ICIC Express Letters
Part B: Applications
Volume **10**, Number **11**, November **2019**

ICIC International ©2019 ISSN 2185-2766
pp. **953–961**

IMPLEMENTATION OF INTEGRAL IMAGE USING ASIC DESIGN METHOD WITH 0.35 μm CMOS TECHNOLOGY

Robby Kurniawan Harahap[1], Eri Prasetyo Wibowo[1,*], Hamzah Afandi[1] and Busono Soerowirdjo[2]

[1]Center for Microelectronics and Image Processing Studies
[2]Department of Electrical Engineering
Gunadarma University
Jalan. Margonda Raya, No. 100, Kota Depok 16424, Indonesia
{ robby_kurniawan; hamzah; busono }@staff.gunadarma.ac.id
*Corresponding author: eri@staff.gunadarma.ac.id

Received May 2019; accepted August 2019

Abstract. *The implementation of the Viola-Jones object detection algorithm in the field of image processing is one of the interesting fields to be studied in the hardware development. One part of the Viola-Jones algorithm that has become interesting to research is the integral image. The function of the integral image is to accelerate the features finding in the feature extraction process. The purpose of this research is to implement an integral image in the existing research using FPGA design method to become Single On-Chip (SOC) using the ASIC design method. The method used was a direct approach for implementing the integral image by bypassing the design steps from HDL to the ASIC block without first doing the FPGA design. This study proposed an architecture implementation of the integral image using an array and an image of a small resolution size. The implementation of the ASIC design method consists of several steps, i.e., specifications, behavioral model, synthesis, design for test, and physical model. The results of this study are an integral schematic image that has been carried out by the DFT process resulting in 100% test coverage and 97.41% fault coverage for the area optimization. This implementation used a base standard-cell 0.35 μm of CMOS Technology.*
Keywords: ASIC, FPGA, Integral image, Implementation, Standard-cell, Test coverage, Viola-Jones

1. **Introduction.** Viola-Jones algorithm, proposed by Paul Viola and Michael Jones [1], is one of the most popular real-time object detection algorithms with high accuracy [2]. This algorithm is widely used for face detection. There are three parts of Viola-Jones algorithm structure, i.e., integral image, feature extraction of Haar and classifier. Of the three parts, the main part is the feature extraction of Haar basis feature. To support finding Haar features, the computation in the previous part is needed, i.e., an integral image. According to Zhang et al.'s studies [3], the integral image has 50% of the total execution time, so that the rapid computation of the integral image is needed.

There are several studies related to developing the rapid computation of the integral image. In the hardware implementation, the faster computation must be in a parallel process. Ehsan et al.'s studies in [4], developed the integral image computation, namely multi-row process. The multi-row method consists of two, i.e., two-row and four-row processes. They have also been implemented in FPGA [5]. Zhang et al.'s studies in [3], developed the computation with a combination of rows first and columns processes. The implementation of computation also used FPGA. Hoseini et al.'s studies in [6], developed the integral image block based on sub-block.

DOI: 10.24507/icicelb.10.11.953

Based on the previous several integral image studies, all of the integral images have been developed in the computational processes on FPGA. The advantages of ASIC technology is the size device and capable to design with custom model compared with FPGA [7]. We then think that the opportunity to be able to study is to develop the implementation target. In the majority of the previous studies, the implementation was done on FPGA, as Ehsan and Zhang did. Hoseini et al.'s works did not explain whether the implementation design was done on FPGA or ASIC. In this paper we propose an implementation of the integral image that has already existed, i.e., the two-row process in FPGA by Ehsan et al. into Single On-Chip using ASIC design [8]. The implementation is using ASIC design methodology for integral image.

Based on Abid and Izeboudjen's studies in [9], it is described that the design method used is the hybrid approach design. The hybrid is the implementation in FPGA first as a model and transferred to ASIC. We think the hybrid in Figure 1 is the complex steps with FPGA first and then ASIC; so with the bypass-step process directly to ASIC it can simplify the steps. For the ASIC design methodology, we refer to Bhatti et al.'s studies in [10] in a curriculum based on the CAD software design. To make it different from Ehsan et al.'s study, our discussion focused on the implementation method for the architecture with an array and used a small image with five different resolution sizes to see the performance. The measurement consists of execution time, utilization of optimum area and delay from synthesis, and coverage test in DFT. The implementation technology uses standard-cell with 0.35 µm CMOS technology.

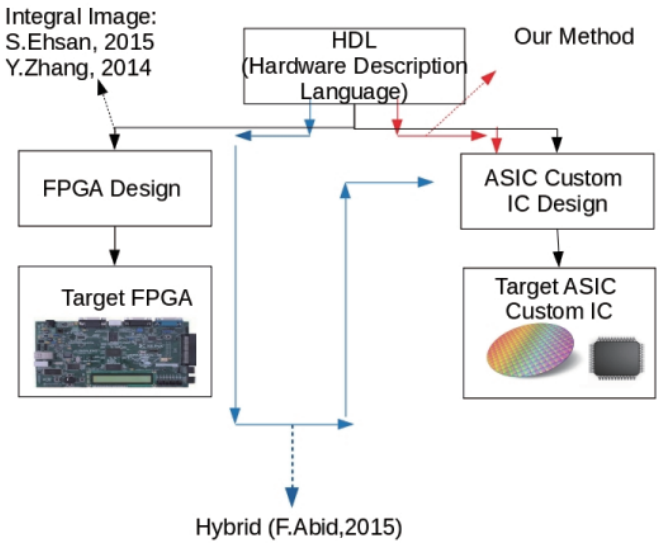

FIGURE 1. Hybrid approach and our method

The organization of the paper is as follows. Basic of integral image and ASIC design methodology as related works are introduced in Section 2. The process to implementation of proposed integral image architechture with ASIC design is in Section 3. The result of implementation is in Section 4. Finally, conclusions are given in Section 5.

2. Related Work.

2.1. Integral image.

The integral image is known as the summed-area table, the word comes from Crow in [11]. Integral image according to Viola and Jones [1] based on Crow is a computational technique in the form of a simple addition to images $i(x, y)$ by adding up all values on the left, top, and itself as shown Figure 2. There are several algorithms using the integral image, two of them, i.e., Viola-Jones and Speed Up Robust Features.

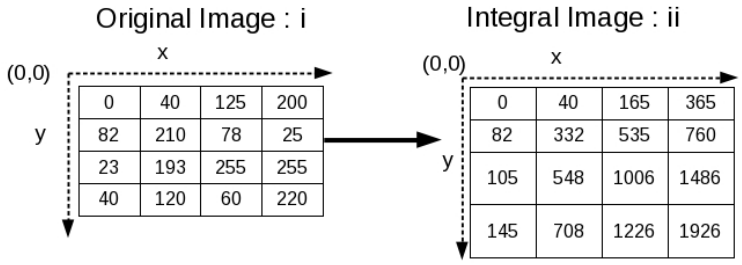

FIGURE 2. Integral image representation

The equation of the integral image for image $M \times N$ size can be computed by using basic Equation (1), where $ii(x, y)$ is an integral image.

$$ii(x, y) = \sum_{x' \leq x, y' \leq y}^{y} i(x', y') \tag{1}$$

In the Viola-Jones studies [1], the computation can be used for small images with scalar $\frac{1}{4}M^2N^2$ in the software implementation. For example, 4×4 can produce 320 addition operation. The problem may occur if the image has a large size resolution. For example, the image 100×100, using Equation (1) is unfit for the hardware implementation, as the equation will generate 250,002,500 addition operations. To solve that problem, the parallel processing is needed by using Equations (2) and (3) [5]. If image size of $M \times N$ is 100×100, then generated 20,000 addition operations. It means using Equations (2) and (3) is reduced by 10^4 times compared with Equation (1).

$$S(x, y) = i(x, y) + S(x, y - 1) \tag{2}$$

$$ii(x, y) = ii(x - 1, y) + S(x, y) \tag{3}$$

Although Equations (2) and (3) can reduce the operation, too large addition operation will be generated. Then by two-row parallel process it can reduce the addition operation and increase the speed of computation [5]. This two-row parallel process refers to Equations (2) and (3) then produce Equations (4) until (7). The parallel process is executed in one clock cycle. One clock cycle means that at one time there are two pixel values in the first and the second rows which are processed and the integral value is obtained. This equation can reduce the additions with scalar $\frac{MN}{2}$. For example, a small image of size 4×4 can produce 8 addition operations, while for large image of size 100×100, it can then produce 5000 addition operations. This equation can reduce 4 times compared with Equations (2) and (3).

$$S(x, y) = i(x, y) + S(x, y - 1) \tag{4}$$

$$S(x + 1, y) = i(x + 1, y) + S(x + 1, y - 1) \tag{5}$$

$$ii(x, y) = ii(x - 1, y) + S(x, y) \tag{6}$$

$$ii(x + 1, y) = ii(x - 1, y) + S(x, y) + S(x + 1, y) \tag{7}$$

2.2. **ASIC design methodology.** There are several previous studies related to the design methods. First is the design OpenRISC or processor for Voice over IP (VoIP) using hybrid approach design with 0.18 μm CMOS technology [9]. In those studies the hybrid approach design is a method for design by using FPGA as a prototype test, then the result will be used to create an IC using ASIC Custom design or, in other words, created on FPGA first, then transferred to ASIC design. Mixture of FPGA and ASIC becomes complexity of the design steps [9]. By simplifying the implementation steps after behavioral model to ASIC design, it will shorten the work time compared with FPGA.

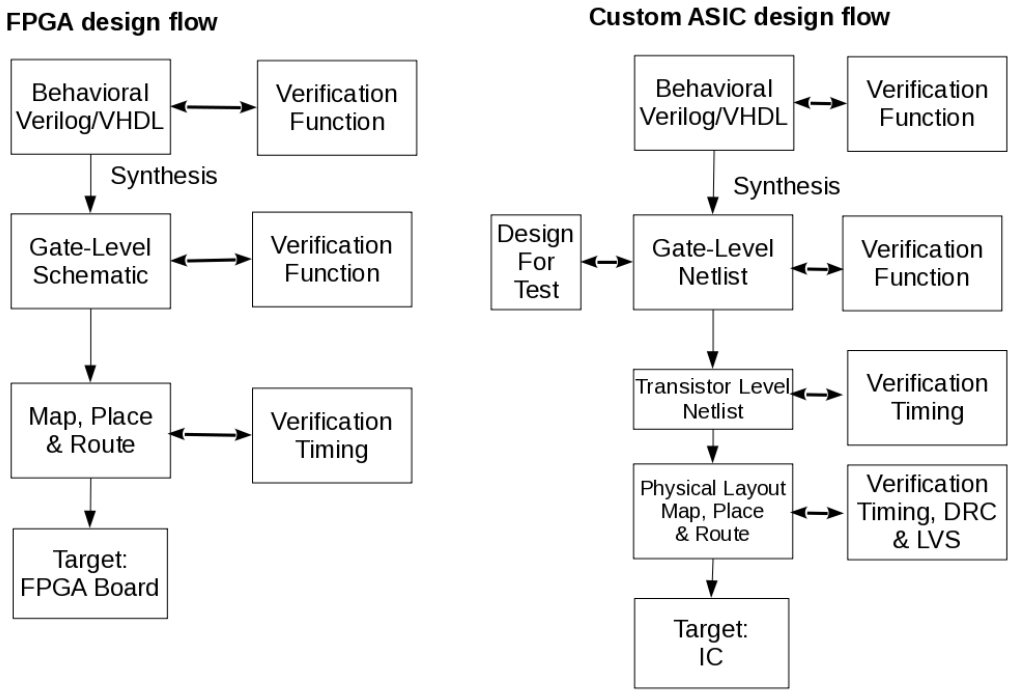

FIGURE 3. Comparation of FPGA and ASIC custom flow

The study using directly the ASIC design method without FPGA first is also carried out on the design of ADC and DAC with 65 nm CMOS Technology [13]. The implementation of the ASIC method used standard-cell. The design flow used was based on CAD tools described in Bhatti et al.'s studies [10] in a curriculum chip design using CAD tools.

3. **Design Method.** The integral image which was implemented in this paper used a two-row process. Our proposed design method is based on the ASIC design methodology in [10] to implement this two-row integral image in [5]. There are several steps of the proposed design method, i.e., specification, behavioral model, synthesis, DFT and physical model. The following is an explanation of each step.

3.1. **Specification.** In this step, we create the specifications of the integral image of a two-row process [4] then test it using a software and determine the CAD software to be used. The input is limited by small size resolutions in terms of square size. There are five sizes resolution, i.e., 2×2, 4×4, 6×6, 8×8, and 10×10. This limited size adjusted to the memory needs to be used. The architecture of the integral image is shown in Figure 5. This architecture consists of 3 blocks, i.e., array, processor and memory. The array block is a location for the arrangement of the input image pixel value. The arrangement is arranged according to original image coordinates $i(x, y)$. In Figure 4, process A is an array process for an image size of more than 10×10. For example, an image size 100×100, with process A in Figure 4 divided into each 10×10 with looping process will then need 10 times looping processes.

According to Figure 4, this paper works only for the implementation of B with a small image to see the performance at a small size. The process of B starts from $(x, y) = (0, 0)$; for example, 2×2 is the first step for $(0, 0)$ and $(1, 0)$, then the second step $(0, 1)$ and $(1, 1)$ which refers to the result of the first step value and move by a column in one step movement or one column $x, y + 1$. For 4×4 until 10×10 there is a line shift $x + 2, y$ when the value in the previous row has been completed. This process integral image is named two-row because $x + 2, y$ two-row works at the same time.

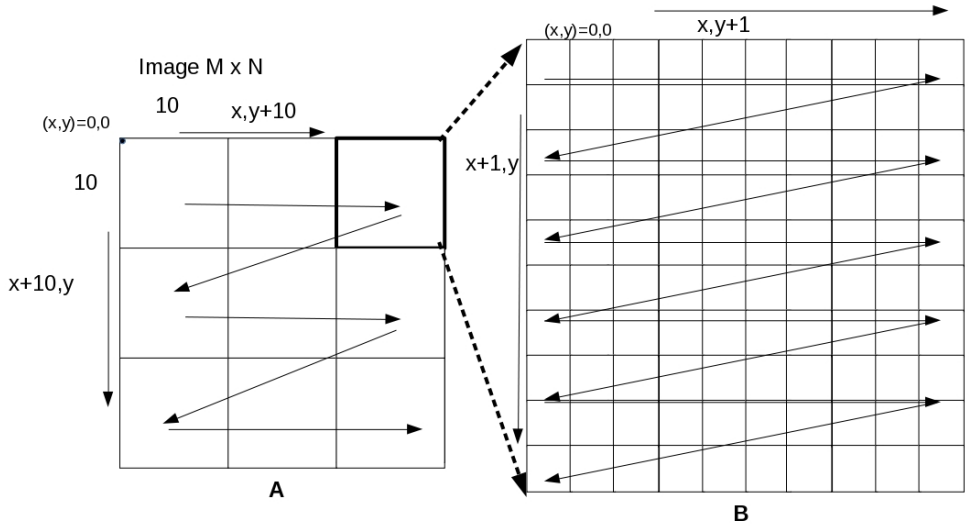

FIGURE 4. (A) For image size $> 10 \times 10$ and (B) for image size $< 10 \times 10$

TABLE 1. Specification integral image for each size

Image size	Input bit	Output bit	Maximum pixel value
2×2	8	10	1020
4×4	8	12	4080
6×6	8	14	9180
8×8	8	15	16320
10×10	8	15	25500

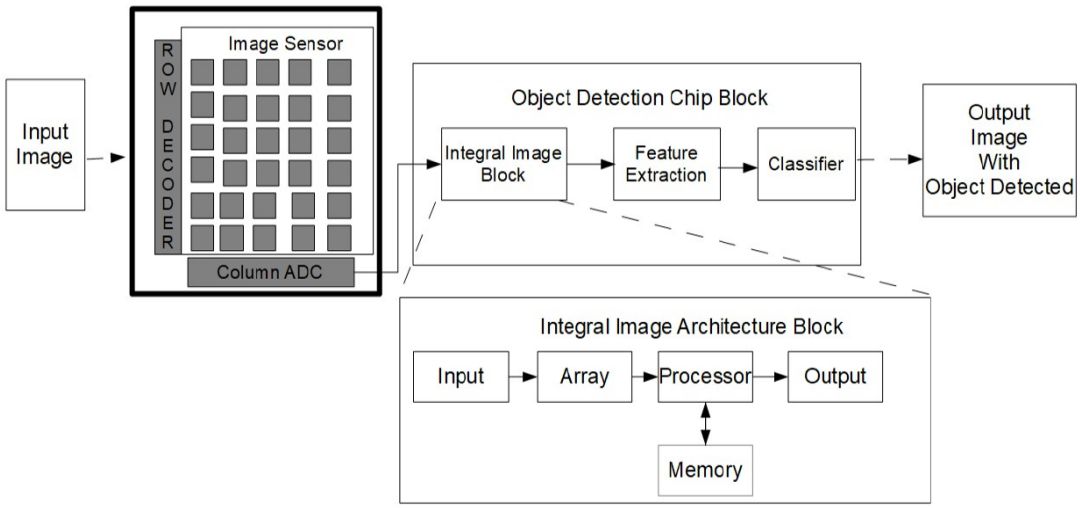

FIGURE 5. Proposed architecture

The input image of $M \times N$ size resolution is assumed to be in gray format with 8 bit (0-255) and the original image value is obtained by converting first. The requirement for the maximum memory size for each size of the integral image output 2×2 with a maximum pixel value of 255 is 10 bits: $ii(0,0) = 510$, $ii(0,1) = 1020$, $ii(1,0) = 510$, and $ii(1,1) = 1020$. The same thing applies to other sizes: 12 bits for 4×4, 14 bits for 6×6, 15 bits for 8×8 and 10×10. The maximum size of memory is shown in Table 1.

Then the equation of two-row process is tested first. The test is done by using octave GNU software to see that the functions are in accordance with the specifications. The test in this software will become a golden model. The software CAD tool used is mentor

graphics for Higher Education Program edition following the design flow in [10]. The architecture in Figure 5 shows our main research. The main research is the object detection chip with Viola-Jones algorithm. This integral image implementation is on a large rectangular box.

3.2. **Behavioral model.** In this step, programming code is using VHDL. Referring to the architecture on Figure 5, the VHDL is created for each block in the integral image. In the input block there are six ports, i.e., clock, reset, input 1 as the input signal of first row, input 2 as the input signal of second row, height as M height size of the image, and width as N width size of the image. In the output block there are five ports, i.e., out 1 as the integral image of first row, out 2 as the integral image of second row, col as the indicator column movement, height, and width. The signal input starts from the input block then forwarded to the array block. The signal input-process into the array is executed in one clock cycle. The array block transfers the value to the processor unit when all the input signals have been completed. The process in the processor works for the integral image Equations (4) until (7). Every value from the addition operation will be stored in the memory block. For the output, the results of Equations (6) and (7) in processor transferred to the output block.

3.3. **Synthesis.** In the synthesis process, the VHDL code is read and 0.35 µm CMOS technology is loaded. The clock constraint used is 10 MHz. This synthesis process parameter uses the area and the delay optimization model then becomes the result of the synthesis of the netlist.

3.4. **Design for test.** Before creating a physical model, the netlist must be tested with DFT. The DFT process uses a scan insertion and Automatic Test Pattern Generation (ATPG) to give results through the ratio of test coverage and fault coverage. In study [12], the good test of DFT is the fault coverage resulting in more than 95%.

3.5. **Physical model.** The netlist of DFT is converted into a schematic representation. The technique uses a semi automatic standard-cell with 0.35 µm CMOS technology.

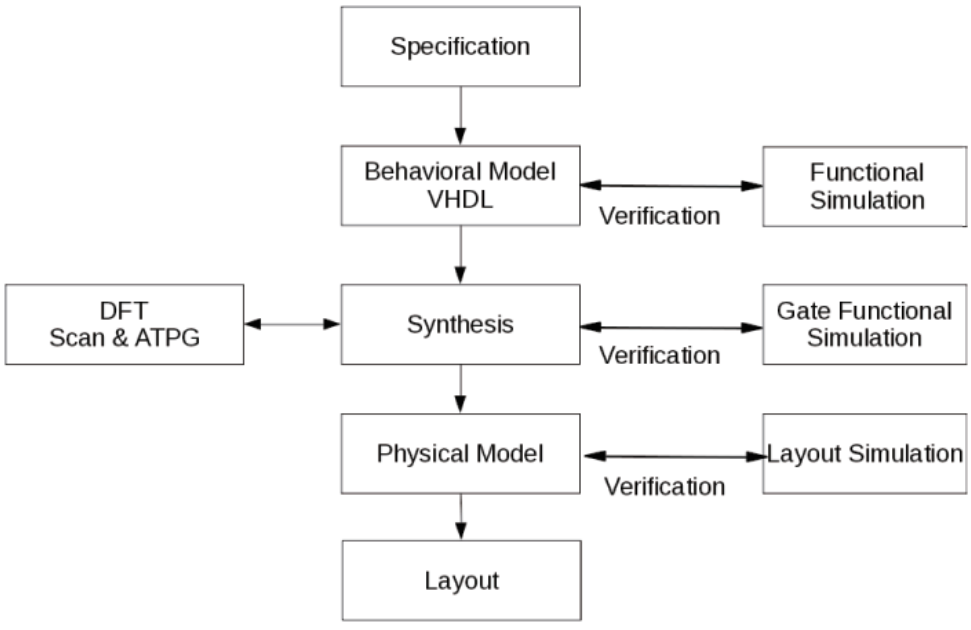

FIGURE 6. Proposed design method flow

4. Result.

4.1. Software implementation.

The implementation using Octave GNU for size 10×10 is shown in Figure 7. According to the original equation in [1, 5], Figure 7 process gives a result of one value at one time with the change column until the last value on the row, after that, change to the next row. For the two-row process, it is the same process as the original, but when the result in the first row is obtained at the same time, the result of the second row is also obtained. For the change of rows, it is the next 2 two rows.

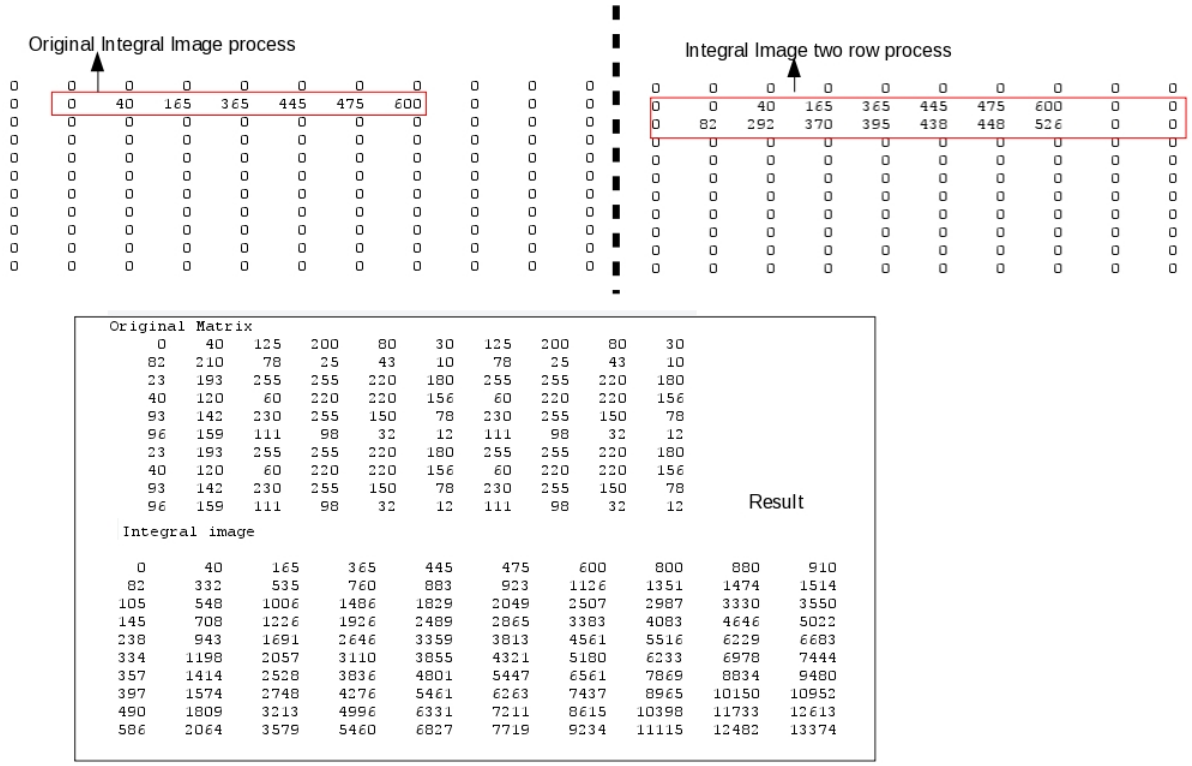

FIGURE 7. Software implementation of original and two row integral image process

4.2. Simulation behavioral model.

Each size of the five image sizes is simulated with the same data. The measurement of the execution time or runtime of each size is listed in Table 2. The execution time is measured by the time difference between the first input time and the last output time. The execution time representation is $\Delta t = tf - ts$ nanosecond. In Table 2 the image size 2×2 starts from the first input signal value of 0 and 82 at $ts = 0$ ns and stops the last output signal of 40 and 332 at $tf = 900$ ns; the result for each size uses one clock cycle in 100 ns.

TABLE 2. Execution time

Image size	Execution time (ns)
2×2	900
4×4	2100
6×6	4100
8×8	6900
10×10	10500

4.3. **Synthesis result.** The measurement of the total area is listed in Table 3. The area optimization has given the result for the total gate being lower than the delay optimization. This means that the area optimization has reduced the gate to get the optimum area. The total area is related to power consumption. Even though the software we use at this step cannot display the estimated power consumption, the more gates that are used, the more power consumption is needed. The slack time result in the optimization shows that it is related to the speed. The slack time in the delay optimization gives a lower time value than the area optimization. The slack time is measured from the difference between the required time and the actual time.

TABLE 3. Area optimization 2×2, 4×4, 6×6, 8×8, and 10×10 with clock 100 MHz

Image size	Area optimization		Delay optimization	
	Total gate	Slack time	Total gate	Slack time
2×2	168732	1.31	179179	0.77
4×4	191883	1.31	202384	0.77
6×6	320429	1.31	330494	0.77
8×8	466976	1.31	479625	0.77
10×10	603330	1.31	614942	0.77

4.4. **DFT.** The DFT is tested using a scan insertion with a full scan in the form of a scan chain. The test pattern ATPG gives the test coverage of the area optimization an amount of 100% which means that all the areas are covered in the test. The fault coverage in the amount of 97.41% means that there is a device fault amounting to 2.59%. This fault coverage is in the basis of good category [14] at least reaching 90-95%. In the delay optimization test coverage the amount is 99.98%; there is 0.02% area in the design which cannot be tested with 97.37% fault coverage.

4.5. **Physical model.** The physical model is shown in Figure 8. This schematic is based on the optimization area. The technique used is a semi automatic conversion with a standart-cell 0.35 μm CMOS Technology.

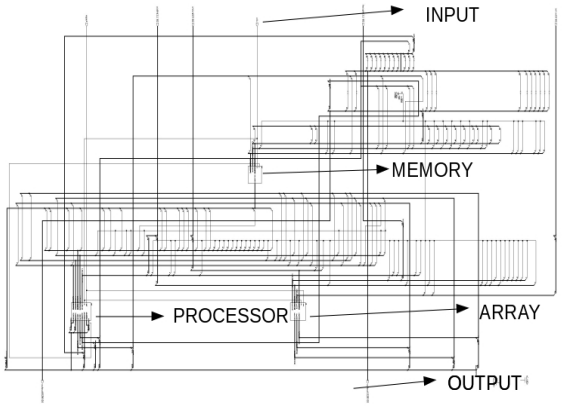

FIGURE 8. Schematic of integral image CHIP

5. **Conclusions.** In this paper, we propose an implementation of the integral image presenting the ASIC design. The bypass or directly implementation to ASIC design can simplify the steps design. This can also be related to the costs of design. For the integral image with 8-bit gray input, there is an increase bit size for the memory which must be considered. Although the size tested is different from the research sources by a limited

small size, the result for the smallest size 2×2 needs 900 ns for the execution time. The area optimization of the synthesis process focuses on the optimum device used and the delay optimization focuses on the slack time or speed. The DFT process shows that this design is in good design with the area optimization because all the areas can be tested with 100% test coverage and there is a device fault which can be covered amounting to 97.41%. This study has not yet reached the schematic simulation step. There are still some steps to be done which can be explored further in the future; and for the real-time application like videos, this method in a small size can be an alternative in the data transfer.

Acknowledgment. This work is partially supported by RISTEK DIKTI, The Ministry of Education and Culture of Republic Indonesia with number 023/KM/PNT/2018 and Gunadarma University for support with scholarship.

REFERENCES

[1] P. Viola and M. J. Jones, Robust real-time face detection, *International Journal of Computer Vision*, vol.57, no.2, pp.137-154, 2004.

[2] E. P. Wibowo, A. S. Talita, M. Iqbal, A. B. Mutiara, C. K. Lu and F. Meriaudeau, *An Improved Calibration Technique for Polarization Images*, IEEE Access, 2019.

[3] Y. Zhang, S. Yin, P. Ouyang, L. Liu and S. Wei, A parallel hardware architecture for fast integral image computing, *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp.2189-2192, 2014.

[4] S. Ehsan, A. F. Clark and K. D. McDonald-Maier, Novel hardware algorithms for row-parallel integral image calculation, *Digital Image Computing: Techniques and Applications (DICTA'09)*, pp.61-65, 2009.

[5] S. Ehsan, A. F. Clark, K. D. McDonald-Maier et al., Integral images: Efficient algorithms for their computation and storage in resource-constrained embedded vision systems, *Sensors*, vol.15, no.7, pp.16804-16830, 2015.

[6] Y. D. Hoseini, S. M. Sayedi and S. Sadri, A novel CMOS image sensor for high speed parallel integral image computation, *The 21st Iranian Conference on Electrical Engineering (ICEE)*, pp.1-6, 2013.

[7] V. E. Kristianti, E. P. Wibowo, A. Pertiwi, B. Soerowirdjo and H. Afandi, Implementation optimization of the DES algorithm on FPGA to support smartcard processors, *ICIC Express Letters, Part B: Applications*, vol.10, no.7, pp.565-570, 2019.

[8] R. K. Harahap, E. P. Wibowo, H. Afandi and B. Soerowirdjo, Redesign integral image on ASIC HDL level, *Proc. of Regional Conference on Computer and Information Engineering (RCCIE 2018)*, 2018.

[9] F. Abid and N. Izeboudjen, ASIC implementation of an OpenRISC based SoC for VoIP application, *The 6th International Conference on Information and Communication Systems (ICICS)*, pp.64-67, 2015.

[10] M. K. Bhatti, A. A. Minhas, M. N. Islam, M. A. Bhatti, Z. U. Haque and S. A. Khan, Curriculum design using mentor graphics higher education program (HEP) for ASIC designing from synthesizable HDL to GDSII, *IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE)*, Hongkong, China, 2012.

[11] F. C. Crow, Summed-area tables for texture mapping, *ACM SIGGRAPH Computer Graphics*, vol.18, no.3, pp.207-212, 1984.

[12] H. Bhatnagar, *Advanced ASIC Chip Synthesis: Using Synopsys® Design Compiler™ Physical Compiler™ and PrimeTime®*, Springer Science & Business Media, 2007.

[13] V. Unnikrishnan and M. Vesterbacka, Mixed-signal design using digital CAD, *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pp.6-11, 2016.

[14] L. Xiu, *VLSI Circuit Design Methodology Demystified: A Conceptual Taxonomy*, John Wiley & Sons, 2007.