Area-Time Efficient FAST Corner Detector using Data-path Transposition

Siew-Kei Lam, *Member, IEEE*, Teck Chuan Lim, Meiqing Wu, Bin Cao, and Bhavan A. Jasani

***Abstract*—Corner detection plays an essential role in many computer vision applications e.g. object recognition, motion analysis and stereo matching. In this paper, we present a novel data-path transposition strategy for the hardware design of the FAST corner detector. The proposed design transposes the data-path of the conventional architecture to enable partial evaluation of multiple corners in a pipelined manner, which reduces the size of the window buffer. Further area savings were achieved by combining the operations for computing the corner scores and determining the member vectors. We show that the proposed design on 180-nm CMOS technology leads to about 22% reduction in the critical path delay and lesser area compared to the previously reported architecture, without notable difference in energy consumption.**

***Index Terms*—corner detection; hardware accelerator; data-path; ASIC; FPGA; embedded vision**

I. Introduction

REAL-time computer vision algorithms are extensively used in a wide range of applications such as vision-based navigation of unmanned vehicles and robots, object tracking, visual SLAM (Simultaneous Localization and Mapping), stereo matching, and ensuring safety in environments with sharp moving objects [1]-[5]. A fundamental step in these applications is the detection of corners which represent identifiable anchor points in the image.

Several hardware designs have been recently proposed for the FAST (Features from Accelerated Segment Test) corner detection [6]-[9]. Existing techniques often exploit the inherent parallelism in the corner detectors to achieve high throughput. However, the computational flow of these architectures remains largely unchanged and there has been little effort undertaken to investigate alternative data-path structures that can lead to higher gains in area-time.

In this paper, we present a novel hardware design strategy using data-path transposition to realize a pipelined FAST corner detector [10]-[11] architecture that does not require intermediate full frame buffering. Instead of employing a large 7x7 window buffer for examining a single pixel at each clock cycle, which is the typical method adopted in existing works [6]-[8], we propose to use a smaller 7x3 window buffer and transpose the data-path to enable multiple pixels to be examined concurrently. This enables the number of registers to be reduced. The combinational area is further reduced by adopting two's complement adders to simultaneously compute the corner scores and determine the member vectors. Synthesis results based on 180-nm CMOS technology show that the proposed architecture leads to about 22% area-time product reduction compared to the existing architecture, without notable differences in energy consumption.

In Section 2, we briefly discuss the existing work in accelerating the FAST corner detector. We then describe the FAST algorithm and the baseline hardware implementation in Section 3. Section 4 presents the proposed design. The synthesis results are shown in Section 5 to demonstrate the benefits of our approach and Section 6 concludes the paper.

S.-K. Lam, T.C. Lim, M. Wu, B. Cao, and B.A. Jasani is with School of Computer Science and Engineering, Nanyang Technological University, 50 Nanyang Avenue, Singapore (e-mail: siewkei_lam@pmail.ntu.edu.sg).

II. Related Work

The FAST algorithm was first presented in [10] and later improved in [11]. The improved version employs machine learning to build a decision tree from a set of training images for classifying corners on future images. The work in [9] presented an FPGA implementation of the machine learned FAST algorithm that is based on a binary look-up table. A FAST architecture using a string matching algorithm was proposed in [12][13]. 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. The work in [6] presented an FPGA implementation of the original FAST algorithm that does not require intermediate frame buffering. This leads to significant area savings as the architecture can directly process the pixels from the camera output with a simple interface. A similar FPGA architecture that runs on 50MHz operating frequency was presented in [7], which showed significant performance gain over the software implementation running on a 1GHz mobile phone.

FAST has also been utilized as a preliminary step for computing feature descriptors such as ORB (Oriented FAST and Rotated BRIEF) [13], which are used for a wide range of applications, e.g. object recognition, visual SLAM, image representation, motion tracking, etc. Recently, the work in [8] presented an FPGA architecture of the FAST feature detector and BRIEF feature descriptor which can process images of resolution 1280x720 at 109 frames per second.

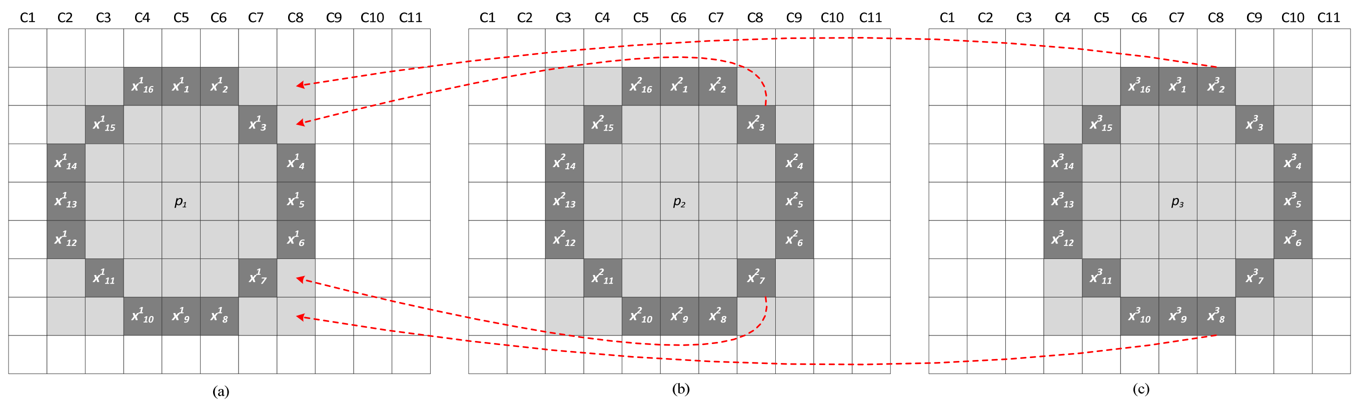
C1 C2 C3 C4 C5 C6 C7 C8 C9 C10 C11
x^1_{16} x^1_1 x^1_2 x^1_3 x^1_4 x^1_5 x^1_6 x^1_7 x^1_8 x^1_9 x^1_{10} x^1_{11} x^1_{12} x^1_{13} x^1_{14} x^1_{15}
p_1
(a)
x^2_{16} x^2_1 x^2_2 x^2_3 x^2_4 x^2_5 x^2_6 x^2_7 x^2_8 x^2_9 x^2_{10} x^2_{11} x^2_{12} x^2_{13} x^2_{14} x^2_{15}
p_2
(b)
x^3_{16} x^3_1 x^3_2 x^3_3 x^3_4 x^3_5 x^3_6 x^3_7 x^3_8 x^3_9 x^3_{10} x^3_{11} x^3_{12} x^3_{13} x^3_{14} x^3_{15}
p_3
(c)

Fig. 1: Testing for corners at pixel (a): p_1 at time t, (b) p_2 at time t+1, and (c) p_3 at time t+2. The dotted lines show that at time t, the pixels in column C8 of the window (light gray) can be used for partial examination of pixels p_1, p_2, and p_3.

III. Baseline Architecture of FAST Corner Detector

The implementations of the original FAST algorithm presented in [6]-[8] adopt similar computational blocks which will be used as our baseline architecture. The original FAST algorithm proposed in [10] tests for a corner at each pixel p_i in an image frame by examining the Bresenham circle of 16 pixels around p_i. A 7x7 window buffer centered on p_i is used to enable parallel examination of the 16 surrounding pixels to facilitate the testing of one pixel per clock. Let x_j^i, where $j = 1,2,\dots,16$, be the pixels on the Bresenham circle that are used in the corner test of pixel p_i.

Fig. 1 shows the 7x7 window (light gray) for corner testing of pixel p_i and the corresponding x_j^i pixels (dark gray) at time t, t+1 and t+2. Each pixel x_j^i in the window is evaluated in parallel with pixel p_i to generate two 16-bit member vectors i.e. bright (m_B^i) and dark members (m_D^i):

$$m_B^i = \begin{bmatrix} x_1^i > T_H^i \\ x_2^i > T_H^i \\ \vdots \\ x_{16}^i > T_H^i \end{bmatrix}, \; m_D^i = \begin{bmatrix} x_1^i < T_L^i \\ x_2^i < T_L^i \\ \vdots \\ x_{16}^i < T_L^i \end{bmatrix} \qquad (1)$$

where $T_H^i = p_i + t$, $T_L^i = p_i - t$, and t is a predefined threshold. Each element in the member vector is set to '1' if the corresponding condition is true, otherwise it is set to '0'. The scores for the bright and dark members ($score_B^i$ and $score_D^i$ respectively) are then calculated as shown in Eq. (2).

$$score_{B/D}^i = \left(m_{B/D}^i\right)^T \cdot \begin{bmatrix} x_1^i/T_L^i - T_H^i/x_1^i \\ x_1^i/T_L^i - T_H^i/x_1^i \\ \vdots \\ x_1^i/T_L^i - T_H^i/x_1^i \end{bmatrix} \qquad (2)$$

A final score value is calculated for each p_i as shown in Eq. (3). A contiguity check (Eq. (4)) is used to determine if there are at least c contiguous elements in m_B^i or m_D^i that are true. In [6]-[8], c = 9 (hence the algorithm is called FAST-9). Finally, non-maximum suppression is applied to determine whether a pixel is a corner or a non-corner. A pixel is a corner if it has a maximal score among the scores of its adjacent neighbors.

$$score_i = max\left(score_B^i, score_D^i\right) \cdot C_i \qquad (3)$$

$$C_i = \left(\vee_{j=1}^{16}\left(\wedge_{k=j}^{j+8} m_B^i (k-1)_{mod\ 16+1}\right)\right) \vee \left(\vee_{j=1}^{16}\left(\wedge_{k=j}^{j+8} m_D^i (k-1)_{mod\ 16+1}\right)\right) \qquad (4)$$

Fig. 2a shows the baseline architecture. We assume a single input pixel of n-bit (in our implementation n = 8 for grayscale image) arrives at each clock cycle. Similar to the implementations in [6]-[8], 7 row buffers are concatenated in the form of FIFO delay buffers to cache the incoming pixels. The size of each row buffer is equivalent to the horizontal resolution of the image, and hence each row buffer effectively delays the input by one row [15]. The pixels at the tail end of each row buffer are shifted into the 7x7 window buffer in Fig. 2a.

The Bright Score Unit (BSU) and Dark Score Unit (DSU) determines the 16-bit member vectors m_B^i, m_D^i and $score_B^i, score_D^i$ in parallel. The architecture of BSU and DSU is shown in Fig. 2b. A 2-stage pipelined adder tree is employed for computing the score values to enable meaningful area-time evaluation with the proposed architecture. The LSB of the score values are truncated to n bits. m_B^i, m_D^i are used by the contiguity check to compute C_i. Registers are included to ensure that the outputs score values and member vectors are synchronized. The Max unit computes the score of p_i based on the bright score, dark score and contiguity check.

Finally, the Non-Maximal Suppression (NMS) unit determines if a pixel is a corner or not by comparing its score value to the score values of its 8 adjacent pixels. To achieve this, 2 row buffers are used in the NMS unit to produce a 1-bit output that denotes whether the corresponding pixel is a corner or non-corner. Note that all the outputs of each module in Fig. 2a are registered, creating a pipelined design with one input and one output per clock cycle without the need of an input or intermediate frame buffer.

The critical path of the baseline architecture lies in BSU/DSU, i.e.: $T_{cp}^b = 3 \cdot T_{ADD} + T_{MUX}$, where T_{ADD} and T_{MUX} is the time required by the adder and multiplexer respectively.

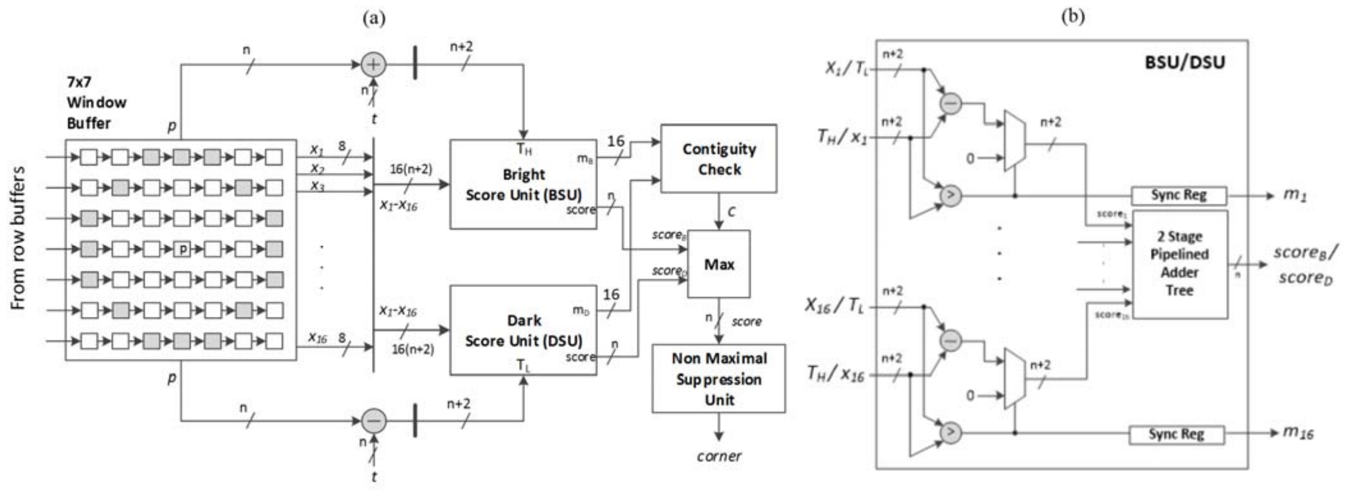

Fig. 2: (a) Baseline architecture, (b) BSU/DSU.

IV. Proposed FAST-DT Architecture

The baseline architecture in Fig. 2a utilizes a 7x7 window buffer to enable the 16 pixels x_j^i in the Bresenham circle to be evaluated in parallel with pixel p_i. Only one pixel p_i is tested for a corner at each time step. This approach results in under-utilization of the other 33 pixels (7x7 – 16) in the window buffer which could be used for partial evaluations of multiple p_i's. This is shown with the help of the red dotted lines in Fig. 1a, where at time *t*, all the pixels in column C8 of the window can be used for partial examination of pixels p_1, p_2, and p_3.

Our proposed architecture, called FAST-DT (DT stands for Data-path Transposition), is shown in Fig. 3. The term 'transposition' refers to the transposition of the vertical BSU/DSU data-path structures in the baseline architecture (see Fig. 2b) to the horizontal BSU/DSU data-path structures in Fig. 3. In our proposed architecture for the FAST algorithm, the 7x7 window buffer is replaced by 7x3 window buffer. 3 columns of the window are required to cache the incoming pixels before the center pixel is read from the FIFO delay buffers. For example, it can be observed in Fig. 1a that we need to cache the pixels in columns C6, C7 and C8 before we can obtain the center pixel p_1 from the FIFO delay buffer.

The BSU/DSU data-path is transposed to partially examine multiple pixels concurrently using all the pixels in the last column of the window buffer. It can be observed that the proposed architecture in Fig. 3 only requires a 7x3 window buffer, and BSU/DSU of the baseline architecture is unrolled into 7 pipeline stages. Each stage performs partial evaluation of a single p_i. The partial results of each p_i (partial member vectors, bright/dark scores) will be passed to the next pipeline stage after each clock cycle. The full evaluation of a single p_i will be completed after 7 clock cycles at the final pipeline stage.

We can use the example in Fig. 1 to describe the data flow. Let's assume that at time *t*, the content of the last window buffer column is C8. At time *t*, the partial results of p_1 is computed using x_4^1, x_5^1, x_6^1 in the first pipeline stage of BSU/DSU. In the next clock cycle at time *t*+1, the partial results p_1 is computed using x_3^1, x_7^1 in the second pipeline stage and the new score values are added to the previously computed score values from the first stage. At the same time, new member vectors are generated and concatenated with the previously identified member vectors. This is repeated until the last pipeline stage at *t*+6 that computes the partial results of p_1 using $x_{12}^1, x_{13}^1, x_{14}^1$ and concatenates/adds the partial results in the previous pipeline stage to obtain m_B^1/m_D^1 and $score_B^1/score_D^1$. Like the baseline architecture, a single output is produced at each clock cycle.

Fig. 4 shows the architecture of the transposed BSU/DSU. Each stage consists of either two or three score units, where the score outputs are added using a 1-stage adder tree. Except for the first pipeline stage, the score values of each stage are added to the score value from the previous pipeline stage. As mentioned earlier, the last pipeline stage produces the final score value of p_i. Although not shown in the figure, the values of $\mathrm{T_H}/\mathrm{T_L}$ are also shifted through the BSU/DSU stages in a pipelined manner, i.e. the score unit at a stage makes use of the shifted $\mathrm{T_H}/\mathrm{T_L}$ values from the previous stage.

The architecture of the BSU and DSU can be further simplified by using two's complement adders to compute $score_B^i, score_D^i$, and using the sign bits of the scores to determine the member vectors as shown in Fig. 4. This effectively removes 16 comparators in each of the BSU and DSU without introducing much additional critical path delay. We denote the proposed design with 1-stage adder tree for computing the score as FAST-DT1, and the critical path of FAST-DT1 is $T_{cp}^{dt1} = 3 \cdot T_{ADD} + T_{INV} + T_{MUX}$, where T_{INV} is the delay of an inverter.

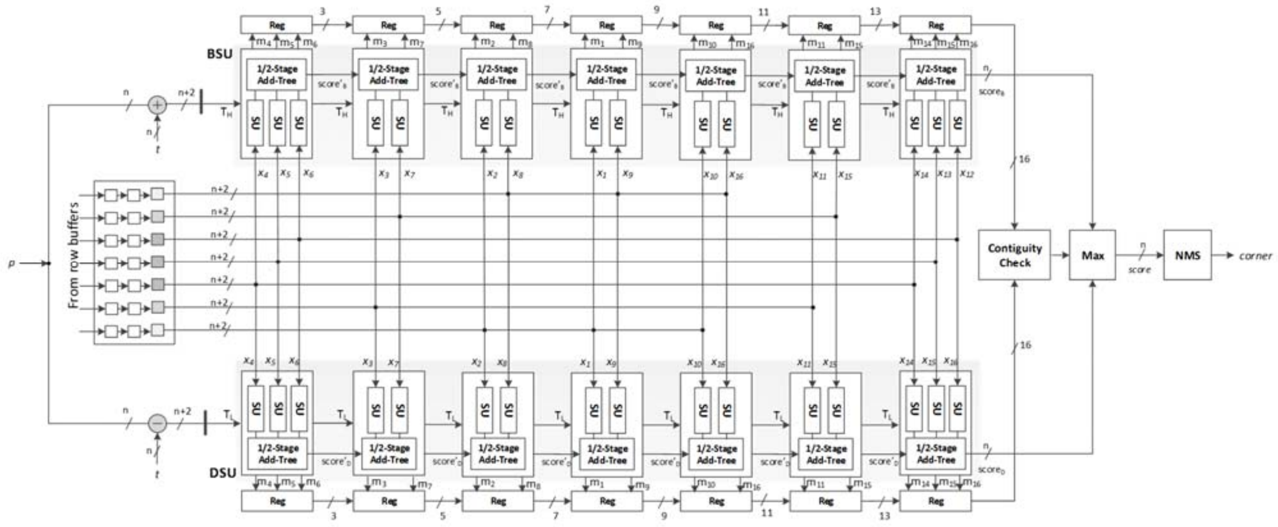

Fig. 3: Proposed FAST-DT architecture.

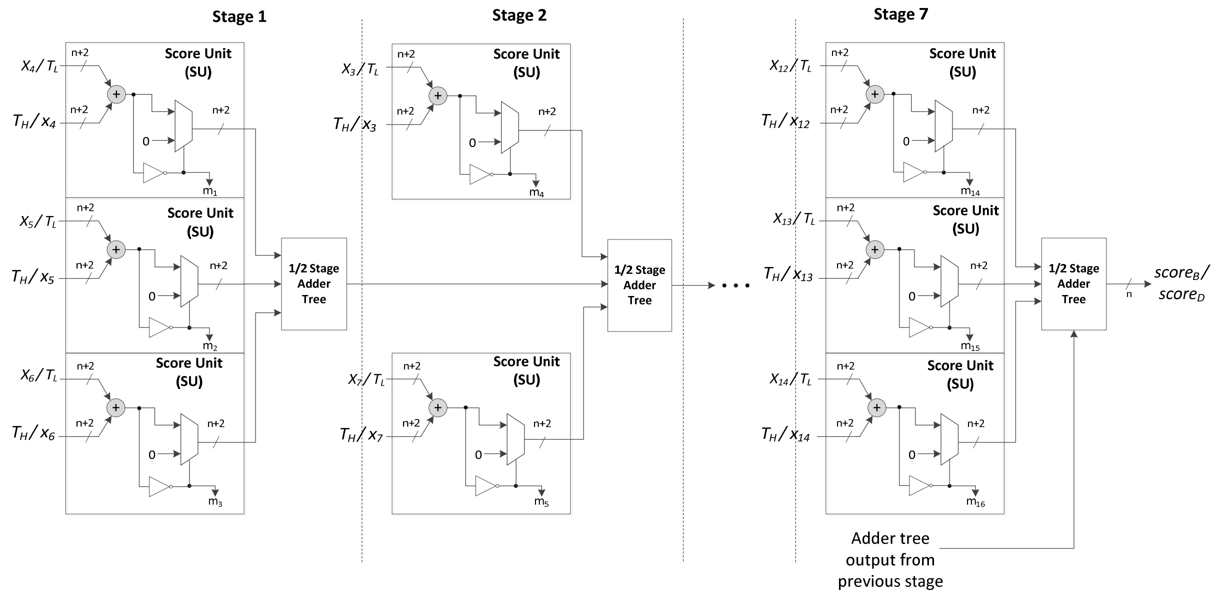

Fig. 4: BSU/DSU pipeline stages for FAST-DT.

FAST-DT1 requires lesser area than the baseline architecture due to the smaller window buffer and the elimination of comparators for determining the member vectors. The area reduction is achieved at a marginal increase in critical path delay (i.e. T_{INV}). Hence, FAST-DT1 and the baseline architecture is expected to have similar throughput. To improve the throughput of FAST-DT1, we include an additional pipeline stage in the adder trees (2-stage adder tree) for computing the score. Additional registers are also included to maintain data synchronization. This design that uses a 2-stage adder tree is denoted as FAST-DT2, which reduces the critical path of FAST-DT1 to $T_{cp}^{dt2} = 2 \cdot T_{ADD} + T_{INV} + T_{MUX}$. The synthesis results in the next section shows that the reduced critical path delay is achieved while still maintaining lesser area utilization than the baseline architecture.

V. Results and Discussion

Table 1 shows the resource analysis for the baseline (Fig. 2), and proposed FAST-DT1 and FAST-DT2 architectures (Fig. 3). The resources for row buffers, Contiguity Check, Max and NMS are not shown since they do not vary among the three architectures. ADD-T refers to the adders for the addition tree that is used to compute the score values. The pipeline registers at the adder trees and synchronization registers are omitted from the resource analysis. It can be observed from Table 1 that FAST-DT1 requires lesser number of registers for the window buffer than the baseline. In addition, FAST-DT1 has a significant reduction in combinational area due to elimination of the comparators. FAST-DT2 has the same combinational area as FAST-DT1, but requires more pipeline registers due to the additional pipeline stages.

TABLE 2: SYNTHESIS RESULTS OF FAST CORNER DETECTOR BASED ON 180-NM CMOS TECHNOLOGY LIBRARY

Design	Area (sq um)	Minimum Clock Period (ns)	Power (mW)	Energy per pixel	Area-Time Product	Frames per second for HD images (1280x720)
Baseline	914534	1.93	416.29	803.44	1765050.29	562
FAST-DT1	905020	1.88	400.76	753.43	1701438.18	577
FAST-DT2	908353	1.50	543.98	815.97	1362530.07	723

TABLE 1: RESOURCE COMPARISON

		Baseline	FAST-DT1	FAST-DT2
Window Buffer	**REG**	49	21	21
BSU/DSU	**ADD**	16	16	16
	ADD-T	15	16	16
	MUX	16	16	16
	Comb	16 COMP	16 INV	16 INV

The baseline and proposed architectures were implemented using Verilog and synthesized with Synopsys DC targeting the 180-nm CMOS technology library. The designs were synthesized to achieve maximum clock frequency. To perform accurate power analysis, we use Synopsys VCS to obtain the actual switching activity statistics of the architectures based on a common input image. Energy per pixel is computed as the product of power (mW) and minimum clock period (ns). As shown in Table 2, the proposed FAST-DT1 architecture has the lowest area utilization and power consumption, which is expected from our resource analysis. The critical path delay of FAST-DT1 is slightly lower than the baseline architecture. Although we expected the converse due to the additional T_{INV} in FAST-DT1 as discussed in the previous section, the delay difference between FAST-DT1 and baseline is negligible. With the introduction of additional pipeline stages, FAST-DT2 achieves 22.2% reduction in the critical path delay compared to the baseline architecture while still maintaining lesser area utilization. While the critical path delay of the baseline architecture can be reduced with additional pipeline stages in the adder tree, this will further increase its area utilization, which at present, is already higher than FAST-DT1 and FAST-DT2. The area-product of the FAST-DT2 architecture is 22.8% lesser than the baseline architecture. The proposed FAST-DT2 architecture has a higher clock operating frequency and hence it consumes more power. However, the energy per pixel is only about 1.6% higher than the baseline. Hence, the proposed FAST-DT2 architecture offers the highest area-time benefits without compromising on energy efficiency compared to existing approaches. The last column of Table 2 shows that the baseline and FAST-DT1 have similar throughput (where FAST-DT1 requires lesser area and power), while FAST-DT2 is capable of processing images with resolution of 1280x720 at over 700 frames per second.

VI. CONCLUSION

Area-time analysis and synthesis results of FAST-DT1 confirm that the proposed transposed data-path structure for FAST corner detector leads to lesser registers with the use of a smaller window buffer for computing partial corner score values in parallel. The use of two's complement adders to simultaneously compute the corner scores and members leads to further area savings. These area optimizations are achieved without notable difference in critical path delay compared to the baseline architecture. By incorporating an additional pipeline stage in the adder trees, the proposed FAST-DT2 architecture resulted in over 22% area-time reduction compared to the baseline architecture, with similar energy efficiency.

REFERENCES

[1] A. Schmidt, M., Kraft, and A. Kasinski, "An evaluation of image feature detectors and descriptors for robot navigation", *Computer Vision and Graphics*, Vol. 6375, pp. 251-259, 2010

[2] 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

[3] A. Gil, O. Mozos, M. Ballesta, and O. Reinoso, "A comparative evaluation of interest point detectors and local descriptors for visual SLAM", *Machine Vision and Applications*, Vol. 21, pp. 905-920, 2010

[4] J. Zhou, J. Yan, T. Wei, K. Wu, X, Chen and S. Hu, "Sharp corner/edge recognition in domestic environments using RGB-D camera systems", *IEEE Transactions on Circuits and System II: Express Briefs*, Vol. 62, No. 10, pp. 987-99, 2015

[5] L. Puglia, M. Vigliar, and G. Raiconi, "Real-time low-power FPGA architecture for stereo vision", *IEEE Transactions on Circuits and Systems II: Express Briefs*, April 2017

[6] M. Kraft, A. Schmidt and A. Kasinski, "High-speed image feature detection using FPGA implementation of FAST algorithm", *Proceedings of the Third International Conference on Computer Vision Theory and Applications*, 2008

[7] D. Soberl, N. Zimic, A. Leonardis, J. Krivic and Miha Moskon, "Hardware implementation of FAST algorithm for mobile applications", *Journal of Signal Processing Systems*, Vol. 79, No. 3, pp. 247-256, 2015

[8] 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

[9] K. Dohi, Y. Yorita, Y. Shibata, and K. Oguri, "Pattern compression of fast corner detection for efficient hardware implementation", *International Conference on Field Programmable Logic and Applications*, pp. 478–481, 2011

[10] E. Rosten and T. Drummond, "Fusing points and lines for high performance tracking", *International Conference on Computer Vision*, pp. 1508–1515, 2005

[11] E. Rosten, R. Porter and T. Drummond, "Faster and better: A machine learning approach to corner detection", *IEEE Transactions Pattern Analysis and Machine Intelligence*, 32, 105–119, 2010

[12] 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

[13] J.S. Park, H.E. Kim, H.Y. Kim, J. Lee and L.S. Kim, "A vision processor with a unified interest-point detection and matching hardware for accelerating a stereo-matching algorithm", *IEEE Trans. Circuits and Systems for Video Technology*, Vol. 26, No. 12, pp. 2328-2343, 2016

[14] E. Rublee, V. Rabaud, K. Konolige and G. Bradski, "ORB: an efficient alternative to SIFT or SURF", *IEEE International Conference on Computer Vision*, 2011

[15] S.L. Chen, "VLSI implementation of a low-cost high-quality image scaling processor", IEEE Transactions on Circuits and Systems II: Express Briefs, Vol. 60, No. 1, 2013, pp. 31-35