Journal of Data Acquisition and Processing
ISSN: 1004-9037
https://sjcjycl.cn/
DOI: 10.5281/zenodo.777205

HIGH-THROUGHPUT AND DYNAMIC POWER AWARE GLOBAL OPTICAL FLOW METHOD USING CLOCK GATING AND MRP

[1]T. Lakshmi Bhargavi, [2]G. Suneetha, [3] k.Babu Rao, [4] Dr.B.Nancharaiah.
[1]M. Tech, Dept. of E.C.E, Usha Rama College of Engineering and Technology, AP, INDIA
[2] Assistant Professor Dept. of E.C.E, Usha Rama College of Engineering and Technology, AP, INDIA
[3] Co-ordinator Dept. of E.C.E, Usha Rama College of Engineering and Technology, AP, INDIA
[4] Professor & H.O.D.Dept. of E.C.E, Usha Rama College of Engineering and Technology, AP, INDIA

ABSTRACT:
Optical Flow (OF) approaches for motion estimation calculate vector fields for the apparent velocities of objects in image sequences. Most very-large-scale integration (VLSI) designs of the global optical flow method focus on reducing external memory accesses due to global and iterative processes for low power operation in mobile systems. However, to achieve this goal, considerable amounts of resources are used and the throughput is decreased. To address these issues, we propose a multi row based propagation approach and an efficient VLSI architecture. This approach divides an image into multiple small sub images. The flow of each sub image is then estimated sequentially using a small amount of internal memory without accessing any external memory. To avoid discontinuity artifacts stemming from the boundaries of the sub images, boundary conditions are imposed based on the smoothness of the optical flow. In addition, the main equations of the global optical flow method are simplified to boost the processing speed. Further this project is enhanced by using clock gating based memory in order to reduce both power and throughput.
Keywords: Clock Gating, artifacts, Optical flow, multi row based propagation, Memory organization

INTRODUCTION: Optical flow estimation is one of the oldest reconstruction problems. This problem which consists of computing the 2D apparent motion field between two consecutive frames of an image sequence, is indeed an inescapable prerequisite in a wide range of applications such as passive navigation, stereovision, image sequence restoration, video compression, etc. Several techniques have been proposed to estimate velocity fields. Among others, one can distinguish spatiotemporal filtering methods,13 tensor-based techniques,20 correlation-based techniques1 or minimizing energy-based techniques.18 The latter, despite leading to intensive calculations, yields the most accurate flow fields in terms of precision21 or in terms of spatial discontinuity estimation.5,23 Nevertheless, the ability of such methods to be embedded in real time applications is limited by the amount of computing power needing to solve them. The computation cost is so high that nowadays it is unimaginable to reach that goal on conventional computers.

LITERATURE SURVEY:

A fundamental component in the construction of a machine's vision system is the computation of optical flow which is obtained by estimating a dense motion field corresponding to the displacement of each pixel in consecutive frames of an image sequence. Its reliable calculation comprises one of the main challenges in computer vision. Optical flow can be combined with various computer vision tasks such as video coding, segmentation, tracking [1] and multi view-reconstruction [2]. Some other fields where optical flow has played an important role includes fluid mechanics [3], solar physics [4], autonomous driving [5], biomedical images [6], breast tumors [7], bladder cancer [8] surveillance and traffic monitoring [9], virtual reality [10], face recognition and tracking [11], and action recognition videos [12]. Optical flow is the pattern of the apparent motion of objects in a visual scene caused by the motion of an object or camera or both. When a camera records a scene for a given time, the resulting image sequence can be considered as a function of gray values at image pixel position (x,y) and the time t. If the camera or an object moves within the scene, this motion results in a time-dependent displacement of the gray values in the image sequence. The resulting two-dimensional apparent motion field in the image domain is the Optical Flow Field. At present, optical flow estimation stands at its peak with a steady progress. In last 4 decades, a whole class of various techniques and novel concepts has evolved in this area. Particularly, remarkable development has been witnessed in the last decade. On one hand, the advance level datasets such as Middlebury [13], MPI-Sintel [14]and KITTI [15, 16] presented substantial novel challenges for the optical flow algorithms, on the other hand traditional methods such as LDOF [17], DeepFlow [18], EpicFlow [19], DiscreteFlow [20], FlowFields [21] and MirrorFlow [22] came up with a significant number of novel strategies. These innovations solved the correspondence problem with outstanding performance. However, the enhanced accuracy subsequently increased the evaluation time. Consequently, none of the major traditional methods runs in real time currently.

EXISTING METHOD:

MULTIROW-BASED PROPAGATION (MRP):

The global optical flow method can extract dense and accurate flows owing to the aforementioned global and iterative processes. However, it suffers severely from excessive access to external memory caused by the following processes. Before entering the loop in Algorithm 1, the structure tensors, J nm i, j , are computed and stored in external memory for global execution. During the execution of the loop, J nm i, j , uk i, j , vk i, j , and their neighbors are read for the current iteration, and uk+1 i, j and vk+1 i, j are buffered for the next iteration. These processes continue until the end of the loop, whereas numerous accesses to external memory occur. To reduce this external memory access while using minimal internal memory, we propose the MRP approach. The proposed scheme executes partially in the units of the multirow (r), composed of multiple rows, instead of the entire image, as shown in Algorithm 2. With a small multirow size, the structure tensors (J nm i, j) of derivation can be stored using a small amount of internal memory. Then, derivation is also computed iteratively using only the internal memory without accessing external memory. The next multirow starts after finishing the iteration for the current multirow, and this process is applied sequentially until the last multirow to obtain the optical flow of the entire image. Although this multirow-based approach can significantly reduce external memory access, discontinuity artifacts are caused because of a lack of neighboring flows, i.e., (u,v)k+1 r,i−1, j or (u,v)k r,i+1, j , when computing at the upper or lower boundary of the multirow. To alleviate these artifacts, which adopt flows estimated at the previous multirow as the neighboring flows. This is done because the global optical flow method generally assumes smoothness of the flow over

Algorithm 2 Proposed Approach

```
1  for $r = 0, ..., R-1$ do
2      for $i = (r \times H_R) + 0, ..., (r \times H_R) + (H_R - 1)$ do
3          for $j = 0, ..., W-1$ do
4              $Computation\ of\ Eq.\ (7)$
5          end
6      end
7      for $k = 0, ..., K-1$ do
8          for $i = (r \times H_R) + 0, ..., (r \times H_R) + (H_R - 1)$ do
9              for $j = 0, ..., W-1$ do
10                 $Computation\ of\ Eq.\ (8)\ using\ Eq.\ (6)$
11             end
12         end
13     end
14 end
```

Notation: r is the multirow index. R denotes the total number of multirows; i.e., $R = \frac{H}{H_R}$. Here, H_R and H denote the height of the multirow and the height of the image, respectively.

the entire image

$$(u,v)^{k+1}_{r,i-1,j} = (u,v)^{K-1}_{r-1,i-1,j} \quad \text{if } i = (r \times H_R) + 0$$

$$(u,v)^{k}_{r,i+1,j} = (u,v)^{k}_{r,i,j} \quad \text{if } i = (r \times H_R) + (H_R - 1)$$

Here, r is the multirow index. k and K denote the index and the number of iterations, respectively, and i and j likewise denote the vertical and horizontal locations in an image. HR is the multirow height. We use the estimated flows at the bottom row of the (r−1)th multirow as the upper neighboring flows, i.e., Nu and Nv , at the top row of the (r)th multirow, as the first condition of (1). Furthermore, the currently estimated flows at the bottom row of the (r)th multirow during the iterations are used directly as the lower boundary of the (r)th multirow using the second condition. With these conditions, the estimated flows of the previous multirow are propagated indirectly through the subsequent multirows. Consequently, the artifacts derived from the boundaries are substantially mitigated.

As the outputs ((u,v)k+1 i, j) are directly used as inputs ((u,v)k+1 i, j−1) for the next pixel, a waiting delay occurs, and the complex operations lengthen the delay. Furthermore, because the hardware unit used to compute (5) is duplicated and pipelined to accelerate the iterations [30]–[32], the logic is increased proportionally with the complexity of (5). Simplification of (5) is critical to boost the processing speed and reduce the use of logic. We extract some coefficients from the loop body and move them outside of the loop by means of precomputation to simplify

(5), as shown in (7) and the first loop body of Algorithm 2. Because ω, α, |Nu |, and |Nv | are constants, they are calculated using J nm i, j in advance and stored in internal

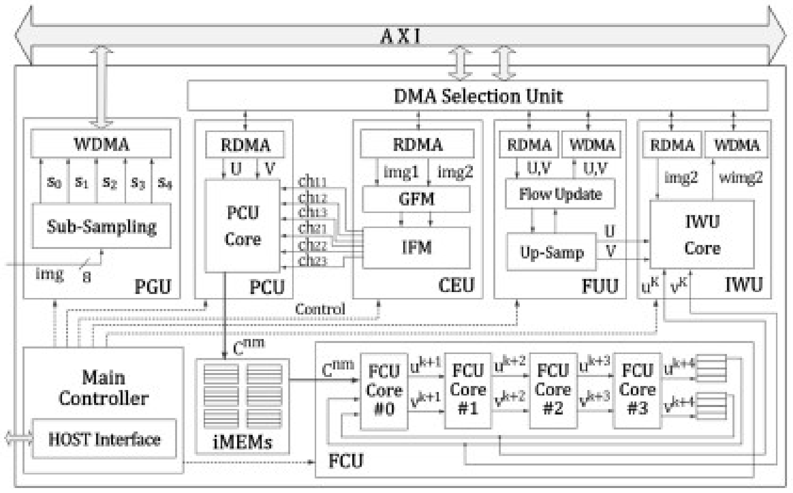


Fig 1: Overall block diagram of the proposed VLSI architecture. Memory

$$C^{nm}_{r,i,j} = \begin{bmatrix} \dfrac{\omega}{|N_u| + \dfrac{1}{\alpha}J^{11}_{r,i,j}} & \dfrac{-C^{11}_{r,i,j}J^{12}_{r,i,j}}{\alpha} & \dfrac{-C^{11}_{r,i,j}J^{13}_{r,i,j}}{\alpha} \\ \dfrac{\omega}{|N_v| + \dfrac{1}{\alpha}J^{22}_{r,i,j}} & \dfrac{-C^{21}_{r,i,j}J^{21}_{r,i,j}}{\alpha} & \dfrac{-C^{21}_{r,i,j}J^{23}_{r,i,j}}{\alpha} \end{bmatrix}.$$

In this equation, Cnm denotes the component (n, m) of the above matrix. For the definitions of the other notations, readers can refer to (5). Consequently, (5) is simplified, as shown in (8) and the second loop body of Algorithm 2. With this simplification, five divisions of the loop body related to k are removed entirely. Furthermore, the six multiplications and eight additions are reduced to three and six, respectively

$$u^{k+1}_{r,i,j} = (1-\omega)u^{k}_{r,i,j} + C^{11}_{r,i,j}\left(\sum N_u\right) + C^{12}_{r,i,j}\left(v^{k}_{r,i,j}\right) + C^{13}_{r,i,j}$$
$$v^{k+1}_{r,i,j} = (1-\omega)v^{k}_{r,i,j} + C^{21}_{r,i,j}\left(\sum N_v\right) + C^{22}_{r,i,j}\left(u^{k+1}_{r,i,j}\right) + C^{23}_{r,i,j}.$$

We propose a VLSI design based on our MRP approach. The proposed design shown in Fig. 2 consists of six units, a main controller, and internal multirow memories (iMEMs). The channel extension unit (CEU) is designed for converting a single-channel gray image into three channels. Using the three outputs from the CEU, flows are computed recursively and optimized by the precomputation unit (PCU) and flow calculation unit (FCU), which implement the first and second loop bodies of Algorithm 2, respectively. To support the coarse-tofine warping approach, the image warping unit (IWU) warps an original image using computed flows, and these flows are enlarged by the flow upsampling unit (FUU) for a finer scale. In addition, the pyramid generation unit (PGU) generates an input image pyramid with five scales in parallel and writes them to external memory independent of the flow computation. The main controller handles the operational sequence of all units and has a host interface module to receive parameters

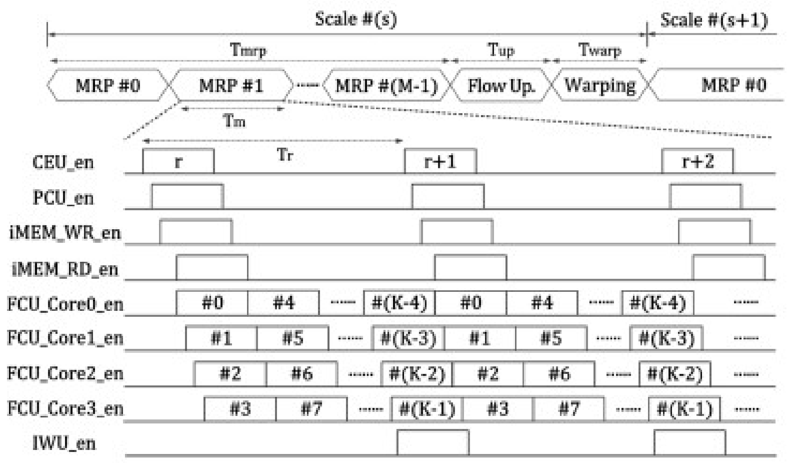


Fig 2: Main timing diagram to control the operation.

such as ω and α from the host CPU. Furthermore, write direct memory access (WDMA) and read direct memory access (RDMA) modules are placed to access the external memory. To share one advanced extensible interface (AXI) channel, the direct memory access (DMA) selection unit controls the DMA modules using a round-robin strategy. We describe the operation of the aforementioned units in detail as follows.

PROPOSED EXTENSION:

MEMORY DESIGN USING CLOCK GATING:

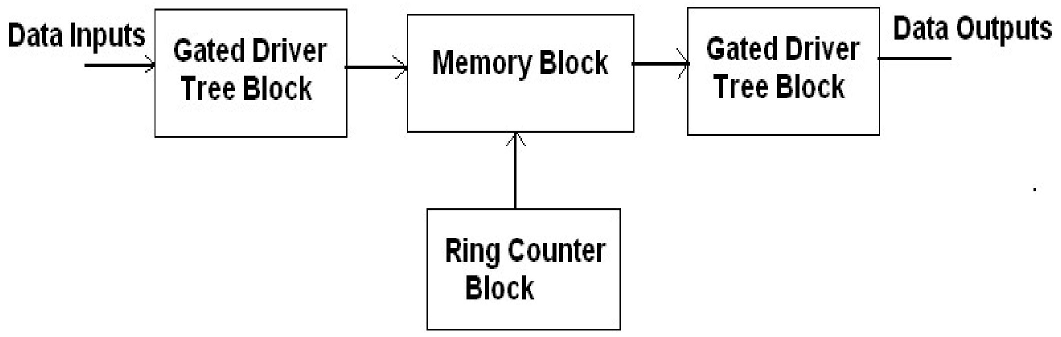


Fig 3: Block Diagram For Proposed Delay Buffer

GATED DRIVER THREE:

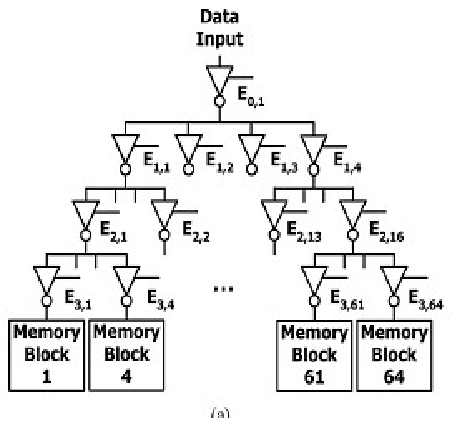


Fig 4: Gated Driver Tree

Gated driver tree derived from the same clock gating signals of the blocks that they drive. Thus, in a quad-tree clock distribution network, the “gate” signal of the gate driver at the level (CKE) should be asserted when the active DET flip-flop

MODIFIED RING COUNTER:

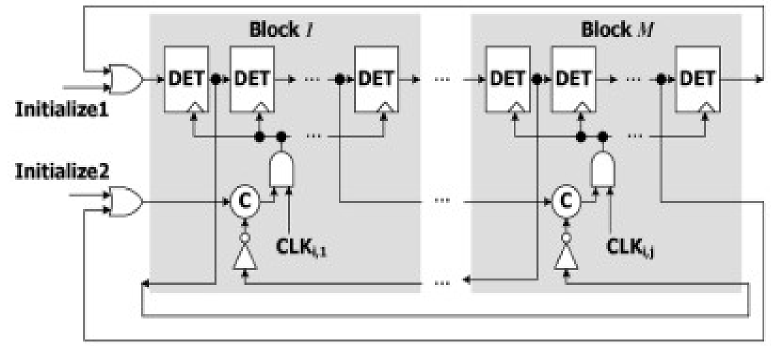


Fig 5: Modified Ring Counter

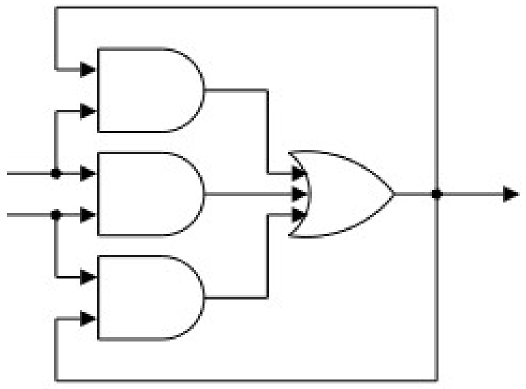
DET (Double edge triggered flip-flops:

Double-edge-triggered (DET) flip-flops are utilized to reduce the operating frequency by half The logic construction of a double-edge-triggered (DET) flip-flop, which can receive input signal at two levels the clock, is analyzed and a new circuit design of CMOS DET . by transmission gates

C ELEMENT:

The Muller **C-element**, or Muller C-gate, is a commonly used asynchronous logic component originally designed by David E. Muller. It applies logical operations on the inputs and has hysteresis.

Here is the truth table for a 2-input c-gate. Y_{n-1} denotes a "no change" condition.

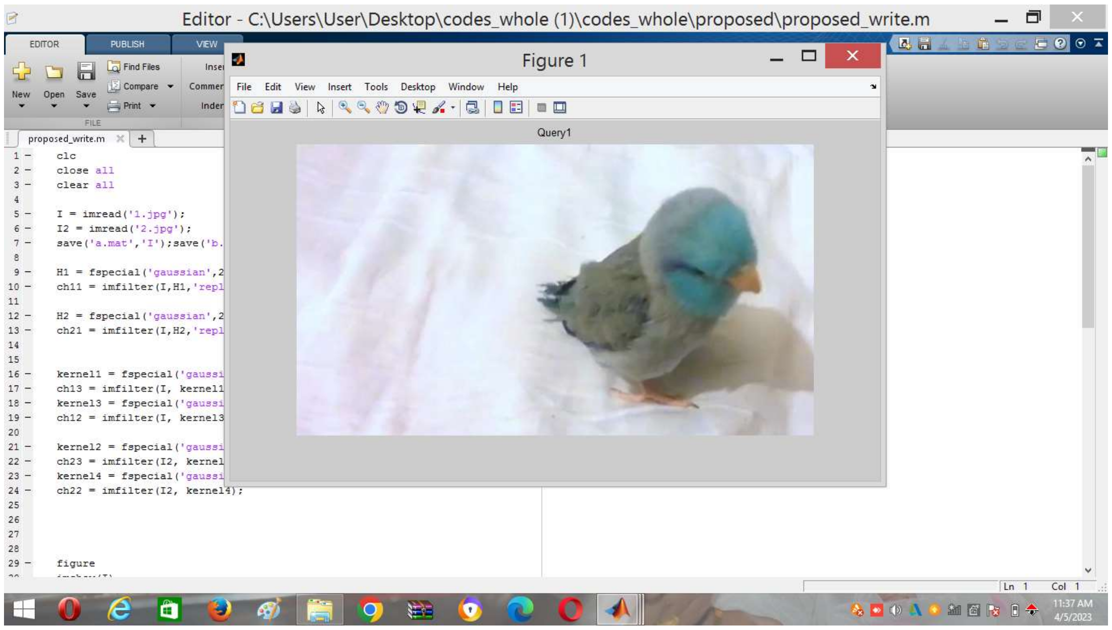


A	B	Q
0	0	0
0	1	Q(t-1)
1	0	Q(t-1)
1	1	1

Fig 6: C- Element **Table 1:** Truth Table For C-Element

The C-element stores its previous state with two cross-coupled inverters, similar to an SRAM cell. One of the inverters is weaker than the rest of the circuit, so it can be overpowered by the pull-up and pull-down networks.

RESULTS:


Editor - C:\Users\User\Desktop\codes_whole (1)\codes_whole\proposed\proposed_write.m
proposed_write.m
Figure 1
Query1

```
clc
close all
clear all

I = imread('1.jpg');
I2 = imread('2.jpg');
save('a.mat','I');save('b.

H1 = fspecial('gaussian',2
ch11 = imfilter(I,H1,'repl

H2 = fspecial('gaussian',2
ch21 = imfilter(I,H2,'repl


kernel1 = fspecial('gaussi
ch13 = imfilter(I, kernel1
kernel3 = fspecial('gaussi
ch12 = imfilter(I, kernel3

kernel2 = fspecial('gaussi
ch23 = imfilter(I2, kernel
kernel4 = fspecial('gaussi
ch22 = imfilter(I2, kernel4);




figure
```

FIig : Proposed Write input image

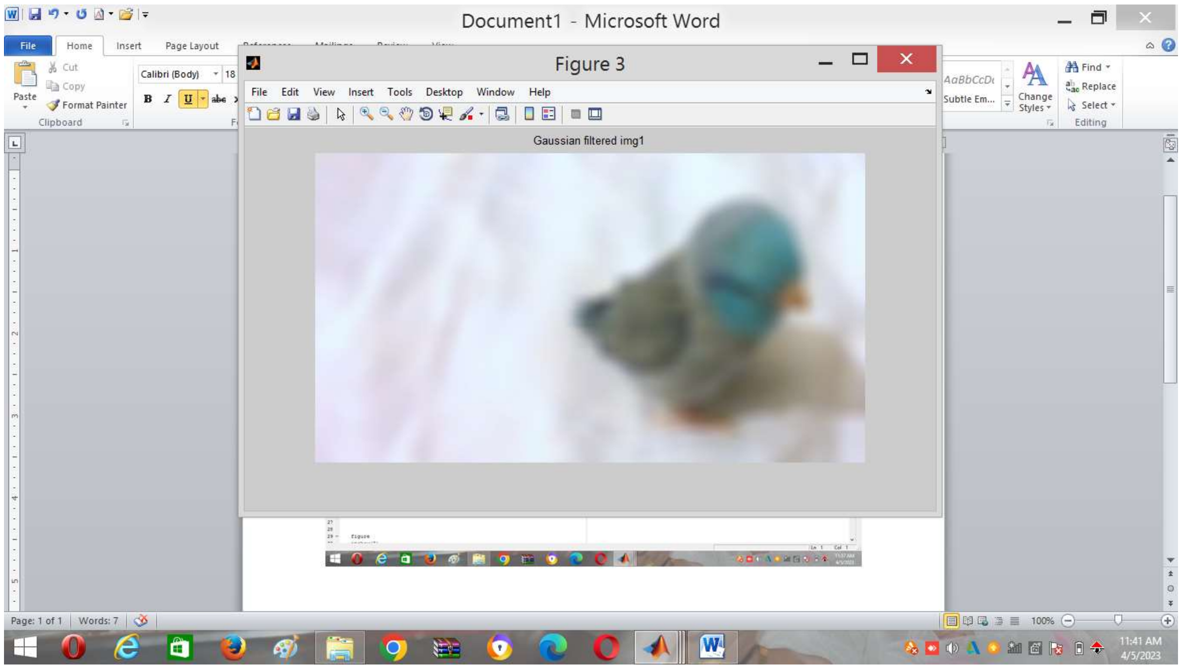


Fig: proposed Gaussian filtering image

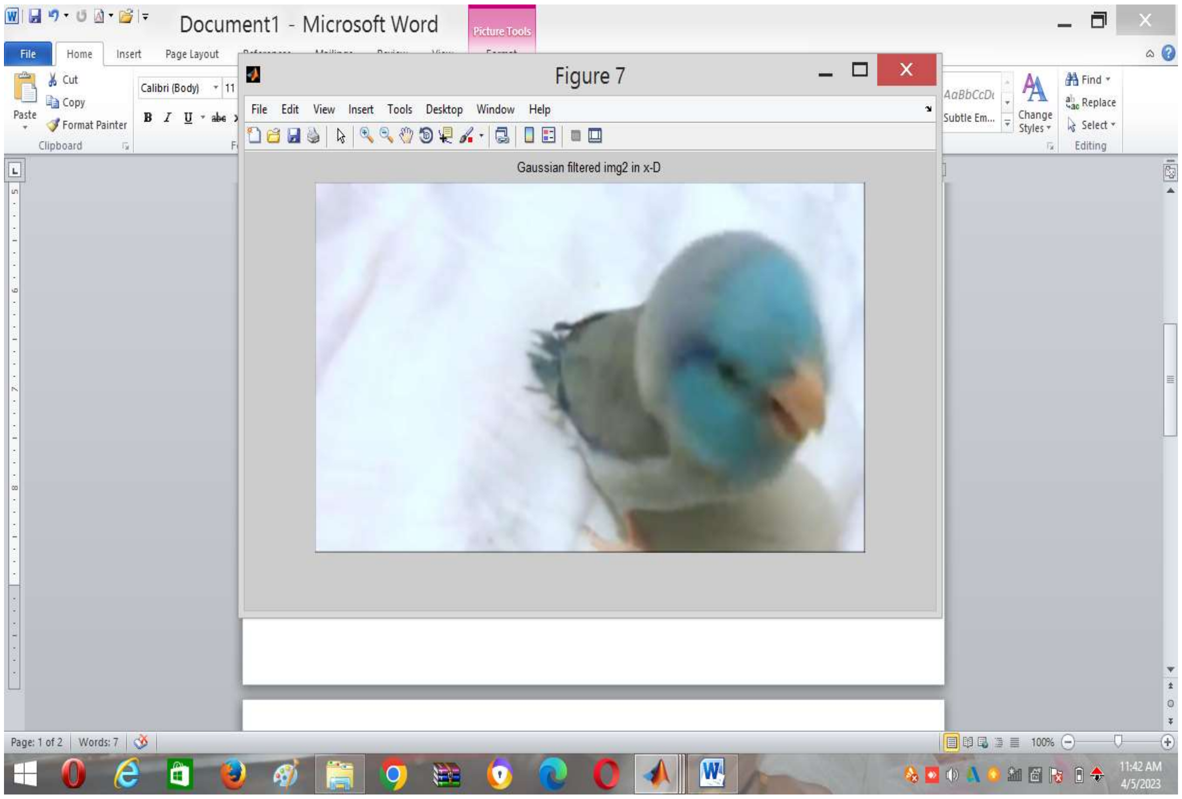
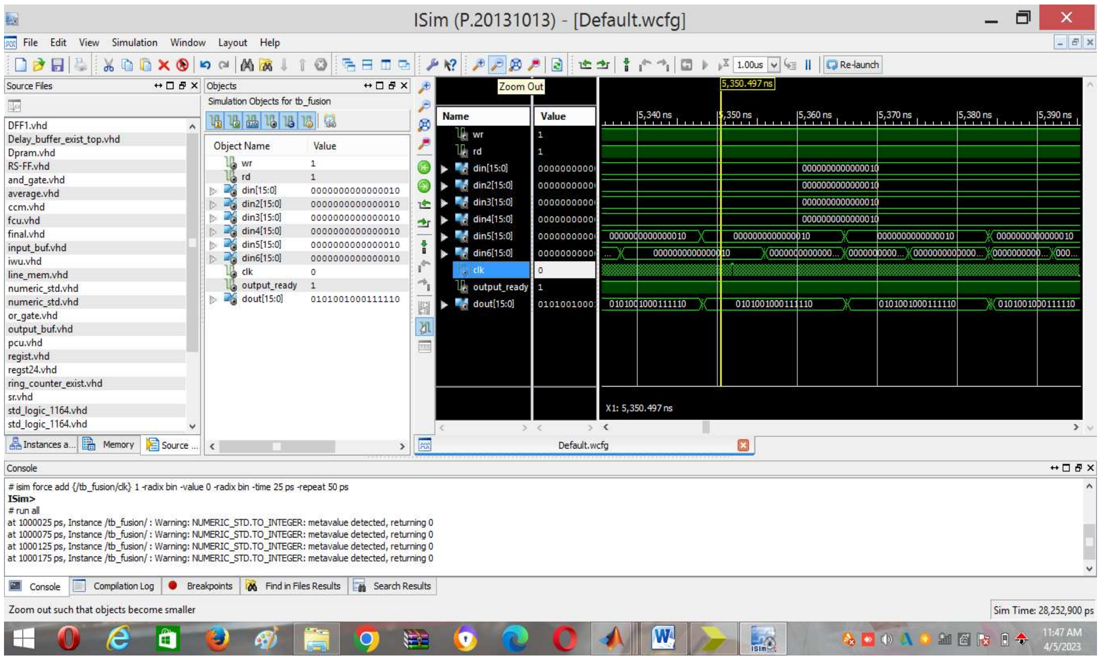

Document1 - Microsoft Word
Figure 7
File Edit View Insert Tools Desktop Window Help
Gaussian filtered img2 in x-D

Fig : proposed gaussian filtering img XD



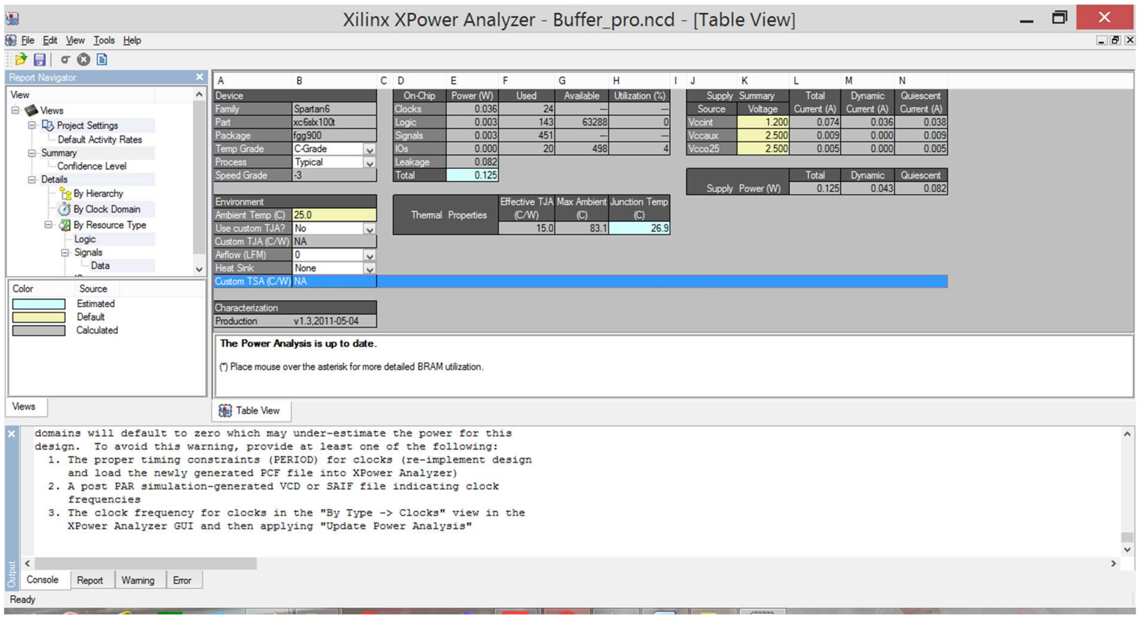
```
DFF1.vhd
Delay_buffer_exist_top.vhd
Dpram.vhd
RS-FF.vhd
and_gate.vhd
average.vhd
ccm.vhd
fcu.vhd
final.vhd
input_buf.vhd
iwu.vhd
line_mem.vhd
numeric_std.vhd
numeric_std.vhd
or_gate.vhd
output_buf.vhd
pcu.vhd
regist.vhd
regst24.vhd
ring_counter_exist.vhd
sr.vhd
std_logic_1164.vhd
std_logic_1164.vhd
```

Object Name	Value
wr	1
rd	1
din[15:0]	0000000000000010
din2[15:0]	0000000000000010
din3[15:0]	0000000000000010
din4[15:0]	0000000000000010
din5[15:0]	0000000000000010
din6[15:0]	0000000000000010
clk	0
output_ready	1
dout[15:0]	0101001000111110

```
# isim force add {/tb_fusion/clk} 1 -radix bin -value 0 -radix bin -time 25 ps -repeat 50 ps
ISim>
# run all
at 1000025 ps, Instance /tb_fusion/ : Warning: NUMERIC_STD.TO_INTEGER: metavalue detected, returning 0
at 1000075 ps, Instance /tb_fusion/ : Warning: NUMERIC_STD.TO_INTEGER: metavalue detected, returning 0
at 1000125 ps, Instance /tb_fusion/ : Warning: NUMERIC_STD.TO_INTEGER: metavalue detected, returning 0
at 1000175 ps, Instance /tb_fusion/ : Warning: NUMERIC_STD.TO_INTEGER: metavalue detected, returning 0
```

Zoom out such that objects become smaller

Fig 7: proposed simulation result


Xilinx XPower Analyzer - Buffer_pro.ncd - [Table View]

Device	
Family	Spartan6
Part	xc6slx100t
Package	fgg900
Temp Grade	C-Grade
Process	Typical
Speed Grade	-3

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.036	24	---	---
Logic	0.003	143	63288	0
Signals	0.003	451	---	---
IOs	0.000	20	498	4
Leakage	0.082			
Total	0.125			

Supply Summary		Total	Dynamic	Quiescent
Source	Voltage	Current (A)	Current (A)	Current (A)
Vccint	1.200	0.074	0.036	0.038
Vccaux	2.500	0.009	0.000	0.009
Vcco25	2.500	0.005	0.000	0.005

	Total	Dynamic	Quiescent
Supply Power (W)	0.125	0.043	0.082

Environment	
Ambient Temp (C)	25.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	0
Heat Sink	None
Custom TSA (C/W)	NA

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	15.0	83.1	26.9

Characterization	
Production	v1.3,2011-05-04

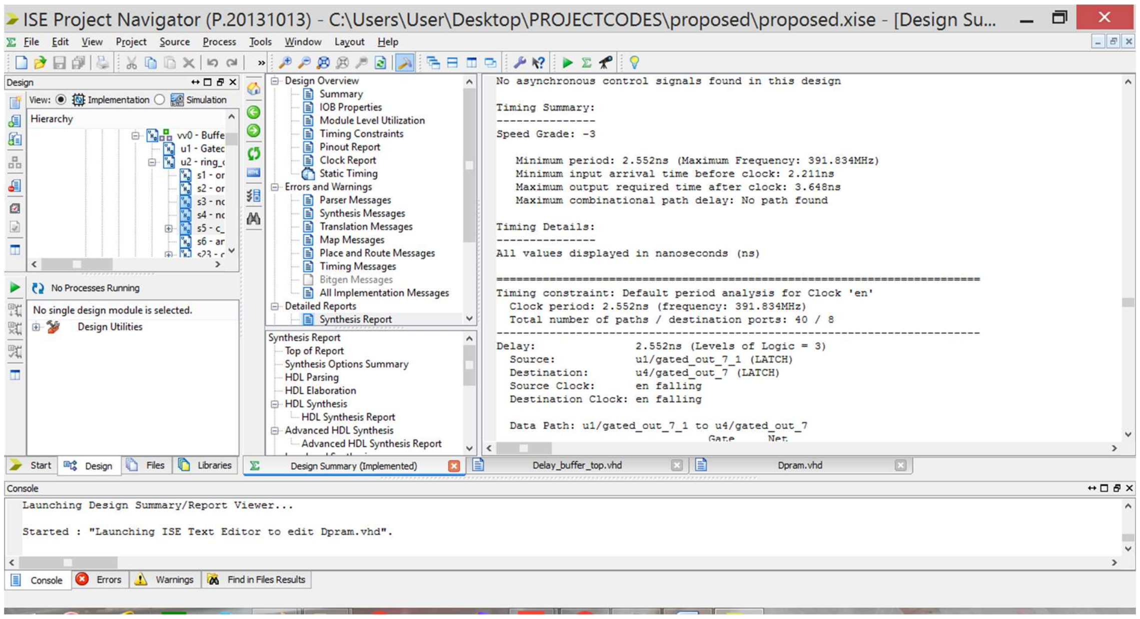
The Power Analysis is up to date.

(*) Place mouse over the asterisk for more detailed BRAM utilization.

```
domains will default to zero which may under-estimate the power for this
design.  To avoid this warning, provide at least one of the following:
  1. The proper timing constraints (PERIOD) for clocks (re-implement design
     and load the newly generated PCF file into XPower Analyzer)
  2. A post PAR simulation-generated VCD or SAIF file indicating clock
     frequencies
  3. The clock frequency for clocks in the "By Type -> Clocks" view in the
     XPower Analyzer GUI and then applying "Update Power Analysis"
```

Ready

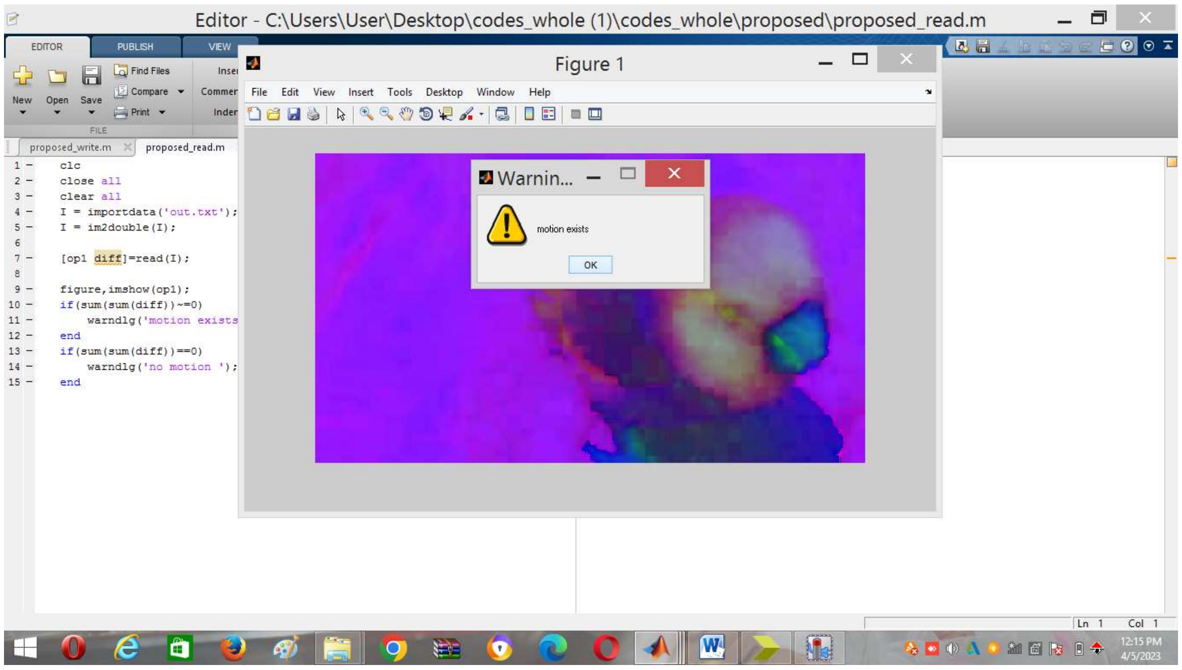
Fig 8: proposed power reduction analysis


ISE Project Navigator (P.20131013) - C:\Users\User\Desktop\PROJECTCODES\proposed\proposed.xise - [Design Su...

```
No asynchronous control signals found in this design

Timing Summary:
---------------
Speed Grade: -3

   Minimum period: 2.552ns (Maximum Frequency: 391.834MHz)
   Minimum input arrival time before clock: 2.211ns
   Maximum output required time after clock: 3.648ns
   Maximum combinational path delay: No path found

Timing Details:
---------------
All values displayed in nanoseconds (ns)

=========================================================================
Timing constraint: Default period analysis for Clock 'en'
  Clock period: 2.552ns (frequency: 391.834MHz)
  Total number of paths / destination ports: 40 / 8
-------------------------------------------------------------------------
Delay:               2.552ns (Levels of Logic = 3)
  Source:            u1/gated_out_7_1 (LATCH)
  Destination:       u4/gated_out_7 (LATCH)
  Source Clock:      en falling
  Destination Clock: en falling

  Data Path: u1/gated_out_7_1 to u4/gated_out_7
                                Gate     Net
```

```
Launching Design Summary/Report Viewer...

Started : "Launching ISE Text Editor to edit Dpram.vhd".
```

Fig 9: proposed Throughput analysis



```
clc
close all
clear all
I = importdata('out.txt');
I = im2double(I);

[op1 diff]=read(I);

figure,imshow(op1);
if(sum(sum(diff))~=0)
    warndlg('motion exists
end
if(sum(sum(diff))==0)
    warndlg('no motion ');
end
```

motion exists

Fig : Motion Exit

	EXISTING With out clock gating technique	**PROPOSED With clock gating technique**
POWER (mW)	266	125
THROUGHPUT(Mbps)	13	23

Table 2: Comparison table

CONCLUSION and FUTURE SCOPE:

Finally, proposed MRP approach for reducing excessive access to external memory caused by the global and iterative process of the global optical flow method while using internal memory resources minimally. The proposed approach divides an image into multiple small sub images consisting of multiple rows, i.e., multirow, to iteratively compute the flows using a small amount of internal memory instead of using external memory. The flows of each multi row were sequentially estimated without any overlap between multirows to avoid any degradation of the speed. In particular, to avoid discontinuity artifacts deriving from the boundaries of multirow, we introduced effective boundary conditions that exploited the estimated flows at the bottom row of the previous multirow for the flow estimation of the current multirow. In addition, the complex main equations of the global method are simplified using precomputations to boost the processing speed and reduce the complexity. The proposed system can be modified further to detect and classify objects with very high speed. Adoptation of recent numerical optimization and implementation techniques will be explored for future research.

REFERENCES:

[1] P. F. U. Gotardo, T. Simon, Y. Sheikh, and I. Matthews, “Photogeometric scene flow for high-detail dynamic 3D reconstruction,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Dec. 2015, pp. 846–854.
[2] S. Kumar, Y. Dai, and H. Li, “Monocular dense 3D reconstruction of a complex dynamic scene from two perspective frames,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Oct. 2017, pp. 4659–4667.
[3] S. Milz, G. Arbeiter, C. Witt, B. Abdallah, and S. Yogamani, “Visual SLAM for automated driving: Exploring the applications of deep learning,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW), Jun. 2018, pp. 360–370.
[4] D. Honegger, L. Meier, P. Tanskanen, and M. Pollefeys, “An open source and open hardware embedded metric optical flow CMOS camera for indoor and outdoor applications,” in Proc. IEEE Int. Conf. Robot. Autom., May 2013, pp. 1736–1741.
[5] F. Zhang, Y. Gao, and J. D. Bakos, “Lucas-Kanade optical flow estimation on the TI C66x digital signal processor,” in Proc. IEEE High Perform. Extreme Comput. Conf. (HPEC), Sep. 2014, pp. 1–6.
[6] M. Liu and T. Delbruck, “Block-matching optical flow for dynamic vision sensors: Algorithm and FPGA implementation,” in Proc. IEEE Int. Symp. Circuits Syst. (ISCAS), May 2017, pp. 1–4.
[7] J. Y. Bouguet, “Pyramidal implementation of the Lucas–Kanade feature tracker,” Microprocessor Res. Intel Corp., Labs, Tech. Rep., 2000.
[8] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, “High accuracy optical flow estimation based on a theory for warping,” in Proc. Eur. Conf. Comput. Vis., 2004, pp. 25–36.
[9] A. Bruhn, J. Weickert, and C. Schnörr, “Lucas/Kanade meets Horn/schunck: Combining local and global optic flow methods,” Int. J. Comput. Vis., vol. 61, no. 3, pp. 1–21, Feb. 2005.
[10] C. Liu, “Beyond pixels: Exploring new representations and applications for motion analysis,” M.S. thesis, Massachusetts Inst., Cambridge, MA, USA, Tech. Rep., 2009.
[11] T. Brox, C. Bregler, and J. Malik, “Large displacement optical flow,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2009, pp. 41–48.
[12] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid, “DeepFlow: Large displacement optical flow with deep matching,” in Proc. IEEE Int. Conf. Comput. Vis., Dec. 2013, pp. 1385–1392.
[13] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid, “EpicFlow: Edge-preserving interpolation of correspondences for optical flow,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2015, pp. 1164–1172.
[14] C. Bailer, B. Taetz, and D. Stricker, “Flow fields: Dense correspondence fields for highly accurate large displacement optical flow estimation,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 41, no. 8, pp. 1879–1892, Aug. 2019.

[15] Z. Chen, H. Jin, Z. Lin, S. Cohen, and Y. Wu, "Large displacement optical flow from nearest neighbor fields," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2013, pp. 2443–2450.
[16] J. Xu, R. Ranftl, and V. Koltun, "Accurate optical flow via direct cost volume processing," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jul. 2017, pp. 5807–5815. [17] Q. Chen and V. Koltun, "Full flow: Optical flow estimation by global optimization over regular grids," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 4706–4714.
[18] A. Dosovitskiy et al., "FlowNet: Learning optical flow with convolutional networks," in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Dec. 2015, pp. 2758–2766.
[19] A. Ranjan and M. J. Black, "Optical flow estimation using a spatial pyramid network," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 2720–2729.
[20] T.-W. Hui, X. Tang, and C. C. Loy, "LiteFlowNet: A lightweight convolutional neural network for optical flow estimation," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 8981–8989.
[21] D. Sun, X. Yang, M. Liu, and J. Kautz, "PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 8934–8943.
[22] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," Int. J. Comput. Vis., vol. 92, no. 1, pp. 1–31, Mar. 2011.
[23] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation," in Proc. Eur. Conf. Comput. Vis., 2012, pp. 611–625.
[24] N. Sundaram, T. Brox, and K. Keutzer, "Dense point trajectories by GPU-accelerated large displacement optical flow," in Proc. Eur. Conf. Comput. Vis., 2010, pp. 438–451.
[25] R. Ranftl, K. Bredies, and T. Pock, "Non-local total generalized variation for optical flow estimation," in Proc. Eur. Conf. Comput. Vis., 2014, pp. 439–454.
[26] D. Buyukaydin and T. Akgun, "GPU implementation of an anisotropic huber-L1 dense optical flow algorithm using OpenCL," in Proc. Int. Conf. Embedded Comput. Syst., Archit., Modeling, Simululation (SAMOS), Jul. 2015, pp. 326–331.
[27] F. Barranco, M. Tomasi, J. Diaz, M. Vanegas, and E. Ros, "Parallel architecture for hierarchical optical flow estimation based on FPGA," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 20, no. 6, pp. 1058–1067, Jun. 2012.
[28] H.-S. Seong, C. E. Rhee, and H.-J. Lee, "A novel hardware architecture of the Lucas–Kanade optical flow for reduced frame memory access," IEEE Trans. Circuits Syst. Video Technol., vol. 26, no. 6, pp. 1187–1199, Jun. 2016.
[29] G. K. Gultekin and A. Saranli, "An FPGA based high performance optical flow hardware design for computer vision applications," Microprocessors Microsyst., vol. 37, no. 3, pp. 270–286, May 2013.
[30] M. Komorkiewicz, T. Kryjak, and M. Gorgon, "Efficient hardware implementation of the Horn-Schunck algorithm for high-resolution realtime dense optical flow sensor," Sensors, vol. 14, no. 2, pp. 2860–2891, 2014.

[31] M. Kunz, A. Ostrowski, and P. Zipf, “An FPGA-optimized architecture of Horn and Schunck optical flow algorithm for real-time applications,” in Proc. 24th Int. Conf. Field Program. Log. Appl. (FPL), Sep. 2014, pp. 1–4.
[32] W. Chen, Z. Wang, Q. Wu, J. Liang, and Z. Chai, “Implementing dense optical flow computation on a heterogeneous FPGA SoC in C,” ACM Trans. Archit. Code Optim., vol. 13, no. 3, pp. 1–25, Sep. 2016.
[33] J. Lee, C. Kim, S. Choi, D. Shin, S. Kang, and H.-J. Yoo, “A 46.1 fps global matching optical flow estimation processor for action recognition in mobile devices,” in Proc. IEEE Int. Symp. Circuits Syst. (ISCAS), May 2018, pp. 1–5. [34] B. K. P. Horn and B. G. Schunck, “Determining optical flow,” Artif. Intell., vol. 17, nos. 1–3, pp. 185–203, Aug. 1981.