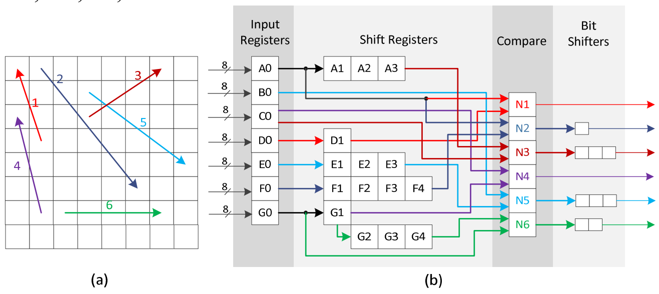

Fig. 5. (a) An 8x8 image patch with 6 BRIEF point-pairs (i.e. m = 8 and n = 6, where the arrow denotes $s \rightarrow d$), (b) Proposed DC.

The proposed DC (Fig. 5(b)), comprises of four stages:

- *Input Registers*: Unlike the baseline implementation, the BRIEF computation is performed on the averaged pixel values as soon as they are available from the row buffers. Only a single column of input registers (i.e. *A0*, *B0*, …, *G0*) is required. Note that each input register can be used in the binary tests of more than one BRIEF point pair. For example, *A0* can be simultaneously used for the binary tests of *N1*and *N2* (for different image patches). Only 7 input registers are required (instead of 8) as none of the averaged pixel values in the last row of Fig. 7 is used for binary tests. Compared to the 8x8 window buffer in the baseline implementation, the input registers in the proposed DC are more efficiently utilized for BRIEF computation.
- *Shift Registers*: The purpose of the shift registers is to delay the averaged pixel values such that the correct pair of *P(s)* and *P(d)* values are used in the binary tests. For example, for the binary test of *N3*, the corresponding *P(d)* that arrives at *A0* must be delayed by 3 clock cycles (via shift registers *A1 - A3*) before the corresponding *P(s)* is available at *C0*. Only when both *P(s)* and *P(d)* of *N3* are available, the binary test (Eq. (1)) can be performed. The number of shift registers required for each binary test therefore depends on the column offset of each *s-d* pair. Note that we have employed register sharing to further reduce the number of shift registers. For example, *G1* is used to delay *P(s)* of *N4* and *P(d)* of *N6*.
- *Compare*: In this stage, the corresponding BRIEF point-pair values (*P(s)* and *P(d)*) are available, and the binary test (Eq. 1) can be performed. Like the baseline implementation, 6 comparators are used to perform the 6 binary tests in parallel.
- *Bit Shifters*: Unlike the baseline implementation, the BRIEF point-pairs that are being evaluated at the *Compare* stage may not belong to the same image patch. This is because each averaged pixel value in the *Input Registers* could be used to simultaneously compute multiple binary tests of different image patches. As such, the output of the *Compare* stage must be synchronized to produce the BRIEF descriptor that corresponds to each image patch. This is achieved by using bit shifters (since the output of each binary test is a bit). The number of bit shifters depends on the column offset from the leftmost column of the image

patch to the column that s or d (whichever comes first) resides on. For example, there are two bit-shifters for $N6$, since s of $N6$ resides on the 3rd column of the image patch in Fig. 5(a).

The number of registers required by the proposed DC for the example in Fig. 5(a) is about 23 (normalized to 8-bit registers) compared to 64 which is required by the baseline DC. It is evident that the proposed design can lead to significantly lower number of registers, however this depends on the configuration of the BRIEF point-pairs. Specifically, the number of required shift registers depends on n (size of the descriptor), the total column offsets between the BRIEF point pairs, and the opportunities for register sharing. In addition, the number of bit shifters depends on n and the distance of s or d from the leftmost column of the image patch. We will show that the proposed DC leads to notably lesser area utilization in most cases.

V. Results and Discussion

Table 1 shows the resource analysis for the baseline and proposed AF (Baseline-AF and Proposed-AF). For the Baseline-AF, the number of registers and adders required by the 6-level adder tree with 49 inputs is 52 and 48 respectively. In addition, Baseline-AF requires a divider. As can be observed from Fig. 4, Proposed-AF requires only 26 registers (including output register) and 11 adders. Note that we have not considered the registers for the 7x7 window buffer that is required for both the baseline and proposed method. It is evident that the resources that are required by the proposed implementation is lesser than the baseline as the accumulation of the 49-pixel values is achieved in two stages, *Row Add* and *Column Add* (see Fig. 4), to accumulate the row and column values separately.

Table 2 shows the number of registers for the proposed DC modules (Proposed-DC-GI to Proposed-DC-GIV). The number of registers for the baseline design is fixed at 1121 (i.e. 33x33 + 256/8), while the number of registers for the proposed design is dependent on the BRIEF point-pair configurations. GI - GIV refers to the different BRIEF point-pair configurations in Fig. 1. We implemented a software program to automatically generate optimized sampling scheme for the various BRIEF-point-pair configurations. In particular, the program generates the shift registers with resource sharing, the bit shifters, and the custom interconnect as described earlier.

TABLE 1: RESOURCE FOR BASELINE AND PROPOSED AF

	Baseline-AF	**Proposed-AF**
REG	52	26
Adder	48	11
Divider	1	-

TABLE 2: REGISTERS FOR PROPOSED DC (NUMBER OF REGISTERS FOR BASELINE DC IS FIXED AT 1121)

	Input Registers	**Shift Registers**	**Bit Shifters**	**Total**
Proposed-DC-GI	33	773	392	**1198**
Proposed-DC-GII	33	433	416	**882**
Proposed-DC-GIII	33	170	529	**732**
Proposed-DC-GIV	33	478	406	**917**

We only report the number of registers in Table 2 as both the baseline and proposed design require the same number of comparators (i.e. 256). Note that the registers in Table 2 are normalized to 8-bit registers. It can be observed that apart from the GI configuration, the proposed design requires notably lesser number of registers. Proposed-DC-GI has slightly more registers than the baseline due to the need for larger number of shift registers to accommodate the BRIEF point-pairs with larger column distances (see Fig. 1). These results demonstrate that in general, the proposed optimized sampling scheme can effectively take advantage of register reuse and reduce the overall registers for computing the BRIEF point-pairs.

The baseline and proposed architectures were implemented using Verilog and synthesized with Synopsys DC targeting the 65-nm CMOS technology library. The designs were synthesized to achieve maximum clock frequency. Table 3 shows the area-delay synthesis results for the baseline and proposed AF and DC modules. The area utilization of the proposed AF module is about 36% lower than the baseline AF. In addition, as discussed in the previous section, the critical path delay of the baseline AF is governed by the latency of the divider, which is larger than the critical path delay of the proposed AF (i.e. latency of one adder).

The baseline DC implementation for GI – GIV BRIEF point-pair configurations have similar critical path delay and area utilization. This is expected as the baseline DC design is the same for all the configurations. The critical path delay of the baseline and proposed DC is similar as they are both governed by the latency of the comparator. However, unlike the baseline implementation, the proposed DC exhibits differing area utilization for the various BRIEF point-pair configurations. These results are consistent with our resource analysis in Table 2, where the area utilization of Proposed-DC-GI is slightly higher than the baseline DC, but for the rest of the BRIEF point-pair configurations, we can observe a significant reduction in area utilization. The proposed DC implementation for GI – GIV BRIEF point-pair configurations achieves -2.5%, 20.4%, 30%, and 17.4% area reduction when compared to the corresponding baseline implementation. These results clearly show the effectiveness of the proposed pipelining approach for DC.

Table 4 reports the synthesis results for the FAST-BRIEF core (FAST, AF and DC) of the baseline and proposed designs. It can be observed that the proposed design achieves an average reduction of 17% in area utilization over the baseline design. The last column of Table 4 shows the percentage Area-Delay Product (ADP) reduction of the proposed FAST-BRIEF core over the baseline. On average, the ADP reduction of the proposed design over the baseline design is over 40%.

Table 5 shows the power consumption and energy per pixel of the baseline and proposed streaming FAST-BRIEF architecture for 640x480 images. This requires 44 row buffers, where the size of each row buffer is 640. Energy per pixel is computed as the product of power (mW) and minimum clock period (ns). Although the power consumption of the baseline architecture is lower than the proposed (due to the lower operating frequency), the energy per pixel of both architectures have marginal difference.

TABLE 3: AREA-DELAY SYNTHESIS RESULTS OF AF AND DC BASED ON 65-NM CMOS TECHNOLOGY LIBRARY

Design	Baseline					Proposed				
	Delay (ns)	Area (sq um)			Area x Delay	Delay (ns)	Area (sq um)			Area x Delay
		Comb	Non-Comb	Total			Comb	Non-Comb	Total	
FAST	1.3	8022	6421	14443	18775.9	Same as Baseline				
AF	2.1	4703	2822.4	7525.4	15803.3	0.7	2124.36	2645.28	4769.64	3338.7
DC-GI	0.7	13311.0	65100.9	78411.9	54888.3	0.7	12686.8	67720.3	80407.1	56284.9
DC-GII	0.7	14492.5	65179.1	79671.6	55770.1	0.7	12854.9	50528.9	63383.8	44368.7
DC-GIII	0.7	12355.2	64907.6	77262.8	54083.9	0.7	11923.9	42154.9	54078.8	37855.1
DC-GIV	0.7	12817.1	65009.5	77826.6	54478.6	0.6	11850.5	52460.6	64311.1	38586.7

TABLE 4: AREA-DELAY SYNTHESIS RESULTS OF FAST-BRIEF CORE BASED ON 65-NM CMOS TECHNOLOGY LIBRARY

	Baseline			Proposed				
	Delay (ns)	Area (sq um)	Area x Delay	Delay (ns)	Area (sq um)	Area x Delay	Area Reduction (%)	ADP Reduction (%)
FAST-BRIEF (GI)	2.1	100403.7	210847.7	1.5	97352.7	146029.0	3.0	30.7
FAST-BRIEF (GII)		100578.7	211215.2		80617.7	120926.5	19.8	42.7
FAST-BRIEF (GIII)		99957.7	209911.1		72455.7	108683.5	27.5	48.2
FAST-BRIEF (GIV)		100674.7	211416.8		82574.7	123862.0	18.0	41.4

TABLE 5: POWER, ENERGY-PER-PIXEL AND THROUGHPUT (FPS) OF FAST-BRIEF ARCHITECTURE FOR 640X480 IMAGES

	Baseline			Proposed		
	Power (mW)	Energy/ Pixel	FPS (640x480)	Power (mW)	Energy/ Pixel	FPS (640x480)
GI	823.5	1729.35	1550	1150	1725	2170
GII	823.6	1729.56		1137	1705.5	
GIII	823.5	1729.35		1131	1696.5	
GIV	823.6	1729.56		1138	1707	

The proposed architecture achieves about 1.4X higher throughput in terms of frame per seconds (FPS) over the baseline architecture. Note that the FPS for both the baseline and proposed architectures remain the same for different BRIEF configurations, as the critical path delays are the same (see Table 4).

We have also performed accuracy evaluation based on the repeatability criterion using the image dataset from [16]. The repeatability criterion is based on the notion that robust descriptors should be invariant to imaging conditions. The overall difference in repeatability between the proposed and baseline architectures is negligible, which demonstrate that the proposed architecture has similar degree of robustness compared to the baseline architecture.

VI. CONCLUSION

The proposed FAST-BRIEF architecture incorporates hardware optimizations for the smoothing operations and binary tests of the BRIEF point pairs. A novel pipelining approach was introduced to compute the binary tests of multiple BRIEF point pairs from each incoming pixel, and then using bit shifters to synchronize the outputs of the BRIEF descriptors. The proposed design achieves significant area-delay and throughput benefits over the conventional design.

REFERENCES

[1] S. Gauglitz, T. Hollerer, and M. Turk, "Evaluation of interest point detectors and feature descriptors for visual tracking", *International Journal of Computer Vision*, Vol. 94, pp. 335-360, 2011.

[2] J. Soh and X. Wu, “An FPGA-Based Unscented Kalman Filter for System-On-Chip Applications”, *IEEE Transactions on Circuits and Systems II*, Vol. 64, No. 4, pp. 447-451, April 2017.

[3] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, “BRIEF: Binary Robust Independent Elementary Features”, *European Conference o Computer Vision*, pp. 778-792, 2010.

[4] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua, “BRIEF: Computing a Local Binary Descriptor Very Fast”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 34, No. 7, July 2012, pp. 1281-1298.

[5] R. Mur-Artal, J.M.M. Montiel, and J.D. Tardos, “ORB-SLAM: A Versatile and Accurate Monocular SLAM System”, *IEEE Transactions on Robotics*, Vol. 31, No. 5, October 2015, pp. 1147-1163

[6] H. Heo, J.-Y. Lee, K.-Y. Lee, and C.-H. Lee, “FPGA-based Implementation of FAST and BRIEF Algorithm for Object Recognition”, IEEE TENCON, 2013.

[7] C.-S. Seo, S.-Y. Kim, J.-P. Ko, H.-J. Chung, Y.-H. Lee, “Feature Point Detection Hardware Based on Fast and Brief Algorithm”, International Journal of Management and Applied Science (IJMAS), Vol. 3, No. 1, 2015, pp. 81-84.

[8] J.S. Park, H.E. Kim, and L.S. Kim, “A 182mW 94.3 f/s in full HD pattern-matching based image recognition accelerator for an embedded vision system in 0.13-µm CMOS technology”, *IEEE Trans. Circuits and Systems for Video Technology*, Vol. 23, No. 5, pp. 832-845, 2013

[9] W. Farhat, H. Faiedh, and C. Souani, “Real-time Embedded System for Traffic Sign Recognition based on ZedBoard”, Journal of Real-Time Image Processing, April 2017, pp. 1-11.

[10] M. Fularz, M. Kraft, A. Schmidt and A. Kasinski, "A High-Performance FPGA-based Image Feature Detector and Matcher based on the FAST and BRIEF Algorithms", *International Journal of Advanced Robotic Systems*, Vol. 12, 2015.

[11] F. Brenot, J. Piat, and P. Fillatreau, “FPGA-based Hardware Acceleration of a BRIEF Correlator Module for a Monocular SLAM Application”, Proceedings of the 10th International Conference on Distributed Smart Camera, 2016, pp. 184-189.

[12] W. Fang, Y. Zhang, B. Yu, and S. Liu, “FPGA-based ORB Feature Extraction for Real-Time Visual SLAM”, Available: https://arxiv.org/abs/1710.07312.

[13] R. de Lima, J. M-Carranza, A. M-Reyes, and R. Cumplido, “Improving the Construction of ORB through FPGA-based Acceleration”, *Machine Vision and Applications*, Vol. 28, pp. 525-537.

[14] R. Shenghui, Z. Huixin, W. Zhigang, Q. Hanlin, Q. Kun, and C. Kuanhong, “An Improved Non-Uniformity Correction Algorithm and its Hardware Implementation on FPGA”, *Infrared Physics and Technology*, 85, 2017, pp. 410-420.

[15] Y. Voronenko and M. Püschel, “Multiplierless Multiple Constant Multiplication”, *ACM Transactions on Algorithms*, Vol. 3, No. 2, 2007.

[16] Affine Covariant Features. Available: http://www.robots.ox.ac.uk/~vgg/research/affine/