Microprocessors and Microsystems 80 (2021) 103514

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

Real time FPGA implementation of a high speed and area optimized Harris corner detection algorithm

Prateek Sikka *, Abhijit R. Asati, Chandra Shekhar

Electrical and Electronics Engineering Department, Birla Institute of Technology and Science, Vidya Vihar Campus, Pilani, Rajasthan, India

ARTICLE INFO

Keywords:
Hardware description language
Register transfer language
Field-programmable gate array
High-Level Synthesis
Harris corner detection
Vivado
MATLAB HDL coder

ABSTRACT

Harris corner detection is an algorithm frequently used in image processing and computer vision applications to detect corners in an input image. In most modern applications of image processing, there is a need for real time implementation of algorithms such as Harris corner detection in hardware systems such as field-programmable gate arrays (FPGAs). FPGAs allow faster algorithmic throughput, which is required to match real time speeds or cases where there is a requirement to process faster data rates. High level synthesis tools offer higher abstraction level to designers with continued verification during the design flow and hence are getting popular with the design community. This paper proposes a high speed and area optimized implementation of a Harris corner detection algorithm. The proposed implementation was actualized using a novel high-level synthesis (HLS) design method based on application-specific bit widths for intermediate data nodes. Register transfer level (RTL) code was generated using MATLAB HDL coder for HLS. The generated hardware description language (HDL) code was implemented on Xilinx ZedBoard using Vivado software and verified for functionality in real time with input video stream. The obtained results are superior to those of previous implementations in terms of area (smaller gate count on target FPGA) and speed for the same target board.

1. Introduction

For most modern image processing and computer vision systems, extracting the region of interest remains a fundamental problem. This is required in a variety of applications such as advanced driver assistance systems (ADAS) for pedestrian, traffic signal, and blind spot detection; lane departure warning systems; video surveillance applications; and simultaneous localization and mapping (SLAM) [1]. A corner, or a point where two sharp edges meet, is one such feature in an image. Multiple algorithms are used to detect corners in images. Some frequently used corner extraction algorithms include the Moravec algorithm [2], the Susan algorithm, and the Harris corner detector. The Harris corner detector is one of the most precise corner detection algorithms. Although its operation is notably simple, the algorithm is computationally intensive. It is typically used in systems that require real time data processing; thus, conventional CPUs cannot meet the requirements. CPUs are good only if large data volumes are involved or we need to perform floating point computations. Hence, Field-programmable gate arrays (FPGAs) are excellent candidates for deploying such algorithms in real time owing to fast processing speeds and parallel implementations. For complex algorithms implemented on FPGAs, corner detection may have to be deployed in addition to other algorithms on the FPGA. Examples include non-maxima suppression, matching using the sum of absolute differences, matrix computation, and triangulation [3]. Because of this requirement, it is important to improve the efficiency and area requirements of FPGA implementations for these algorithms.

Many researchers have published novel studies on area- and speed-efficient implementations of the Harris corner detector on FPGAs. Liu et al. [4] recently proposed a method that can process RGB565 video in 640 × 480 resolution at a rate of 154 frames per second. Liu et al. implemented the design using a Xilinx ZedBoard. Xu et al. [5] proposed a slightly different algorithm by adding a pre-filter and using a simplified matrix rather than the original Gaussian kernel matrix. This reduced the design complexity and led to efficient hardware resource usage with Spartan 3 FPGA for robotics applications. Experiments conducted with an input image having a resolution of 256 × 256 yielded a processing time of 2.3 ms. Chao et al. [6] attempted to simplify the maximum suppression procedure with the Harris corner detector. Their design, which was specifically developed for use with ZedBoard, achieved a data rate of 144 frames per second in simulations. Lee et al. [7] proposed a

* Corresponding author.
E-mail addresses: prateeksikka@gmail.com (P. Sikka), abhijit_asati@pilani.bits-pilani.ac.in (A.R. Asati), chandra.shekhar@pilani.bits-pilani.ac.in (C. Shekhar).

https://doi.org/10.1016/j.micpro.2020.103514
Received 27 July 2020; Received in revised form 11 September 2020; Accepted 20 November 2020
Available online 26 November 2020

modified Harris corner detector for breast cancer detection from MRI and x-ray images. They used an automated adaptive radius suppression technique, which reduces corner clustering; thus, avoiding the loss of useful corners due to over-suppression. John et al. [8] proposed a generic image feature extractor algorithm and implemented the same on Cyclone 4 FPGA for real time processing achieving a frame rate of 70 frames per second. Hisham et al. [9] used dynamic partial reconfiguration to design a self-adaptive system on chip for Harris corner detection algorithm. Their implementation dissipated less power with a small overhead on performance.

This paper proposes a real time implementation of the Harris corner detector on a Xilinx ZedBoard and demonstrate that the implementation is superior in terms of speed and area usage on the FPGA to previous implementations. The design was developed using a novel high-level design method that synthesizes the design with intermediate signal widths constrained according to the application (input stimulus). The remainder of this paper is organized as follows: Section 2 presents an introduction to high-level synthesis. Section 3 explains the architecture of the Harris corner detector. Section 4 explains the methodology used in the proposed design. Section 5 presents simulation and synthesis results, as well as a comparison with the results presented by other researchers. Section 6 presents concluding remarks.

2. High-level synthesis

High-level synthesis (HLS) is gaining momentum as a methodology that can ensure continued verification throughout the design cycle while allowing designers to describe design behaviors at high abstraction levels. HLS tools include Vivado HLS [10] and MATLAB HDL Coder [11] as well as several open-source tools. Digital designers and architects typically use these to design and deploy algorithms that target varied aerospace, communications, image processing, deep learning, and neural network applications. HLS tools help reduce code complexity by factors ranging from seven to ten. They allow for behavioral IP to be re-used across projects and enable verification teams to use high-abstraction-level modeling techniques such as transaction-level modeling [12].

Furthermore, most contemporary chip systems have embedded processors. Additional software or firmware is involved in the design process owing to the co-existence of microprocessors, digital signal processors (DSPs), memories, and custom logic on a single chip. Hence, an automated HLS process allows designers and architects to experiment with different algorithmic and implementation choices to explore various area, power, and performance tradeoffs from a common functional specification.

Accordingly, the industrial deployment of HLS tools has become more practical with improvements in register transfer level (RTL) synthesis tools.. Proprietary tools have been built by major semiconductor design houses, including IBM [13], Motorola [14], Philips [15], and Siemens [16]. Major Electronic Design Automation (EDA) vendors have also begun to commercialize different HLS tools. In 1995, for instance, Synopsys introduced the “Behavioral Compiler” tool [17], which generates RTL implementations from behavioral hardware description language (HDL) code and connects to downstream tools. Similar tools include “Catapult HLS” from Mentor Graphics [18] and “Stratus High Level Synthesis” from Cadence [19]. A typical flow for HLS in VLSI designs is shown in **Fig. 1**.

3. Harris corner detector

In images, a corner is a point where both gradients of the orthogonal axes are high. The Harris corner detection algorithm detects points in the image where these conditions are met. The algorithm takes a small region of an image and determines whether the window contains corner features. Assume I(x,y) is the image point; the gradient matrix M can be calculated using Eq. (1) as follows:

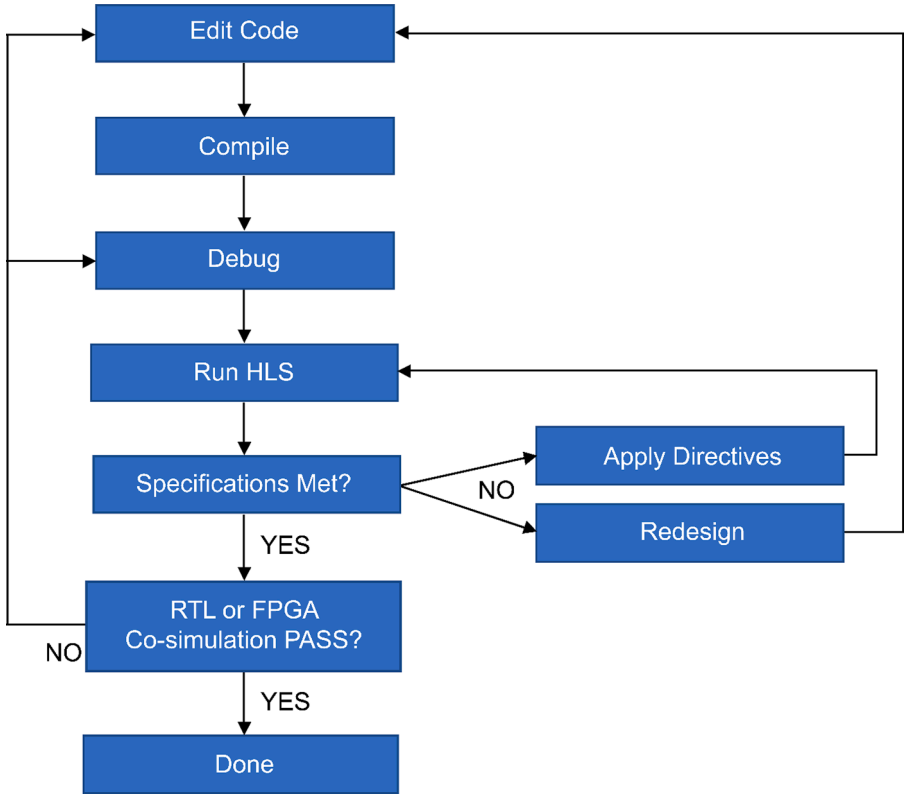

Fig. 1. High Level Synthesis Flow.

$$M = \begin{bmatrix} \sum_{i,j}^{W} \left(\frac{\partial I(i,j)}{\partial x}\right)^2 & \sum_{i,j}^{W} \left(\frac{\partial I(i,j)}{\partial x}\right)\left(\frac{\partial I(i,j)}{\partial y}\right) \\ \sum_{i,j}^{W} \left(\frac{\partial I(i,j)}{\partial x}\right)\left(\frac{\partial I(i,j)}{\partial y}\right) & \sum_{i,j}^{W} \left(\frac{\partial I(i,j)}{\partial x}\right)^2 \end{bmatrix} \quad (1)$$

where i and j are the pixel indices of a window of range *W*. If the eigenvalues of matrix M are high, it is a corner feature point. Finding the eigenvalue is computationally intensive; thus, an approximation formula is used instead, as presented in Eq. (2).

$$R = \mathrm{Det}(M) - k * \mathrm{Trace}(M)^2 \quad (2)$$

where the value of k is approximately 0.04–0.06. If R is larger than the threshold, the center pixel in the window is a corner feature candidate. By scanning the windows at different positions of the entire image, all feature candidates can be found. The Harris algorithm comprises five primary steps: gradient calculation, Gaussian smoothing, Harris value calculation, thresholding, and non-maxima suppression. **Fig. 2** depicts the cascaded stages of the Harris corner detection algorithm.

4. Design methodology

In this study, a cascaded model of the Harris corner detector was created in MATLAB using input video frames of resolution 240 (width) x 320 (Height) x 3 (Colors) with eight bits representing each color. Because HDL implementation uses pixels instead of frames, the input frames were converted to pixels and each pixel was sent as an input to the design per clock cycle. A Simulink library block was also concurrently used for the Harris corner detector, with the Simulink library block obtaining the same input images, and the results were subsequently compared against the hardware implementation.

During the simulation time of 100 s-the duration of the video stream, the absolute minima and maxima for the inputs, outputs, and intermediate nodes in the design were recorded. This database of minima and maxima was appended on subsequent simulation runs until all varied video signal inputs were covered. Subsequently, the width for all the signals was recalculated based on the range of the values for each input, output, and the intermediate nodes. The HLS tool was then constrained to generate the RTL with the updated constraints for all the data nodes along with the primary inputs and outputs. The optimized RTL was then implemented on the FPGA. As the output from FPGA is in the form of pixels, it was converted back to a frame using MATLAB. The output image was a corner-marked image overlaid onto the input image. Fig. 3 shows the complete model with HDL implementation, behavioral implementation, and the common image source. As illustrated in Fig. 3, delay elements were added to the behavioral and input image paths to balance the delay of the actual cycle accurate hardware implementation running on the FPGA.

Fig. 4 displays the input image, the output image from the MATLAB model, and the output from the HDL implementation on the FPGA. As described in the same figure, the input image source, the MATLAB behavioral model, and the FPGA HDL model ran independently and processed different frames from the input video stream..

The optimum datatype and data widths selected for the model in RTL was back-annotated to the HLS model. The back-annotated datatypes for an intermediate stage (gradient calculation) in the model are shown in **Fig. 5**. In the highlighted example, two 18-bit signed fixed-point numbers produced a 36-bit result. In the pessimistic data width model, all widths were 32-bit, which is suboptimal.

Although this method was used for a MATLAB HDL coder-based design for Harris corner detection, the generic methodology is compatible for all HLS tools and all FPGA applications. This method is easily scalable for any number of inputs, outputs, and internal signals that the design may contain. For this study's case, the method of calculation of minima and maxima for each node was fully automated and did not require any manual intervention. Even though we chose 8 bits to represent each color, the optimization method is independent of that. This means we could choose any other datatype and any frame rate for input image. Furthermore, all stages of the algorithm depicted in Fig. 2 are fully pipelined. This leads to a better design throughput at the cost of a slightly increased number of flip-flops.

5. Results and discussion

5.1. FPGA implementation

The RTL code from the HLS model was generated using MATLAB HDL coder with stimulus-aware bit widths. The RTL code was synthesized using Vivado 2019.1 software and implemented on the Xilinx ZedBoard.

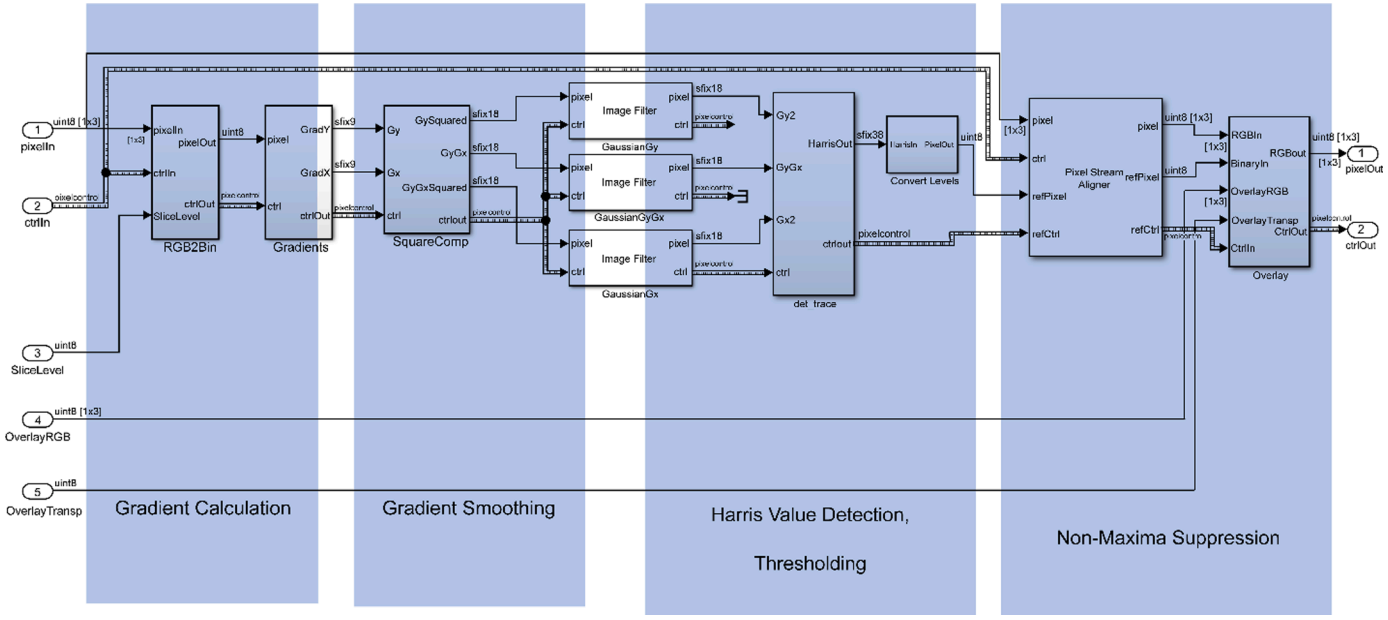

Fig. 2. Cascaded stages of Harris Corner Detector.

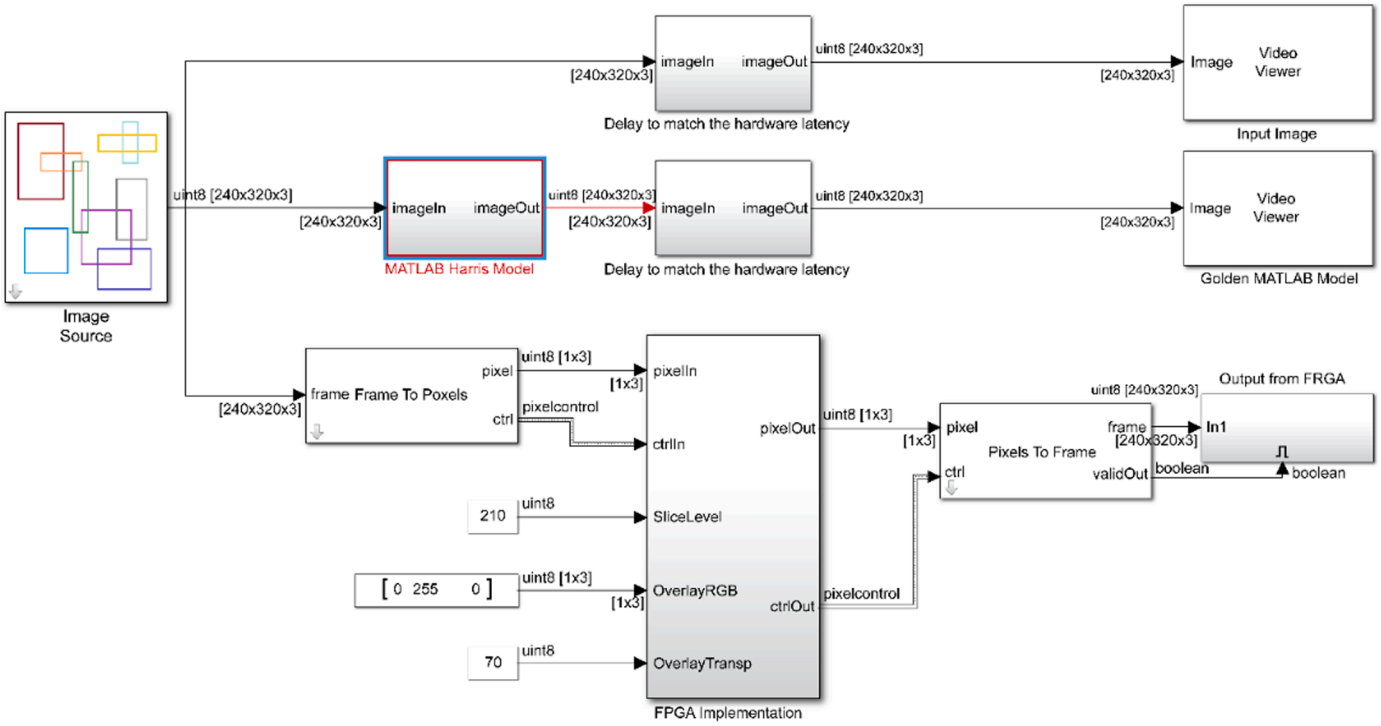

Fig. 3. Harris Corner Detector: Full Model.

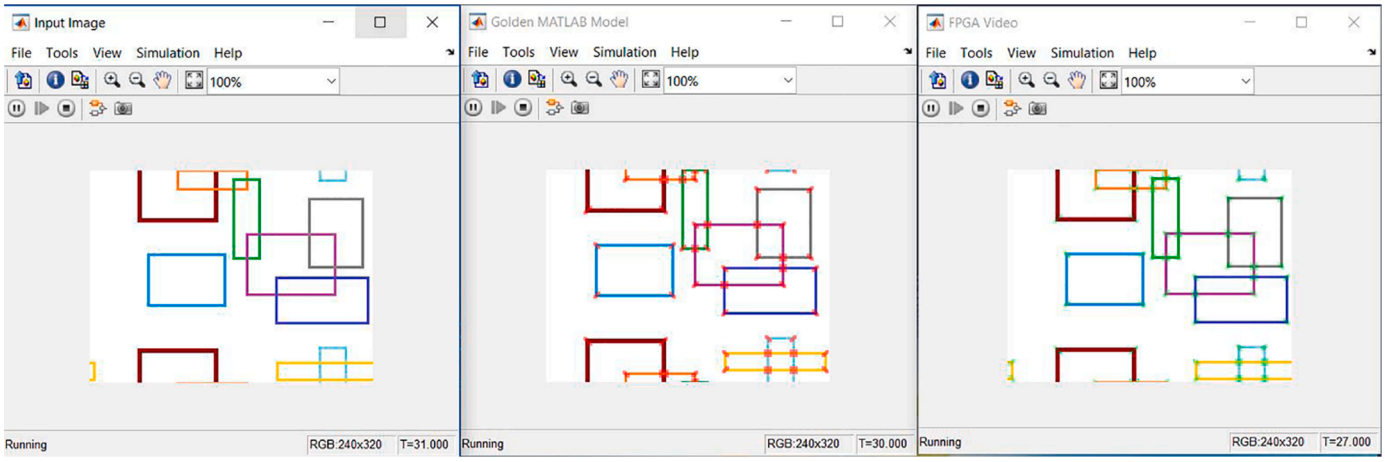

Fig. 4. Input image, output image from model, output image from HDL(FPG).

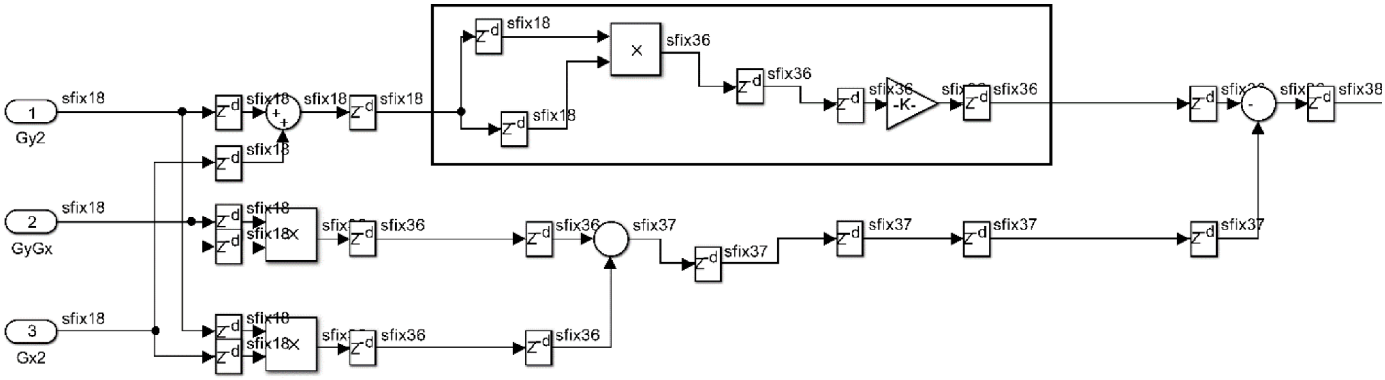

Fig. 5. Internal signals data widths: Harris corner detector.

5.2. Simulation results

The Verilog RTL code generated was simulated with a non-synthesizable testbench using Vivado xSim software. Functional simulation results from the Vivado xSim simulator are presented in **Fig. 6**. As demonstrated in the figure, the reference pixel values are identical to the pixel output from the proposed method. They were also found to be identical to high-level simulation results observed with MATLAB on the optimum bit-width model. This was confirmed after comparison of output images from both paths with the same input image. This study also measured the quantization error introduced because of the choice of the reduced "optimal" signal widths against the double precision model running in MATLAB. This was done using the "FPGA in the loop" co-simulation feature of HDL Verifier from MathWorks [20]. The measured Root Mean Square (RMS) value of the quantization error was less than 1%.

5.3. Synthesis results

The proposed Harris corner detector design was created using optimum bit-width data types, as explained in Section 4, and simulated on XSim, as mentioned in Section 5.2. The same Verilog design was synthesized using Vivado 2019.1 targeting the Xilinx ZedBoard. Table 1 presents the synthesis and implementation results for the proposed design.

The total power reported by the Vivado synthesis tool was 0.315 W, comprising 0.205 W dynamic power and 0.110 W static power. In addition, high-level synthesis tools also offer many additional optimization techniques such as resource sharing and pipelining. These can be used to further improve the above results; however, such optimization is beyond the scope of this study.

5.4. Comparison results

The synthesis results of the proposed algorithm were compared to other algorithms reported in the literature [1,2] along with a reference design created by us without using the proposed methodology. Table 2 summarizes the results of this comparison. As presented in Table 2, the implementation proposed in this study is better in terms of frame rate and area usage for the same target FPGA than other implementations. With regard to area, even with the assumption that a Look Up Table (LUT) uses two Application Specific Integrated Circuit (ASIC) Gates, our implementation used almost 50% less resources than Komorkiewicz et al., 30% less resources than Liu et al., and 20% less resources than Chao et al. Furthermore, regarding the frame rate, our implementation has a better performance than other implementations owing to reduced design complexity. This is because critical paths are shorter and hence a better operational frequency is seen on the FPGA. Other authors have not shared power dissipation data; however, the reduced design complexity of the proposed design leads us to believe that our proposed implementation can display similar comparative results in terms of power dissipation.

Table 1
Proposed Harris corner detector implementation results.

Resource	Utilization	Available	Percentage Utilization
Look Up Table (LUT)	5917	53,200	11.12%
LUT RAM	506	17,400	2.91%
Flip-Flops	10,981	106,400	10.32%
BRAM	37	140	26.79%
DSP	21	220	9.55%
IOs	102	200	51%
BUFG	1	32	3.13%

6. Conclusion

This paper proposed a high speed and optimal (low) area implementation of the Harris corner detection algorithm that is suitable for real time deployment on FPGAs. The design was actualized using a novel HLS design method that constrains the intermediate nodes and primary ports of the design in accordance with the absolute minima and maxima for each node. The generated RTL for the algorithm was implemented on a Xilinx ZedBoard using real time video stream as the input with a resolution of 240 × 320 and 8-bit color inputs. The RTL design was functionally verified using Vivado xSim simulator from Xilinx. We observed that the quantization errors introduced in the implementation using our HLS method was less than 1%. The synthesis results suggest that our implementation performs better than similar extant architectures. Hence, this implementation is well-suited for applications on FPGAs that require real time image processing. Even though this was verified for MATLAB HDL coder workflow and targeting Xilinx Zedboard, the method is applicable to other tools and other FPGA targets as well (technology independent). Even though we do not have results, by virtue of design method, we also believe that this methodology would give better results with ASIC synthesis as well. Further work will involve improvement of the speed, area, and power usage by using optimization methodologies offered by high-level synthesis tool vendors such as MathWorks, Xilinx, Mentor, and Cadence in addition to the proposed implementation method. We also plan to extend this study for ASIC designs in future.

Funding

This research did not receive any grants from funding agencies in the public, commercial, or not-for-profit sector.

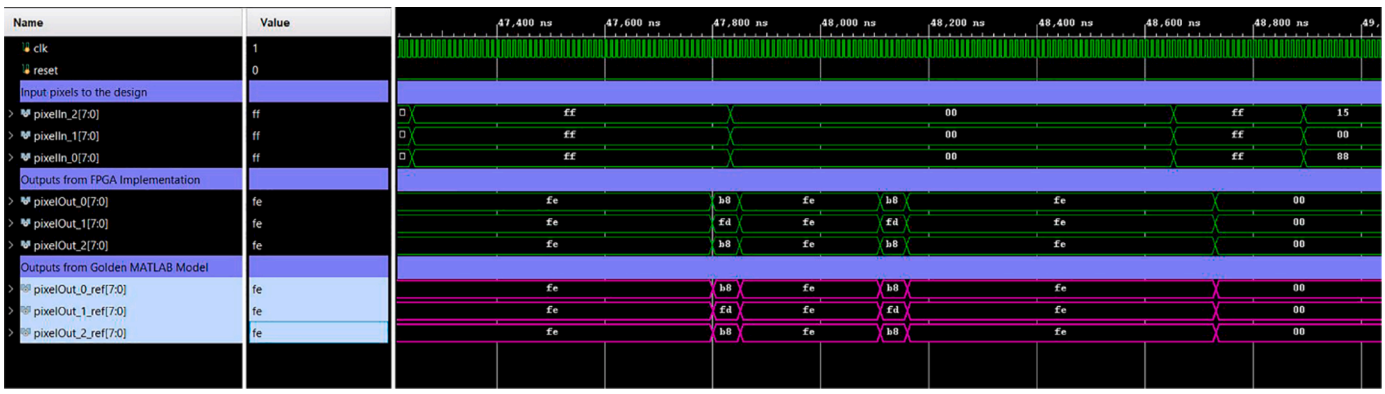

Fig. 6. HDL simulation results (Harris Corner Detector).

Table 2
Comparison of the results for the Harris corner detector.

Resource	Proposed Implementation	Reference 3 Komorkiewicz et al.	Reference 4 Liu et al.	Reference 6Chao et al.
Total Gate Count (2*LUT +FF)	22,815	51,859	37,447	28,626
LUT	5917	20,660	17,555	9485
LUTRAM	506	NA	5443	4131
FF	10,981	10,539	2337	9656
BRAM	37	77	75	64
DSP	21	59	55	110
IOs	102	NA	48	NA
BUFG	1	NA	7	NA
Frame Rate	168 fps	120 fps (Full HD)	154	98
Power Consumption	0.315 W	Not Available	Not Available	Not Available

Geolocation information

India.

Declaration of Competing Interest

None.

Acknowledgments

None.

References

[1] V.H. Schulz, F.G. Bombardelli, E. Todt, A Harris corner detector implementation in SoC-FPGA for visual SLAM, in: F. Santos Osório, R. Sales Gonçalves (Eds.), Robotics. SBR 2016, LARS 2016. Communications in Computer and Information Science 619, Springer, Cham., 2016.
[2] C. Cabani, Implementation of an Affine-Invariant Feature Detector in Field-Programmable Gate Arrays, University of Toronto, 2006, pp. 5–13.
[3] M. Komorkiewicz, T. Kryjak, K. Chuchacz-Kowalczyk, P. Skruch, M. Gorgoń, FPGA based system for real time structure from motion computation, in: 2015 Conference on Design and Architectures for Signal and Image Processing (DASIP), Krakow, 2015, pp. 1–7, https://doi.org/10.1109/DASIP.2015.7367241.
[4] S. Liu, Real time implementation of Harris corner detection system based on FPGA, in: 2017 IEEE International Conference on Real time Computing and Robotics (RCAR), Okinawa, 2017, pp. 339–343, https://doi.org/10.1109/RCAR.2017.8311884.
[5] C. Xu, B. Yunshan, Implementation of Harris corner matching based on FPGA, in: 2017 6th International Conference on Energy and Environmental Protection (ICEEP 2017)., Atlantis Press, 2017.
[6] T.L. Chao, H.W. Kin, An efficient FPGA implementation of the Harris corner feature detector, in: 2015 14th IAPR International Conference on Machine Vision Applications (MVA)., IEEE, 2015.
[7] C.Y. Lee, H.J. Wang, C.M. Chen, C.C. Chuang, Y.C. Chang, N.S. Chou, A modified Harris corner detection for breast IR image, Math. Probl. Eng. 2014 (2014).
[8] J. Vourvoulakis, J. Kalomiros, J. Lygouras, Fully pipelined FPGA-based architecture for real-time SIFT extraction, Microprocess. Microsyst. 40 (2016) 53–73.
[9] H. Ahmed, O. Sidek, An energy-aware self-adaptive System-on-Chip architecture for real-time Harris corner detection with multi-resolution support, Microprocess. Microsyst. 49 (2017) 164–178.
[10] Xilinx, (2018) Vivado design suite: high-level synthesis. https://www.xilinx.com/support/documentation/sw_manuals/xilinx2018_3/ug902-vivado-high-level-synthesis.pdf (accessed 12 Sep, 2019).
[11] MathWorks HDL coder. (2019) https://www.mathworks.com/products/hdl-coder.html, (accessed 14 Aug, 2019).
[12] J. Cong, B. Liu, S. Neuendorffer, J. Noguera, K. Vissers, Z. Zhang, High-level synthesis for FPGAs: from prototyping to deployment, IEEE T. Comput. Aid. D 30 (2011) 473–491. https://doi.org/10.1109/TCAD.2011.211059.
[13] R.A. Bergamaschi, R.A. O'Connor, L. Stok, M.Z. Moricz, S. Prakash, A. Kuehlmann, D.S. Rao, High-level synthesis in an industrial environment, IBM J. Res. Develop. 39 (1995) 131–148. https://doi.org/10.1147/rd.391.0131.
[14] K. Kucukcakar, C.T. Chen, J. Gong, W. Philipsen, T.E. Tkacik, Matisse: an architectural design tool for commodity ICs, IEEE DesTest Comput. 15 (1998) 22–33. https://doi.org/10.1109/54.679205.
[15] P.E.R. Lippens, J.L. van Meerbergen, A. van der Werf, W.F.J. Verhaegh, B. T. McSweeney, J.O. Huisken, O.P. McArdle, PHIDEO: a silicon compiler for high speed algorithms, in: Proceedings of the European Conference on Design Automation, IEEE Computer Society Press, Amsterdam, 1991, pp. 436–441.
[16] J. Biesenack, M. Koster, A. Langmaier, S. Ledeux, S. Marz, M. Payer, M. Pilsl, S. Rumler, H. Soukup, N. Wehn, P. Duzy, The Siemens high-level synthesis system CALLAS, IEEE Trans. Very Large Scale Integr. Syst. 1 (1993) 244–253. https://doi.org/10.1109/92.238438.
[17] D.W. Knapp, Behavioral Synthesis: digital System Design Using the Synopsys Behavioral Compiler, Prentice-Hall, Englewood Cliffs, NJ, 1996.
[18] Catapult H.L.S., (2019) https://www.mentor.com/hls-lp/catapult-high-level-synthesis/.
[19] Stratus H.L.S., (2019) https://www.cadence.com/content/cadence-www/global/en_US/home/tools/digital-design-and-signoff/synthesis/stratus-high-level-synthesis.html.
[20] MathWorks HDL Verifier, (2019) https://in.mathworks.com/products/hdl-verifier.html.

Prateek Sikka is currently pursuing his doctorate at BITS Pilani, Rajasthan, India while working for NXP Semiconductors in Noida, Uttar Pradesh, India. He has over 13 years of experience serving in multiple EDA and semiconductor organizations, such as STMicroelectronics, Cadence, and Mentor Graphics in both R&D and customer facing roles. His-research areas include digital systems design and optimization, FPGA prototyping and emulation, and high-level synthesis. He holds five patents in the same field and has presented at multiple conferences. He holds a Bachelor of Engineering degree from Thapar Institute Patiala and a Master of Technology degree in Integrated Electronics from IIT, Delhi.

Dr. Abhijit Asati is a senior member at IEEE and is currently an associate professor in the Electronics and Electrical Engineering department at BITS, Pilani, Rajasthan, India. He completed his doctorate in 2010, his Master of Engineering degree in 2002 from BITS Pilani, and his Bachelor of Engineering in 1996 from Amravati University. He has taught several courses in electrical and electronics engineering and has supervised three doctoral and 25 bachelors' and masters' theses. He taught at VNIT Nagpur from 1997 to 1999. He has contributed to over 30 journal and conference papers in the fields of microelectronics, VLSI design, and embedded systems.

Dr. Chandra Shekhar currently serves as Professor Emeritus at BITS, Pilani, Rajasthan, India. After attaining his PhD, he joined the Solid State Devices Division at CEERI, Pilani in 1977 as a scientist. Having worked for IC Design related activities, he also served as the director at CEERI, Pilani. He was awarded the UNESCO/ROSTSCA Young Scientist Award in 1986, the CEERI Foundation Day Merit award in 1988, the ISHEER Science Councilor Award in 2005, the Prof. L K Maheshwari Foundation Distinguished Alumnus Award in 2010, a Doctor of Science (honoris causa) by NIT Kurukshetra in 2012, and the IETE Diamond Jubilee Gold Medal award in 2013.