TELKOMNIKA Telecommunication, Computing, Electronics and Control
Vol. 19, No. 2, April 2021, pp. 672~682
ISSN: 1693-6930, accredited First Grade by Kemenristekdikti, Decree No: 21/E/KPT/2018
DOI: 10.12928/TELKOMNIKA.v19i2.18158

Hardware accelerator for anti-aliasing Wu's line algorithm using FPGA

Basma M. K. Younis, Ahmed Kh. Younis
Department of Computer Technology Engineering, Engineering Technical College, Northern Technical University, Iraq

Article Info

Article history:

Received Jun 6, 2020
Revised Nov 26, 2020
Accepted Dec 4, 2020

Keywords:

Anti-aliasing
HLS
Multi-core
Wu's algorithm
Zynq702 FPGA

ABSTRACT

Digital images are suffering from the stair-step effect because they are built from small pixels. This effect termes aliasing and the method uses to decrease so-called anti-aliasing. This paper offers a hardware accelerator of an anti-aliasing algorithm using HLS (high level synthesis) along straight-line segments or edges. These straight-line segments are smoothed by modifying the intensity of the pixel. The hardware implementation of two different architectures which is based on Zynq FPGA are presented in this work. The first architecture is built from one core while the second architecture is built from multi-core and uses a parallel technique to speed up the algorithm by dividing line segments into sub-segments and drawing them after smoothing instantaneously to formulate the main line. This parallel usage leads to a very fast execution of Wu's algorithm which is represented one-tenth hardware runtime for one core only. Also, the optimized resource utilization and power consumption for different cores have been compared, through single-core design which utilizes 8% and consumes 1.6 W, while utilized resources using 10 cores are 77% with a power consumption of 2 W.

Corresponding Author:

Ahmed Kh. Younis,
Department of Computer Technology Engineering
Engineering Technical College
Northern Technical University, Iraq
Email: ahmedkhazal@ntu.edu.iq

1. INTRODUCTION

Digitizing continuous 2D graphic primitives is one of essential procedures in computer graphics that must be done at the sampling rate of device resolution. Losses in information during this important process produce aliasing (a staircaseeffect) which can be avoided by increasing the resolution of the raster device and consequently cost which is not an economic solution or using available techniques which utilize gray scales to increase the effective spatial resolution [1, 2]. This paper uses the famous Wu's algorithm [1] that efficiently smooth object's edges by antialiased line generator and in order to speed up the rendering time, two hardware solution are proposed which can be embedded in a larger design to integrate a computer graphics system. On the other hand, Bresenham's algorithm is one of the first published algorithms [3] for plotting straight lines on a display device or a plotter where the grid over which the discrete points (or pixels) of the line are drawn . This algorithm was modified later by many researchers, it is also extended to work in three-dimension field and used in several applications and in hardware implementation as can be seen in the following works:

Researchers in [4] try to speed up this algorithm using the properties of linear Diophantine equations in order to obtain a speed factor of almost five in scan converting a line segment. They claimed that their proposed technique could be easily implemented on most hardware systems. Other researchers in [5] offered a

new fast algorithm for line drawing differ from the original Bresenham's algorithm. In this paper, a fast line drawing algorithm, which totally based on integer calculations is 3 times faster than Bresenham's algorithm in the feature of average time. The optimization is done by using slope symmetry. Comparing with Bresenham's algorithm, this algorithm reaches an important increase in entire efficiency. Additional advantage of the algorithm is its simplicity and compatibly for hardware implementation similar to Bresenham algorithm. The three-dimensional extension based on the same idea of Bresenham's algorithm is used in [6] based on minimum distance between the grid points and the plotted line. The organization of the Voronoi diagram is proposed for grid points to which line pixels may be approximated. To increase the efficiency of the calculation, integer arithmetic and symmetry are also used for their 3D extension of the algorithm. The simplification of the three-dimension Bresenham's algorithm is accomplished to be compatible for hardware requirement during the implementation stage [7]. In this work, all the hardware outputs are compared to that results from OpenGL product for verification. The graphic sub-system for three-dimension Bresenham's algorithm is done for real-time applications using Spartan3E FPGA.

The designed soft processor with a 3D graphics coprocessor [8] in Line Generator stage uses Bresenham's line drawing algorithm on the cheapest available FPGA board with HDMI connector, containing only 8K logic elements. Also, a 3D Stereo Rendering architecture, presented in [5], is successfully tested and results are proven. These results include the performance of the 3D rendering operations which is based on off-axis technique to create stereo pairs and Bresenham's line drawing algorithm to draw objects, which have been implemented on FPGA hardware. Also, researchers in [9], improved an approach to speed up the Bresenham algorithm by partitioning each line into a number of segments, finding the points belong to those segments, and then formulating the overall line by drawing them simultaneously. By employing 32 cores in the Field Programmable Gate Array, a line of length 992 points is formulated in 0.31μs only. The whole system is carried out using Zybo board which includes the Xilinx Zynq-7010 chip.

Although Bresenham's algorithm plots lines very rapidly, but it does not solve anti-aliasing oppositely. Wu's algorithm which has anti-aliasing function is relatively fast but is certainly slower than Bresenham's algorithm. The algorithm involves drawing pairs of pixels along the line, each colored according to its distance from the line. Pixels at the line ends are calculated alone. Aliasing along edges or straight-line segments is investigated by researcher in [10] from different points of view including its origin and effect of line slope. Then, the aliasing problem is solved by modifying gray level of each pixel to produce a smooth line segment. Hardware implementation of this method is finally formulated and tested using field programmable gate arrays (FPGA).

2. THEORY

A deep study of antialiasing effect was done with the goal of mixing two graphics methods to use in a computer graphics hardware accelerator for acceptable performance. So, this section is divided into two parts. The first part explains the theoretical part of Bresenham's line drawing algorithm, while the second part explains the details of Xiaolin Wu's line algorithm since both of them have been used in the design of the proposed system. The detailed discussion for the above two computer graphics theories are presented with numerical and experiment examples are presented in the next sections to show results that may affect the designed hardware.

2.1. Bresenham's line generation algorithm

The 2D Bresenham's algorithm is an incremental scan conversion algorithm using the minimum difference between distances to calculate pixels' positions. This is done by moving through the x-axis in one pixel intervals at each step selecting between two diverse y coordinates that are nearer to the original line [3, 7]. The Bresenham's algorithm has the advantages of being a fast-incremental algorithm using only integer calculations. The simplest form for this algorithm is shown in Figure 1. The only drawback in this algorithm is the aliasing effect which can be enhanced. This used algorithm in this work is to calculate the endpoints for individual line segments, then the Wu's algorithm is used to display these anti-aliased line segments [11, 12]. Although Wu's algorithm is usually used in modern computer graphics because it can draw smooth lines or solves aliasing effect. The rapidity and simplicity of Bresenham's line drawing algorithm make it still imperative. It is essential in many software graphics libraries and it is also used in the hardware design of contemporary graphics cards [7, 13].

2.2. Xiaolin Wu's line algorithm

The Wu's algorithm is relatively slower than Bresenham's algorithm, but it solves the aliasing problem. It distributes intensity between nearest neighbors so that the total one is constant, but the intensity of each pixel is determined by its relative distance from the line [1]. A magnified view of the relation between the desired line and its neighbor pixels is shown in Figure 2 [1, 14]. The upper part from this figure is for a line drawn using Bresenham's algorithm, while the lower part is the producers of the Wu's algorithm. According to Wu's

algorithm (at each step), the calculation is made for the two closest to line pixels, and they have different intensity according to their distance from original line. The desired line is drawn in yellow, while the distance to the nearest cell is either green for lower pixel or red for upper pixel. If these distances are equal, they will have the same color each 50% intensity [2]. Otherwise, the intensity is divided between the pixels on both sides of the line as illustrated in Table 1. Finally, the traditional Wu's line antialiasing algorithm that was concluded is shown in Figure 3.

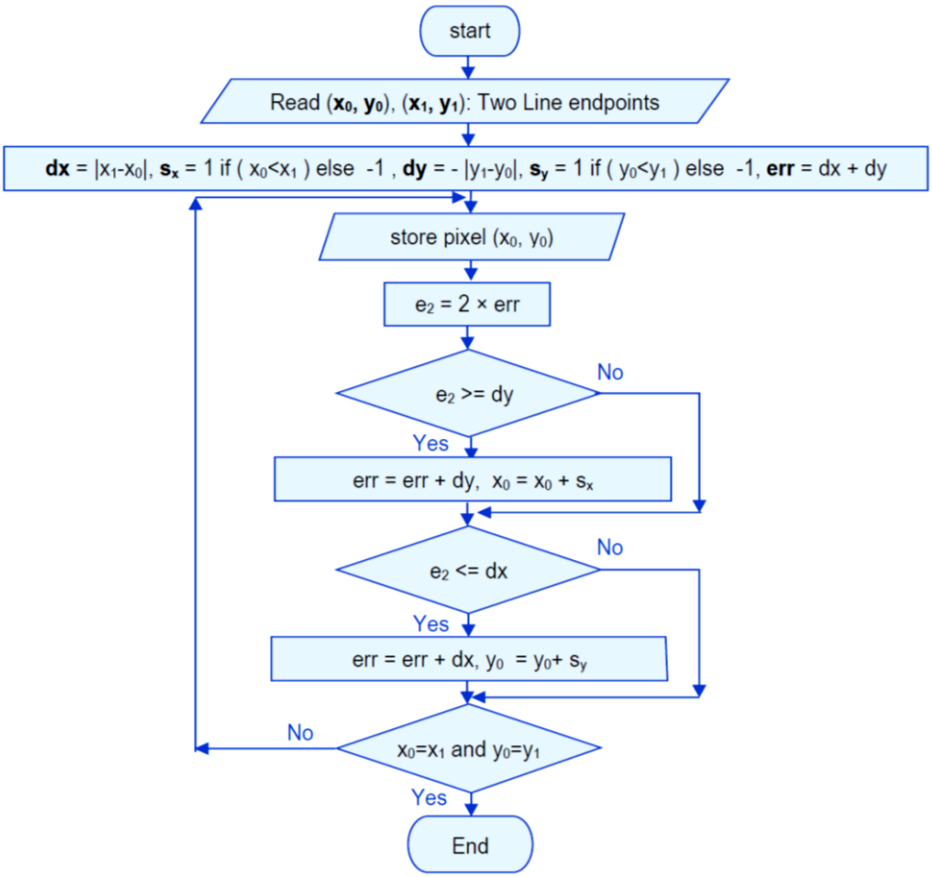


Figure 1. 2D Bresenham's algorithm

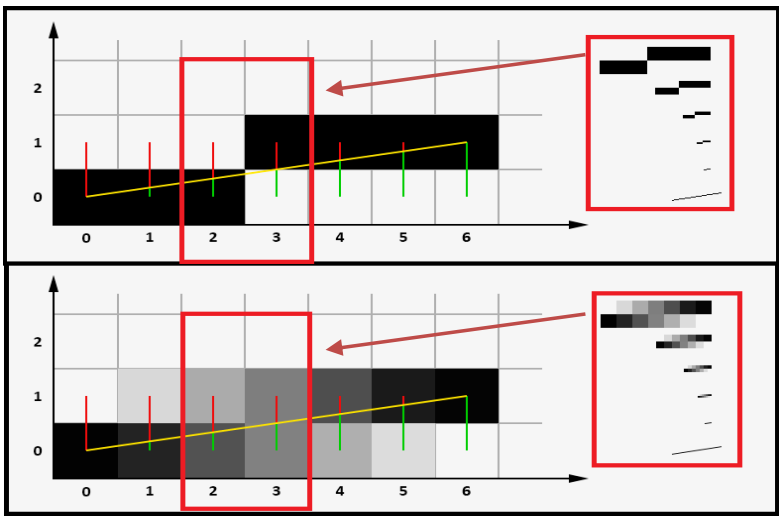


Figure 2. Difference between Bresenham's (upper) and Wu's

Table 1. Wu's Algorithms intensity calculations for Figure 2

x	Upper red distance	Lower green distance	Upper pixel intensity	Lower pixel intensity
0	1	0	100%	0%
1	0.9	0.1	90%	10%
2	0.75	0.25	75%	25%
3	0.5	0.5	50%	50%
4	0.25	0.75	25%	75%
5	0.1	0.9	10%	90%
6	0	1	0%	100%

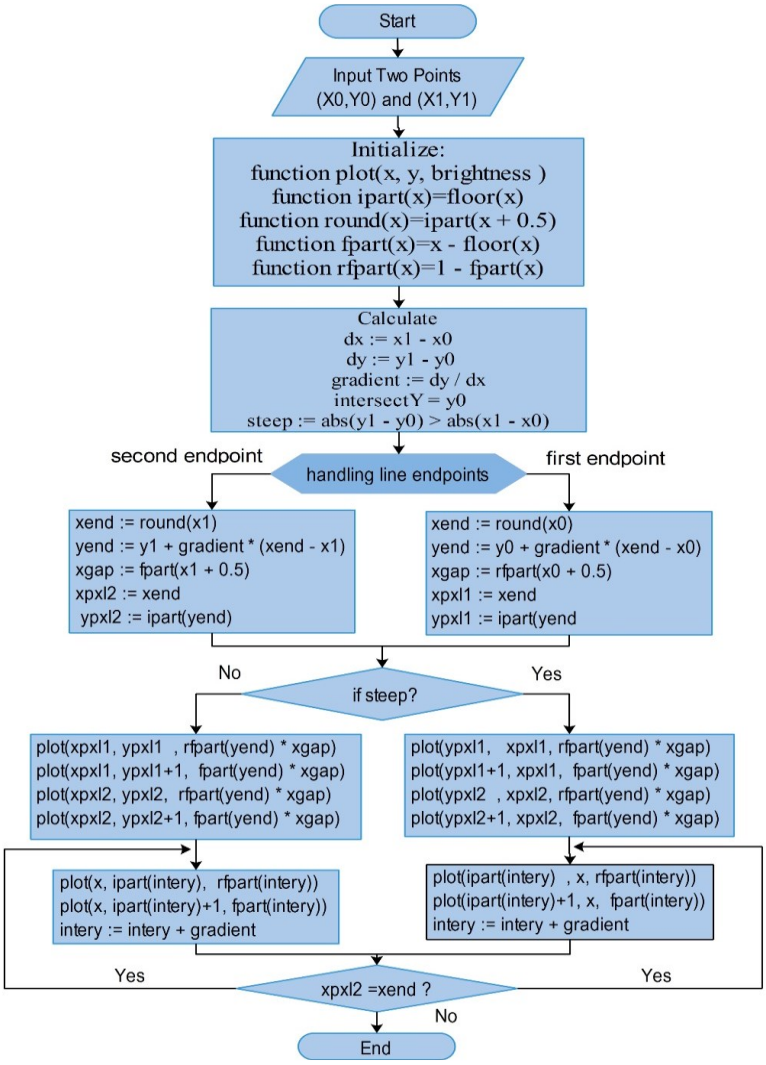


Figure 3. Wu's algorithm

3. THE PROPOSED HARDWARE ACCELERATOR

High level synthesis (HLS) technique has been used in many designs in the past several years [15-17]. Its acceptance remains to grow because it is the fastest way to convert complex algorithms into efficient hardware implementations since the computer graphics algorithms are complex in its nature [18-20]. Therefore, we try to use HLS in our work as mentioned before in Wu's algorithm. The line pixels pairs are generated pair after pair starting from point (x_a, y_a) towards the endpoint (x_b, y_b). Thus, the time necessary to calculate all the pairs is growing as long as the line extended, making the plotting process slow down. Our approach fuses Bresenham's algorithm with Wu's algorithm. First, the Bresenham's algorithm is used to divide the line into equal-length segments and calculates the individual line segments endpoints coordinates. Then, Wu's algorithm is used to anti-alias these line segments simultaneously. Many optimation directives are utilized in order

to enhance the performance of our designed algorithm such as PIPELINE, DATAFLOW, and LOOP_MERGE directives that help the hardware algorithm within the HLS environment to speed up the execution time.

3.1. Implementation using single-core

The ZC702 board which is populated with the Zynq-7000 XC7Z020 AP SoC has been used in implementing the proposed design in this work as a hardware platform. It consists of an SoC-style combined processing system (PS) and programmable logic (PL) on a single chip [21]. Figures 4 and 5 illustrate the proposed design for Wu's algorithm only using this platform and its relative flowchart is shown in Figure 6. The Zynq AXI_lite interface connection is used to join the hardware core of Wu's algorithm with other hardware elements in the FPFA environment such as shared memory block and processing system (Figures 4 and 5). The shared memory in our design is used to keep the initial line endpoints and then, it receives all coordinates of the line segments that are generated by the hardware cores of our designed algorithm in order to send them to the PC via serial port (Figure 6).

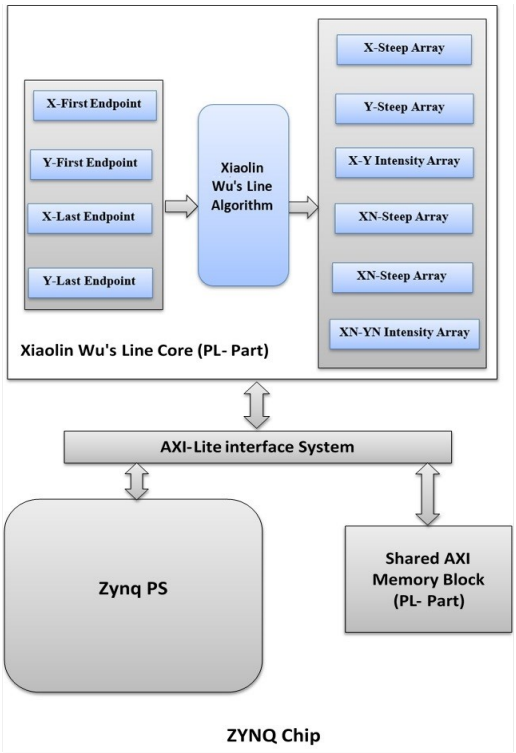


Figure 4. Proposed Wu's algorithm on Zyng chip

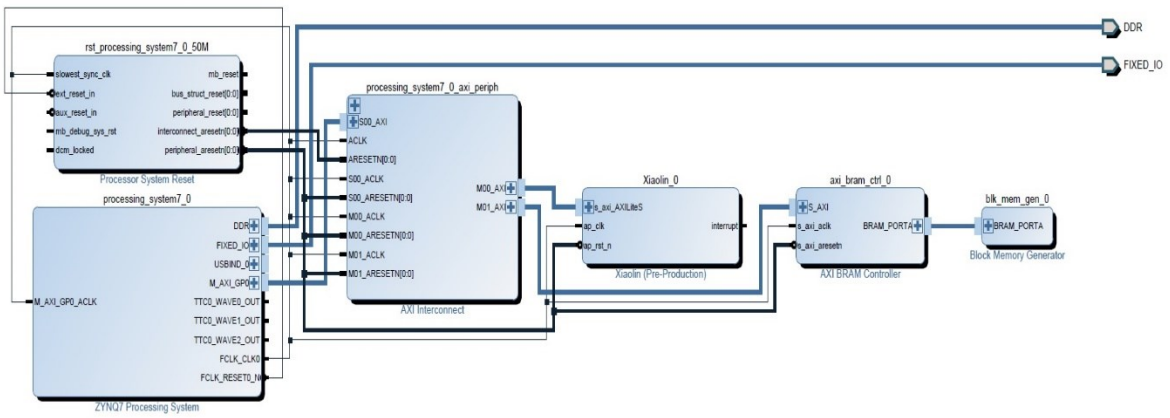


Figure 5. Implementation of Wu's algorithm on Zyng chip

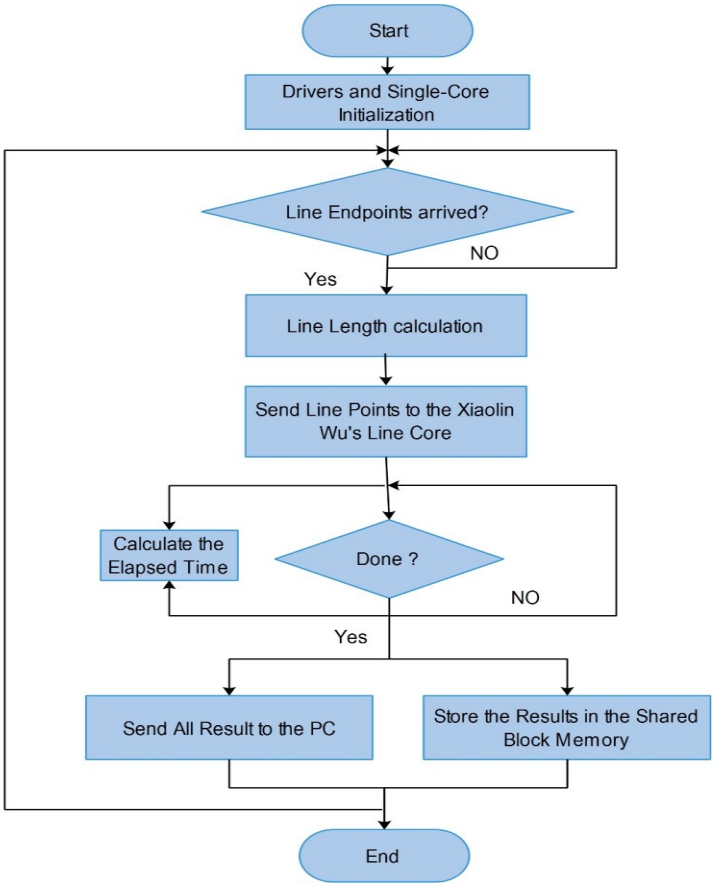


Figure 6. Single core flowchart

3.2. Implementation using multi-core

The used Zynq hardware platform in our design comprises a processing system (PS) organized around a dual-core ARM Cortex-A9 processor, and programmable logic (PL), that equivalent of traditional FPGA with additional new features such as integrated memory, variety of peripherals, and high-speed communications interfaces [21-23]. Figure 7 and Figure 8 illustrate the overall designed system and its relative flowchart respectively. Bresenham's algorithm for dividing line into pieces is done using the ARM A9 Cortex or the processor system (PS) available on the used Zynq chip followed by directing the calculated endpoints via AXI4-Lite bus, which is simple, easy and does not require memory mapping, to the programmable logic (PL) part located on the same Zynq chip.

Wu's algorithm is done on the FPGA or the programmable logic (PL) which contains up to 10 parallel cores each of those cores can perform an isolated procedure simultaneously with other cores. Therefore, to smooth a line, first, divide it into up to 10 equal-length parts and then, compute the endpoints of each part separately then draw all anti-aliased parts in parallel. As we mentioned in the previous section, each hardware core is associated with special block memory to store the output coordinates of each algorithm (Figure 7). Also, we implement the binary tree algorithm witch is the simple and fast algorithm [24-26] to divide the original line into the equal line segment which is based on the number of the hardware cores (10 cores in our system) in order to calculate the initial endpoints for each segment and then sent each pair of endpoints to specific Wu's hardware algorithm (Figure 8). As a result, the designed cores save the time required to find the total points of the line. In other words, Wu's algorithm is implemented concurrently with a number of times equal to the number of cores involved in the process. To decrease the total time needed to draw the line, the segments' number must be increased.

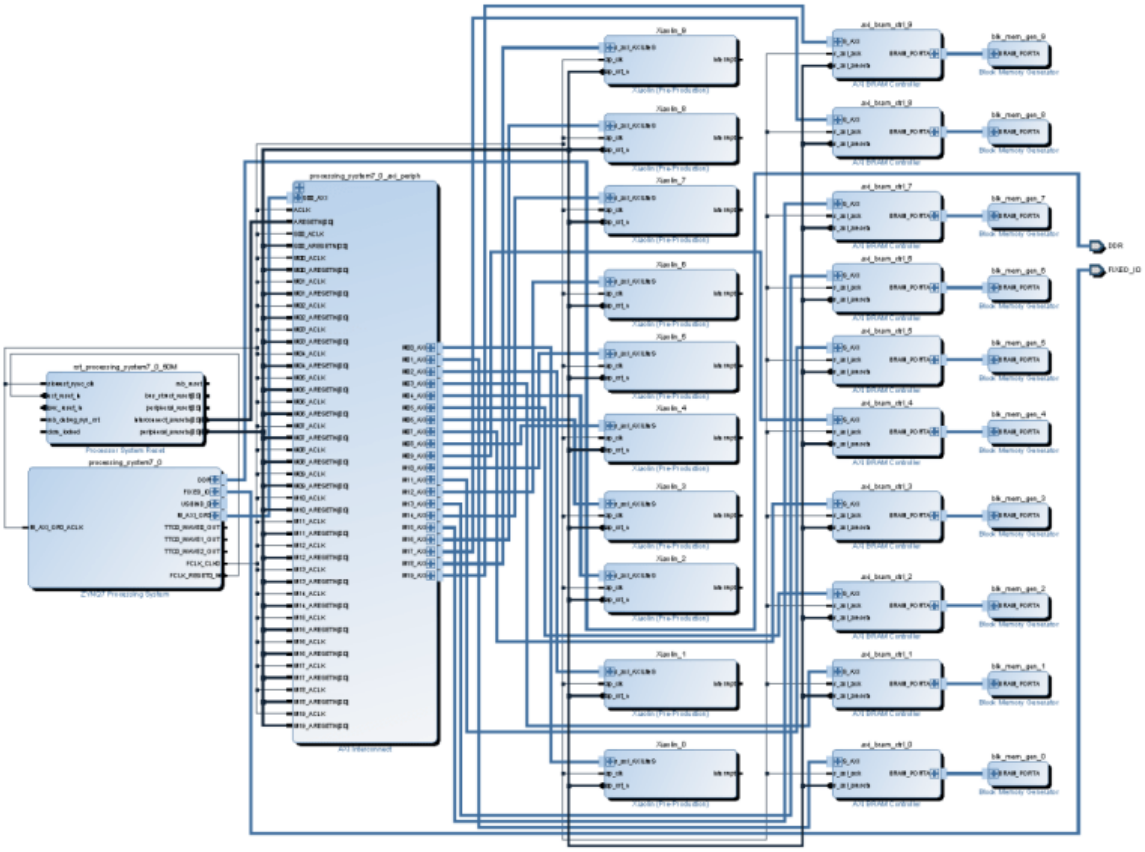


Figure 7. Overall designed system on Zynq chip

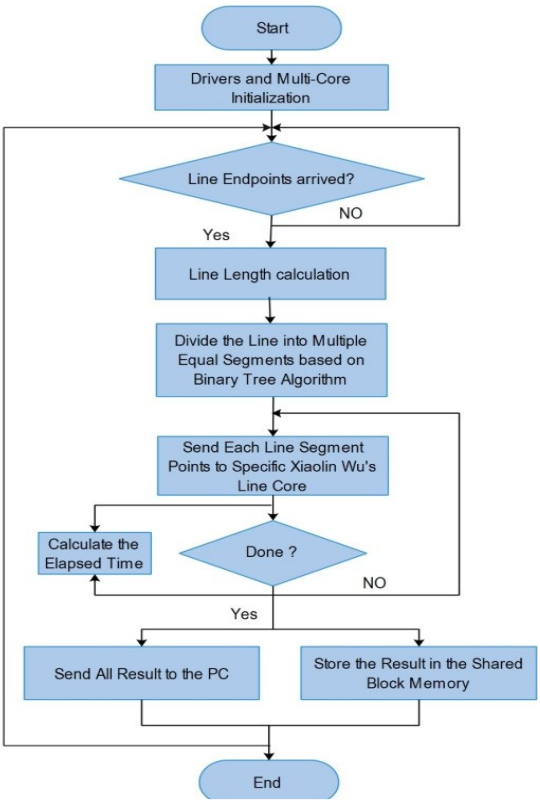
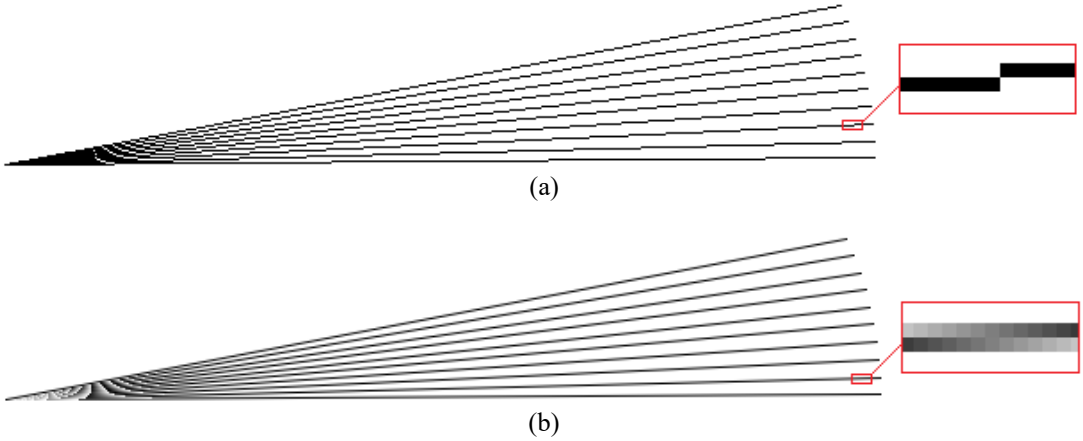


Figure 8. Multi core flowchart

4. RESULTS AND ANALYSIS

In this work, all the design processes are done using the Vivado Design Suite package as well as design optimization, which is necessary to meet timing requests. The Xilinx release of Vivado Design Suit 2016.1 supports Zynq702 with a wide variety of FPGA devices. It replaces the previous design tool by its extra features of high-level synthesis and SoC [22, 23, 27, 28].



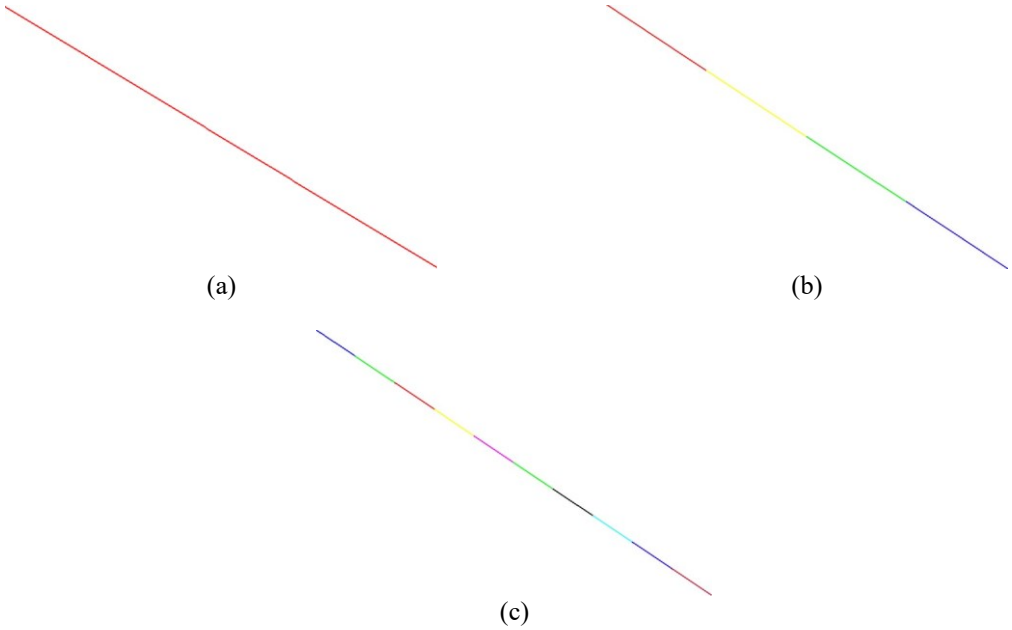
(a)

(b)

Figure 9. (a) Line segments with aliasing and (b) antialiasing

4.1. Graphical analysis

Different slopes same length line segments are drawn using 2D Bresenham's algorithm directly without smoothing as depicted in Figure 9 (a) then, the same lines are drawn with antialiasing using Wu's algorithm as shown in Figure 9 (b). It is obvious that from this figure the efficiency of the Wu's algorithm is omitted the aliasing effect. After that, the number of segments produced from the main line using Bresenham's algorithm to calculate line segments endpoints rapidly depends on the number of FPGA cores used. These cores work concurrently for antialiasing purpose using Wu's algorithm. The generated points from two algorithms are plotted using Matlab as shown in Figure 10. Colors are used to differentiate the start and end of each section. The line pixels computed in (a) 1-core, (b) 4-cores, (c) 10-cores.



(a)

(b)

(c)

Figure 10. (a) Line computed using; 1- core, (b) 4-cores, (c) 10-cores

4.2. Timing analysis

The Xilinx Zynq ZC702 board that used in our design works at 156.25 MHz frequency [24]. The cores are used to share the calculation of antialiased line pixels. The hardware runtime is decreased to half when the number of cores used in the design is doubled. The best time accomplished is 0.31µs when 10 cores are involved. This time represents the splitting up time (in PS) in addition to the time necessary for antialiased pixels calculation (in PL). Figure 11 reflects the increase in the number of cores used against the decrease in the hardware running time.

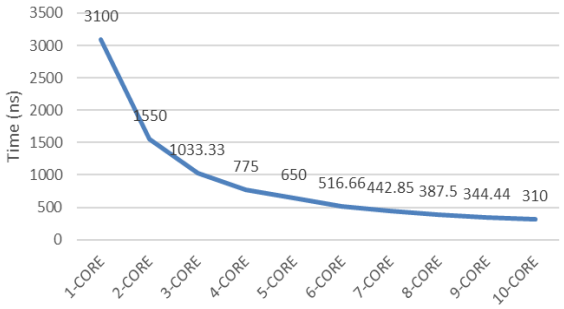


Figure 11. Inverse relationship between number of cores user and hardware time

4.3. Power consumption analysis

This section demonstrates power consumption by each core during the implementation of the designed hardware accelerator on Zynq. The power consumption between the different number of cores used in our design has been compared as shown in Figure 12. It is clear that the power which is consumed by PS (95%) is larger than that is consumed by PL (5%) using one core (total 1.6 W). This difference in the distribution of the consumed power has been shrunken as the number of cores increases to be (69%) in PS and (31%) in PL when using ten cores (total 2 W).

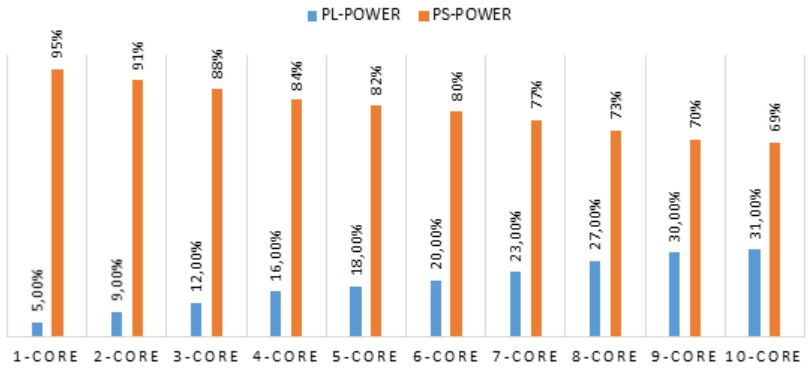


Figure 12. Power consumption: increasing number of cores used against consumed power distribution in both PL and PS Zynq parts

4.3. Resource evaluation

Each designed FPGA cores in the Zynq702 kit contains multiplicity of logic resources which is important to build the allocated digital circuit by the user, like look-up tables (LUT), block RAM and flip-flops [28]. The more cores are used, the more utilization of resources is resulted. The high capability of the Zynq702 platform and the Vivado design suite software package lead to excellent accomplishment in executing time. Conversely, the percentages of used resources that are increased directly with the greater number of cores used as shown in Figure 13.

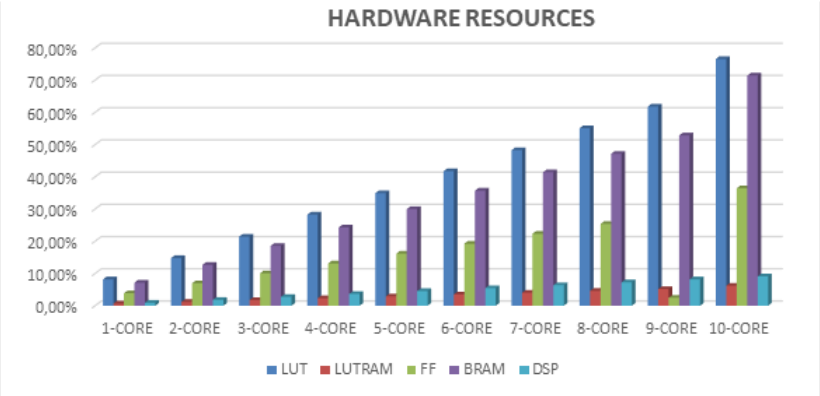


Figure 13. Hardware resources: increasing number of cores used against increasing in the percentage's utilization resources

5. CONCLUSION

Since the multi-core parallel system is used to draw anti-aliased lines, the larger number of segments, the faster the smoother pixels are calculated. This parallel usage of these similar cores leads to a very fast execution of Wu's algorithm (0.13 μs) which is the tenth hardware runtime for one core only. We have concluded that FPGAs are considered as a valuable platform for studying problems related to multi-core CPUs. Their flexibility allows different designs to be evaluated, and their ability to run full-length programs provides an advantage over software simulators. Partitioning an application between core processor and co-processor can be attractive for the theoretical or manufacturing situation. Resource utilization and power consumption parameter of the proposed technique are calculated and targeted on Zynq evaluation board development kit. It is found that through using single core design utilizes maximum 8% and consumes 1.6 W, while utilized resources using 10 cores is 77% with power consumption of 2 W.

REFERENCES

[1] X. Wu, "An efficient antialiasing technique," *Acm Siggraph Comput. Graph.*, vol. 25, no. 4, pp. 143-152, 1991.
[2] A. Dda and F. H. Ali, "Al-Rafidain Engineering," no. June 2008, pp. 25–33.
[3] J. E. Bresenham, "Algorithm for computer control of a digital plotter," *IBM Syst. J.*, vol. 4, no. 1, pp. 25-30, 1965.
[4] E. Angel and D. Morrison, "Speeding Up Bresenham," *IEEE Computer Graphics and Applications*, no. 6, pp. 16-17, 1991.
[5] B. M. K. Younis and M. R. Ahmed, "3D Stereo Rendering Using FPGA," *Comput. Eng. Intell. Syst.*, vol. 10, no. 2, 2019.
[6] C. Au and T. Woo, "Three dimensional extension of Bresenham's Algorithm with Voronoi diagram," Comput. Des., vol. 43, no. 4, pp. 417–426, 2011.
[7] B. M. K. Younis and S. M. am Sheet, "Hardware Implementation of 3D-Bresenham's Algorithm Using FPGA," *Tikrit J. Eng. Sci.*, vol. 20, no. 2, pp. 37–47, 2013.
[8] K. Panek, B. Flak, S. Koryciak, and K. Wiatr, "Basic 3D graphics processor implemented on small FPGA," *Meas. Autom. Monit.*, vol. 64, 2018.
[9] S. Ismael, T. Omar, and Y. T. Qassim, "Hardware/software co-design for a parallel three-dimensional bresenham's algorithm," *Int. J. Electr. Comput. Eng*., vol. 9, no. 1, p. 148, 2019.
[10] F. Hamid Ali, "Anti-Aliased DDA," *AL-Rafdain Eng. J.,* vol. 17, no. 2, pp. 25-34, 2009.
[11] F. Klawonn, "Introduction to Computer Graphics: Using Java 2D and 3D", *Springer Science & Business Media*, Jan 18, 2012
[12] L. Ammeraal, K. Zhang,"Classic 2D Algorithms", *Computer Graphics for Java Programmers*, pp. 91-136, 10 October 2017
[13] B. M. K. Younis and M. R. Ahmed, "3D Stereo Rendering Using FPGA," *Comput. Eng. Intell. Syst.*, vol. 10, no. 2, 2019.
[14] M. Lambert, T. Miriam, F. Susan, "Xiaolin Wu's Line Algorithm," *Betascript Publishing*, 2010.
[15] Alazzawi,Ahmed Khazal; Ercan, Tuncay, "Field-Programmable Gate Array Implementation of Adaptive Neuro-Fuzzy System Using Sensors Monitoring Health-Care Medicinal Internet of Things," *Journal of Medical Imaging and Health Informatics*, vol. 10, no. 1, pp. 169-177, January 2020.
[16] Ahmed Khazal, Tuncay Ercan, "ANFIS Analysis of Wireless Sensor Data with FPGA," *Acta Infologica*, vol. 2, no. 1, pp. 22-32, 2018.
[17] Tuncay Ercan, Ahmed Khazal Al Azzawi, "Design of an FPGA-based Intelligent Gateway for Industrial IoT," *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 8, no. 1.2, pp. 126-130, 2019.
[18] Xilinx Inc., "Introduction to FPGA Design with Vivado High-Level Synthesis," *UG998*, vol. 1, no. 1, January 22, 2019.

[19] Xilinx Inc., "Vivado Design Suite User Guide High-Level Synthesis," *UG902*, vol. 2017, no. 1, April 5, 2017.
[20] Xilinx Inc., "Vivado HLS Optimization Methodology Guide," *UG1270*, vol. 2018, no. 1, April 4, 2018.
[21] Xilinx Inc., "Zynq-7000 All Programmable SoC Data Sheet Overview," *DS190*, vol. 1, no. 11, June 7, 2017.
[22] L. H. Crockett, R. A. Elliot, M. A. Enderwitz and R. W. Stewart, "The Zynq Book: Embedded Processing with the ARM Cortex-A9 on the Xilinx Zynq-7000 All Programmable SoC," *First Edition, Strathclyde Academic Media,* 2014.
[23] Xilinx Inc., "ZC702 Board User Guide," *UG850*, vol. 1, no. 5, September 4, 2015.
[24] I. Grout, "Digital Systems Design with FPGAs and CPLDs," *Google Sch. Digit. Libr.,* pp. 123-176,2008.
[25] F. Fleuret, "Fast binary feature selection with conditional mutual information," *Journal of Machine learning research*, vol. 5, pp.1531-1555, 2004.
[26] Elaidi, Halima, Y. Elhaddar, Z. Benabbou, and H. Abbar. "An idea of a clustering algorithm using support vector machines based on binary decision tree," *International Conference on Intelligent Systems and Computer Vision (ISCV)*, 2018, pp. 1-5.
[27] Xilinx Inc., "Vivado Design Suite User Guide Programming and Debugging," *UG908*, vol. 2016, no. 4, November 30, 2016.
[28] Xilinx Inc., "Vivado Design Suite Tutorial Embedded Processor Hardware Design," *UG940*, vol. 2016, no. 3, October 19, 2016.

BIOGRAPHIES OF AUTHORS

Basma MohammedKamal Younis works at Department of Computer Technology Engineering, Northern Technical University, Mosul, Iraq. She received B.Sc. Degree in Electrical Engineering, Electronic and Communication Engineering from University of Mosul, Iraq in 1995, M.Sc. degree in Computer Engineering from University of Mosul, Iraq in 2000 and Ph.D. degree in Computer Engineering, Microprocessors Architecture from University of Mosul, Iraq in 2007. Her research interests and activity are Embedded System Design using FPGA and Computer Graphics. Now, she teaches Advanced Digital Electronics, Microprocessors Based Systems and Embedded Systems for postgraduate and undergraduate students.

Ahmed Khazal Younis works at Department of Computer Technology Engineering, Engineering Technical College, Northern Technical University, Mosul, Iraq. He Completed his undergraduate education (1998-2002), master degree (2005-208) at Computer Technical Engineering Department in Technical College, North Technical University and Ph.D. degree in Computer Engineering, Embedded System design from Yasar University in Izmir, Turkey in 2019. His research interests include: Neural networks, Neuro-fuzzy system, Microcontrollers, and FPGA devices.