VLSI IMPLEMENTATION OF IMAGE THRESHOLDING ALGORITHMS

Dr. R. Manjith

Associate. Professor,
Department of Electronics and Communication Engineering,
Dr. Sivanthi Aditanar College of Engineering,
Tiruchendur.

Abstract - *In document image processing, binarization plays a significant part especially in degraded document images. Among various thresholding algorithms, local image thresholding algorithm plays a better binarization performance for degraded document images. However, this algorithm is computationally sensitive and intensive to the noises from internal computational circuits. The stochastic logic is a process which operates on probabilistic signals and it can cope with errors and uncertainty. The stochastic computing performs computations with conventional digital logic gates by using streams of random bits. This computing is applied to a reconfigurable architecture that implements processing operations which can be used for the image processing application. The fault tolerant architecture is a reconfigurable architecture as it can compute different function by setting an approximate value to constant register, based on stochastic logic. In this paper, Sauvola and niblack local thresholding algorithms using stochastic computing are analyzed and compared for various performances such as hardware resource utilization, area and threshold value. Simulation results show the superior performance of sauvola algorithm for hardware resource utilization, area and threshold value than niblack algorithm.*

***Key Words*:** Sauvola, Niblack, Threshold, Stochastic.

1. INTRODUCTION

Stochastic computing (SC) is a (re-)emerging computing technique that was invented for low-cost alternative to conventional binary computing. It is unique in that it represents and processes information in the form of probabilities. More recently, it has been shown to be useful in important applications such as image-processing and communications. Document binarization is the first step in optical character recognition (OCR) which has been an active study area in recent years. The binarization can be performed by a method known as thresholding, which selects a threshold value and then process under goes on all pixel intensities above/below this threshold values are set to 1 (background)/ 0 (foreground) or vice versa. Thresholding algorithms are widely classified into global and local methods. In local methods, a threshold value is calculated for each window which is based on local region but in the global method a single threshold values are selected for whole image. Global methods, such as Otsu[1], are often very fast, and give good results for typical scanned documents but does not suits the degraded document images. The global methods tend to produce marginal noise along the page borders and it does not suit due to the changes caused by illumination, complexity in documents and poor quality of documents. Local methods can solve the problem of global method by using local information. Local methods achieve high-quality results on degraded documents and historical documents [5]. Local method leads to more timing than global method as the threshold is calculated for each window. The local methods are applied only to a gray scale images. This technique is sensitive to background noises due to large variance in case of a poor illuminated document. Sauvola method can somehow tolerate noises in the images due to snow, rain, or camera shaking as it deals with local pixel values. However, it is still responsive to the noises from the internal circuits, such as the noises due to soft errors, environmental noises, or process, voltage, and thermal variations. Similar to the other local thresholding methods, processing time of the Sauvola thresholding is usually much long [14].

The conventional fault-tolerance techniques, such as triple modular redundancy (TMR) [7][13], can increase the fault-tolerance ability but the major disadvantages are increase the number of hardware resources and higher power consumption. In order to solve the problem in the conventional method like high computation time, increase the number of hardware resources and higher power consumption stochastic computing (SC) is used [11]. SC has the advantage of low cost, can tolerate a very large number of errors, lower hardware resources and easy conversion of complex operation into the simple operation. Since all bits have the same implication in stochastic representation, a single bit flip in a long bit stream will result in a small change in the value of the stochastic numbers than the conventional methods. And so, stochastic circuits are naturally more fault-tolerant. In this paper, a high-speed, low-power and fault-tolerant stochastic architecture for Sauvola algorithm and Niblack algorithm using SC are designed and comparison are made between them. To implement sauvola and niblack algorithm using stochastic computing, we use a stochastic mean circuit (SMC), a stochastic standard deviation circuit, stochastic square root circuit, an XOR gate, and an AND gate. Section II introduces the background of stochastic logic. Section III presents stochastic implementations of Sauvola and Niblack algorithm. Finally, the simulation results are shown in section IV.

2. BACKGROUND

STOCHASTIC LOGIC

A basic feature of SC is that numbers are represented by bit-streams that can be processed by very simple circuits, while the numbers themselves are interpreted

as probabilities under both normal and faulty conditions. For example, a bit-stream S containing 25% 1s and 75% 0s denotes the number P = 0.25, reflecting the fact that the probability of observing a 1 at an arbitrary bit position is P. Neither the length nor the structure of S need be fixed; for example, (1,0,0,0), (0,1,0,0), and (0,1,0,0,0,1,0,0) are all possible representations of 0.25. Note that P depends on the ratio of 1s to the length of the bit-stream, not on their positions, which can, in principle, be chosen randomly. A bit streams of this type and the probabilities they represent as stochastic numbers.

2.1 Stochastic Computing

In SC [14], computations are performed on the Boolean domain which are are transformed into probabilistic real domain[11]. Stochastic numbers are represented by streams of random bits in two formats[8] as unipolar and bipolar. Bipolar format is associated with negative numbers directly. For a given stream length, the precision of the unipolar format is twice that of the bipolar format[13].The unipolar representation of stochastic bit streams is utilized in this paper.

2.2. Stochastic Operations

Addition and multiplication: The scaled addition is used by using simple multiplexer (MUX) which can replace the normal addition. Similarly the scaled multiplication is performed by using simple AND gate for the unipolar format [13].

Stochastic mean circuit (SMC): SMC is used to average 2^n input bit streams, and produces a single output. It uses a 2^n to 1 MUX with n uncorrelated select bit streams, each one representing the 0.5 value [14].

Square root: A stochastic square root circuit presented in [3], two different pulse streams can represent the same value with a different pattern of pulses. The circuit is looking for a stochastic pulse stream that tends to the input stream when multiplied by itself.

Subtraction: For independent inputs, the XOR gate performs the function $z = x_1*(1 - x_2) + x_2*(1 - x_1)$. However, when fed with correlated inputs where x_1 and x_2 have maximum overlap of 1s, the circuit computes $z = |x_1 - x_2|$[12][14].

Image processing is another potential application area for SC of great practical importance. Many imaging applications involve functional transformations on the pixels of an input image [11]. The pixel level functions are usually simple, but because of the large number of pixels involved, the overall transformation process is extremely computation intensive. If these functions are implemented using SC, then low cost, highly parallel image processing becomes possible, as has been demonstrated in a smart SC based image sensing chip.

2.3 Fault Tolerance

Fault tolerance[9] is the property that enables a system to continue operating properly in the event of the failure of (or one or more faults within) some of its components. If its operating quality decreases at all, the decrease is proportional to the severity of the failure, as compared to a naively designed system in which even a small failure can cause total breakdown. Fault tolerance is particularly sought after in high-availability or life-critical systems. A fault-tolerant design enables a system to continue its intended operation, possibly at a reduced level, rather than failing completely, when some part of the system fails.

The advantage of the stochastic architecture[10] in terms of resources is that it tolerates faults gracefully. Compare a stochastic encoding to a standard binary radix encoding, say with M bits representing fractional values between 0 and 1. Suppose that the environment is noisy; bit flips occur and these afflict all the bits with equal probability. With a binary radix encoding, suppose that the most significant bit of the data gets flipped. This causes a relative error of $(2^{M-1})/2^M=1/2$. In contrast, with a stochastic encoding, the data are represented as the fractional weight on a bit stream of length 2^M. Thus, a single bit fli

p only changes the input value by $1/2^M$, which is small in comparison [9].

3. ALGORITHMS

3.1. Stochastic Implementation of sauvola algorithm

For stochastic implementation one have to scale down all pixel intensities values from [0, 255] to [0, 1] interval. In the Sauvola equation, *R is* constant which is assumed to be 1. The modified Sauvola equation [14] in stochastic will be

$$t(x, y) = m \cdot [1 + 0.5(S - 1)]$$
$$= m \cdot (S + 1)/2$$

Where m is average value and S is standard deviation output. The processing on the input pixel values are performed by using section 2.1. The conversions of pixel values into stochastic streams are performed by using randomizer unit. The generation of threshold streams are performed by using fig.1 consists of SMC, circular shifter, scaled addition, AND gate, XOR gate and stochastic square root circuit which is placed in processing unit. Initially all the pixel values are converted into the stochastic streams and feeds as a input to SMC. The SMC produces the average output. In order to obtain mean$(x)^2$ AND gate is used, first input of the AND gate is output of SMC and second input is provided by shifting the SMC output using circular shifter to offer uncorrelated version of input. Having mean(x^2) and mean$(x)^2$ bit streams, the next step is to perform scaled subtraction using a XOR gate. The XOR output is fed as a input to the stochastic square root circuit. The result of square root circuit is 'S' which is the standard deviation output that is fed as a input to the scaled addition. The second input of MUX is considered to be '1'. The resulting output of the scaled addition and SMC are provided as input to the AND gate in order to obtain the threshold bit streams. Next to convert the stochastic bit streams into the binary bit streams a de randomizer unit is used. Fig. 1 shows the stochastic implementation of sauvola thresholding algorithm.

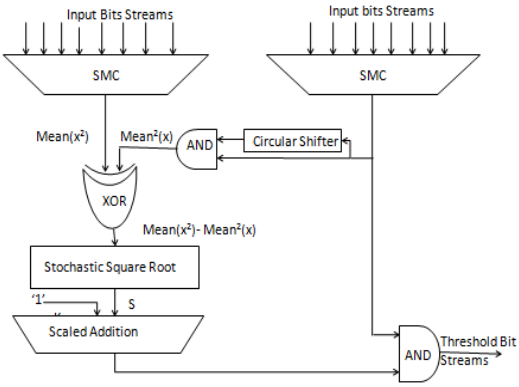


Fig -1: Stochastic implementation of sauvola thresholding algorithm.

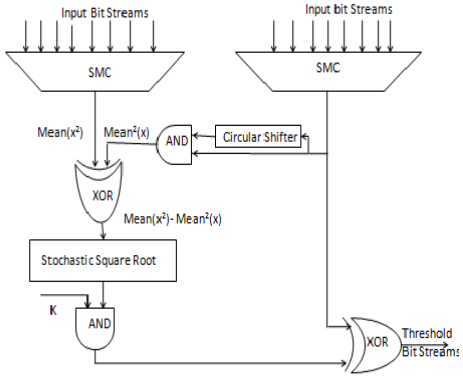


Fig -2: Stochastic implementation of niblack thresholding algorithm.

3.2 Stochastic implementation of niblack algorithm

Similar to the stochastic implementation of sauvola algorithm, the niblack algorithm [6] is also designed to scale down all pixel intensities from [0, 255] to [0, 1] interval. The K value is assumed to be –0.2 or -0.1 based on the noise present in images. The new modified Niblack equation for our stochastic design will be

$$t(x, y) = m + k(S) = m - 0.2S$$

Where m is average value and S is standard deviation output. The processing on the input pixel values are performed by using section 2.1. The conversions of pixel values into stochastic streams are performed by using randomizer unit. The generation of threshold streams are performed by using fig.1 consists of SMC, circular shifter, scaled addition, AND gate, XOR gate and stochastic square root circuit which is placed in processing unit. Initially all the pixel values are converted into the stochastic streams and feeds as a input to SMC. The SMC produces the average output. In order to obtain mean$(x)^2$ AND gate is used, first input of the AND gate is output of SMC and second input is provided by shifting the SMC output using circular shifter to offer uncorrelated version of input. Having mean(x^2) and mean$(x)^2$ bit streams, the next step is to perform scaled subtraction using a XOR gate. The XOR output is fed as a input to the stochastic square root circuit. The result of square root circuit is 'S' which is the standard deviation output that is fed as a input to the AND gate. First, we should do a simple AND gate to convert the s bit stream to $0.2S$. Second, XOR gate for producing the threshold bit streams of m – 0.2S. Fig. 2 shows the stochastic implementation of niblack thresholding algorithm.

4. SIMULATION RESULTS

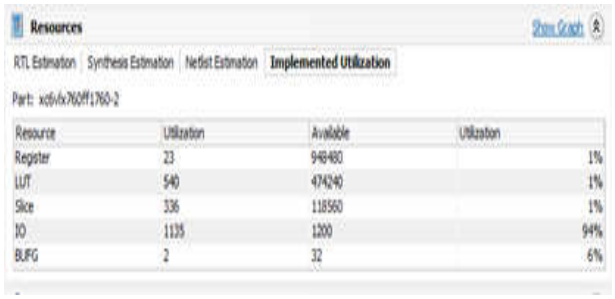
In this paper the stochastic implementation of sauvola and niblack algorithm is implemented by Verilog using Xilinx 14.5. The comparison is made between hardware resource utilization, area and threshold value, by implementing all 16-, 32-, 64-, 128- and 256- bit stream for stochastic architectures. By using cadence 6.1.5 area is analyzed.

4.1. Hardware resource utilization comparison

The stochastic implementation reduces the hardware resource utilization than conventional implementation. Table I shows the comparisons of sauvola and niblack algorithm for hardware resource utilization, in which the sauvola algorithm have less hardware resource utilization than niblack algorithm. Fig 3 & 4 shows the hardware resources utilization for niblack and sauvola Algorithm.

Table -1: Comparison of hardware resources utilization for niblack and sauvola algorithm

Resources	Hardware Resources Utilization							
	Niblack Algorithm				Sauvola Algorithm			
	16 Bits Streams	32 Bits Streams	64 Bits Streams	128 Bits Streams	16 Bits Streams	32 Bits Streams	64 Bits Streams	128 Bits Streams
Register	23	23	23	23	23	23	23	23
LUT	99	180	227	540	91	194	223	537
Slice	66	115	167	336	61	107	163	345
IO	215	351	615	1135	207	343	607	1127
BUFG	2	2	2	2	2	2	2	2

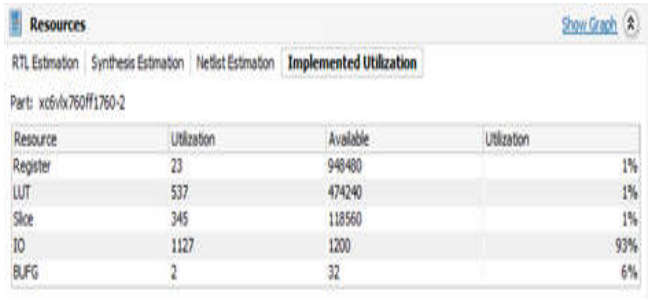

Resources

RTL Estimation | Synthesis Estimation | Netlist Estimation | Implemented Utilization

Part: xc6vlx760ff1760-2

Resource	Utilization	Available	Utilization
Register	23	948480	1%
LUT	540	474240	1%
Slice	336	118560	1%
IO	1135	1200	94%
BUFG	2	32	6%

Fig -3: Hardware resources utilization of niblack algorithm for 128 bit streams


Resources

RTL Estimation | Synthesis Estimation | Netlist Estimation | Implemented Utilization

Part: xc6vlx760ff1760-2

Resource	Utilization	Available	Utilization
Register	23	948480	1%
LUT	537	474240	1%
Slice	345	118560	1%
IO	1127	1200	93%
BUFG	2	32	6%

Fig -4: Hardware resources utilization of sauvola algorithm for 128 bit streams

4.2. Area *comparison*

Table 2 shows the comparisons of sauvola and niblack algorithm for area, in which the sauvola algorithm have less area than niblack algorithm, which is analyzed on basis of number of cells. Fig 5 & 6 shows the area of niblack and sauvola algorithm for 256 input bit streams.

Table -2: Comparison of area for niblack and Sauvola Algorithm

No. Of Bit Streams	Area(Cells)	
	Niblack Algorithm	Sauvola Algorithm
16	31605	31233
32	50869	51614
64	90739	90074
128	168311	168051
256	309486	309299

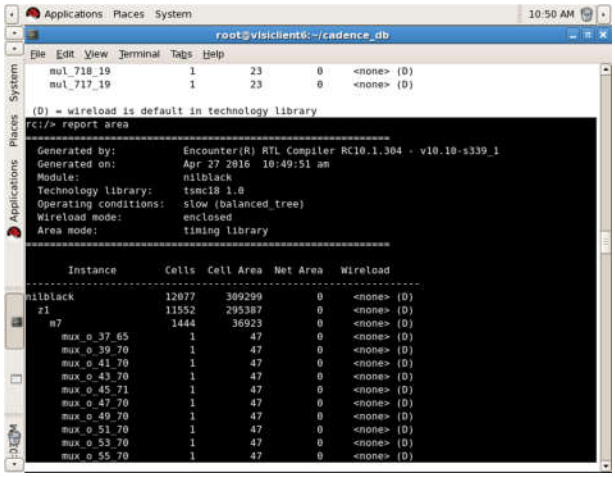


Fig -5: Area of niblack algorithm for 256 input bit streams

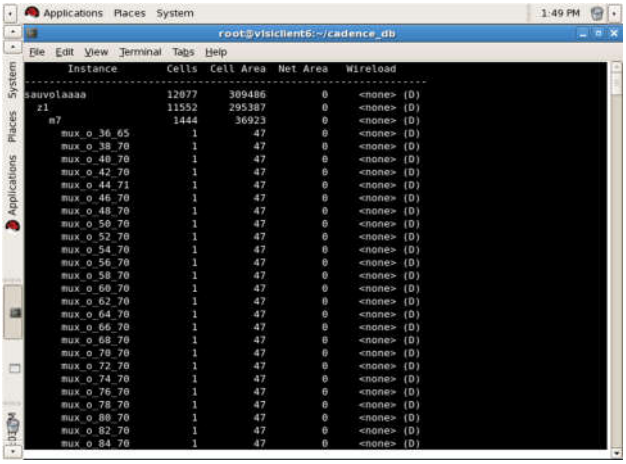


Fig -6: Area of sauvola algorithm for 256 input bit streams

4.3. Threshold value

Table 3 shows the comparisons of sauvola and niblack algorithm for threshold value, in which the sauvola algorithm have better threshold value than niblack algorithm. Fig 7 & 8 shows the Simulation waveform of niblack algorithm for 256 input bit streams. On the basis of probability, the sauvola algorithm has a threshold value of 0.5 which gives a better performance when compared with niblack algorithm which has a threshold value of 0.125.

Table -3: Comparison of threshold value for niblack and sauvola algorithm

No. Of Bit Streams	Threshold Value	
	Niblack Algorithm	Sauvola Algorithm
16	10000000(1/8)	01010101(4/8)
32	00100000(1/8)	00110010(4/8)
64	00000010(1/8)	01001101(4/8)
128	00000001(1/8)	11100001(4/8)
256	00100000(1/8)	00011101(4/8)

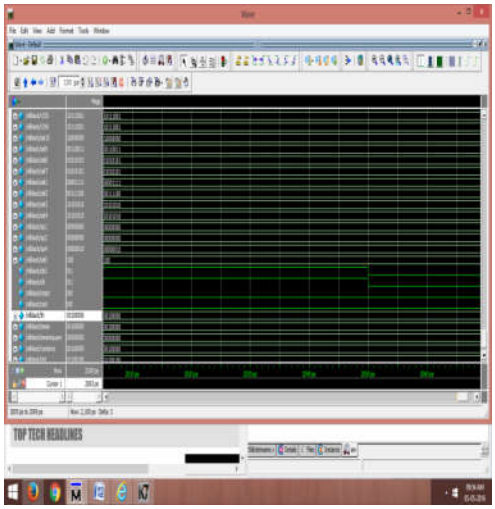


Fig -7: Simulation waveform of Niblack algorithm for 256 input bit streams

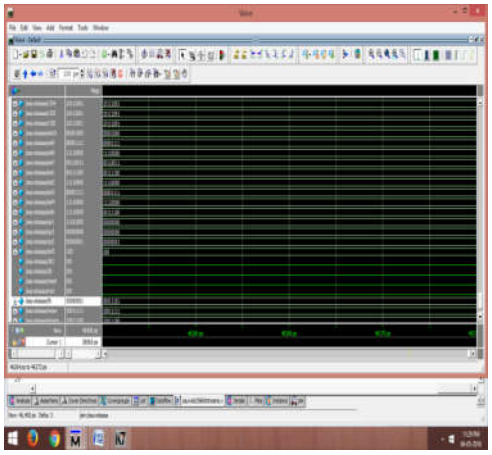


Fig -8: Simulation waveform of sauvola algorithm for 256 input bit streams

CONCLUSION

Stochastic computing architecture provides a superior performance compared with the conventional architecture. In this paper, the niblack and Sauvola algorithm are implemented using stochastic computing and analyzed using Xilinx 14.5 for hardware resource utilization and threshold value, and cadence 6.1.5 for area. The simulation results show the superior performance of sauvola algorithm when compared with niblack algorithm in terms of hardware resources utilization, area and threshold value. The simulation results shows that the algorithm obtains better threshold values for improving the quality of image. In the future work, the sauvola and niblack algorithms will be analyzed for delay, power consumption and area to show the superior performance of Sauvola than niblack algorithm. The sauvola algorithm will be analyzed and compared with other thresholding algorithms for document images to obtain better performance in terms of hardware resource utilization, delay, power consumption, area and threshold value.

REFERENCES

1. Otsu. N, "A threshold selection method from gray-level histograms," *IEEE Trans. Syst., Man, Cybern.*, vol. 9, no. 1, pp. 62–66, Jan. 1979.
2. Sauvola. J and Pietikäinen M, "Adaptive document image binarization,"*Pattern Recognit.*, vol. 33, no. 2, pp. 225–236, 2000.
3. *Toral. S.L, Quero. J.M and Franquelo L.G,* "stochastic pulse coded arithmetic", May 2000.
4. Brown. B. D and Card. H. C, "Stochastic neural computation. I. Computational elements," *IEEE Trans. Comput.*, vol. 50, no. 9, pp. 891–905, Sep. 2001.
5. Shafait. F, Keysers. D, and Breuel. T. M, "Efficient implementation of local adaptive thresholding techniques using integral images," in *Proc. 15th Doc. Recognit. Retr. Conf. (DRR-2008), Part IS&T/SPIE Int. Symp. Electron. Imag.*, vol. 6815. San Jose, CA, USA, Jan. 2008.
6. Khurshid. K, Siddiqi. I, Faure. C, and Vincent. N, "Comparison of Niblack inspired binarization methods for ancient documents," *Proc. SPIE, Doc. Recognit. Retr. XVI*, vol. 7247, 2009.
7. Li. P and Lilja. D. J, "A low power fault-tolerance architecture for the kernel density estimation based image segmentation algorithm," in *Proc. IEEE Int.Conf. Appl.-Specific Syst., Archit. Process. (ASAP)*, pp. 161–168, Sep. 2011.
8. Li. P and Lilja. D. J, "Using stochastic computing to implement digital Image processing algorithms," in *Proc. IEEE 29th Int. Conf. Comput. Design*, pp.154–161, Oct. 2011.
9. Qian. w, Li. X, Riedel. M. D, Bazargan. K, and Lilja D. J, "An architecture for fault-tolerant computation with stochastic logic," *IEEE Trans. Comput.*, vol. 60, no. 1, pp. 93–105, Jan. 2011.
10. David J. Lilja, " A Stochastic Reconfigurable Architecture for Fault-Tolerant Computation with Sequential Logic" IEEE 30th International Conference on Computer Design (ICCD)", *pp: 303-308 , 2012.*
11. Alagh. A and Hayes. J. P, "Survey of stochastic computing," *ACM Trans, Embedded Comput. Syst.*, vol. 12, no. 2s, pp. 1–19, 2013.
12. Alaghi. A, Li. C, and Hayes. J. P, "Stochastic circuits for real-time image-processing applications," in *Proc. 50th ACM/EDAC/IEEE Design Autom.Conf. (DAC)*, pp. 1–6, May/Jun. 2013.
13. P. Li, D. J. Lilja, W. Qian, K. Bazargan, and M. D. Riedel, "Computation on stochastic bit streams digital image processing case studies," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 3, pp. 449–462, Mar. 2014.
14. Hassan Najafi. M and Mostafa Salehi. E "A Fast Fault-Tolerant Architecture for Sauvola Local Image Thresholding Algorithm Using Stochastic Computing" IEEE Trans. Very Large Scale Integr. (VLSI) Syst., 2015.