A Design of Stochastic Circuit With Exact Synthesis

Xiang He, and Zhufei Chu*
EECS, Ningbo University, Ningbo 315211, China
* Email: chuzhufei@nbu.edu.cn

BIOGRAPHY

The corresponding author Zhufei Chu received the B.S. degree in electronic information science and technology from Shandong University, Weihai, China, in 2008 and the M.S. degree in communication and information system from Ningbo University, Ningbo, China, in 2011, and the Ph.D. degree in the same subject, in 2014. From 2014 to 2017, he is a lecturer at Ningbo University. Now, he holds an Associate Professor position at Ningbo University. During 2016 to 2017, he works as a researcher at the Integrated Systems Laboratory at EPFL, Lausanne, Switzerland in the group of Prof. Giovanni De Micheli.

ABSTRACT

Stochastic computing is an emerging paradigm that converts traditional binary numbers into stochastic bit streams to perform complex arithmetic computing using simple digital circuits. Stochastic computing are highly fault-tolerant and have been used in many applications. However, due to the large solution space of the problem, the use of heuristics is complex and does not lead to an optimum solution. In this paper, a novel exact synthesis method using Boolean satisfiability (SAT) is proposed to obtain an optimal stochastic circuit, which is a dedicated homogeneous network by using majority-inverter graphs (MIGs) as underlying logic primitives. The experimental results show that the exact synthesis approach produces better circuits, compared with the heuristic approach.

***Index Terms*—stochastic computing; exact synthesis; logic synthesis; majority**

INTRODUCTION

Modern circuit development is limited by a number of constraints, such as voltage variations, thermal variations and soft errors. These physical phenomena are susceptible to errors, so reliability has become an important issue. Stochastic computing has received increasing attention as an unconventional computing method for solving these problems. It is unique in that it represents and processes information in probabilistic form and is highly fault-tolerant for bit flips. Stochastic computating converts numbers $x \in [0,1]$ into stochastic bit streams containing only 0 and 1. The number is represented by the probability of a one in the bit stream. Thus stochastic computing transforms a complex computing unit into a simple circuit with gate. For example, multiplication in stochastic computing can be implemented with an AND gate, as shown in Fig.1. Assume that the two input stochastic bit streams are independent. Stochastic computing has been applied in many applications [1] such as image processing, neural networks, decoding of LDPC, filter design, and so on.

The work was supported by NSFC under Grant 61871242.

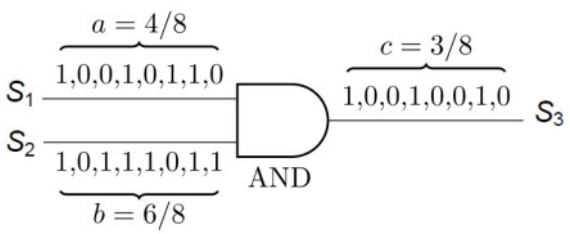

Fig. 1. AND gate performs multiplication by stochastic bit streams.

In stochastic computing, many different Boolean functions can be implemented to compute the same function, which gives great design scope for the synthesis of stochastic circuits. In order to adapt stochastic computing to more functions, researchers have recently proposed several synthesis methods. One of the state-of-the-art methods is a heuristic breadth-first search algorithm [2],which is based on assigning cubes (i.e., product terms) to the on-set of the Boolean function. However, heuristics cannot find optimal solutions. Exact synthesis is a new approach, which is a problem of finding logical networks that represent given Boolean functions and respect given constraints. With exact synthesis it is possible to find optimum networks, e.g., in size or depth. In this paper, we use exact synthesis method to obtain an optimal stochastic computing circuit, which is a dedicated homogeneous network by using MIGs as only logic primitives. The experimental results show that the exact synthesis approach produces better circuits, compared with the heuristic approach.

BACKGROUND

Figure. 2 shows a general form of stochastic circuit with combination circuit. For a function of single variable x, the stochastic number generator (SNG) generates n inputs $X_1, \ldots, X_n$, whose probability is x, where n is the highest degree of the variable x in the function. SNGs usually consist of a pseudo-random number generator and a comparator. In addition, the linear feedback shift register (LFSR) generates m inputs $Y_1, \ldots, Y_n$ with probability 0.5. The stochastic bit

streams of these n+m inputs are processed by a combination logic circuit.

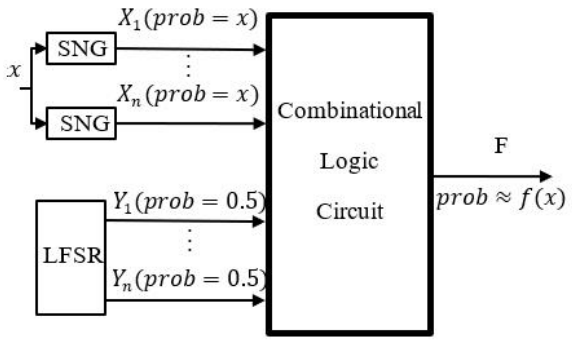

Fig. 2. A general form of stochastic computing circuits.

As shown in [2], the output probability of the circuit in Fig. 2 obtains $G(i)$ is the number of minterms $(X_1, \ldots, X_n, Y_1, \ldots, Y_m)$ satisfying that $F(X_1, \ldots, X_n, Y_1, \ldots, Y_m) = 1$ and $\sum_{j=1}^{n} X_j = i$. $(G(0), \ldots, G(n))$ is recorded as the problem vector. For example, (0, 3, 2) is a problem vector using the truth table shown in Fig. 3 with $n = 2$ and $m = 2$. Its output probability realizes the following function

$$f(x) = \frac{3}{4}x(1-x) + \frac{2}{4}x^2 \tag{1}$$

TABLE I
THE TRUTH TABLE OF A BOOLEAN FUNCTION WITH PROBLEM VECTOR (0, 3, 2).

$Y_1Y_2\backslash X_1X_2$	00	01	11	10
00		1	1	1
01		1	1	
11				
10				

The Boolean function shown in Fig. 3 is represented in a simplified sum-of-product (SOP) form.

$$F = X_2\overline{Y_1} + X_1\overline{Y_1Y_2} \tag{2}$$

In this way, the problem vector can form a logical circuit for stochastic computing, but there are many kinds of Boolean functions of the same problem vector. Fig. 3 is just one of the 356 Boolean functions of problem vector (0, 3, 2).

PROPOSED DESIGNS

Exact synthesis is the problem of finding optimal logic networks by giving a set of primitives. It is well know that the optimal size of a normal network is found via SAT formula [3]. Based on the Knuth algorithm [4], we propose our encoding method about Majority operations. Being a, b, and c the 3 inputs, each gate can realize $< abc >$, $< \bar{a}bc >$, $< a\bar{b}c >$ or $< ab\bar{c} >$. By setting any variable to constant 0 or 1, one can behave as AND or OR, respectively. ab, $\bar{a}b$, $a\bar{b}$, and $a+b$ should also be considered. The defined variables [5] are constrained by a set of clauses. For each gate r, the operation variable o_{rw} is true if the operation of gate r is w, where w is one of the 8 possible normal majority operations.

For example, suppose the selection variable S_{iabc}, and the input combination is (100), then we can check $o_{r1} =< abc >$ outputs 0, while $o_{r8} = a + b$ outputs 1. Thus the clause is added as follows.

$$\begin{aligned}
&\bar{S}_{iabc} \vee \bar{x}_{it} \vee o_{r3} \vee o_{r4} \vee o_{r7} \vee o_{r8} \\
&\bar{S}_{\text{iabc}} \vee \bar{x}_{it} \vee \bar{o}_{r1} \\
&\ldots \\
&\bar{S}_{\text{iabc}} \vee x_{it} \vee o_{r1} \vee o_{r2} \vee o_{r5} \vee o_{r6} \\
&\bar{S}_{iabc} \vee x_{it} \vee \bar{o}_{r3} \\
&\ldots \\
&\bar{S}_{iabc} \vee x_{it} \vee \bar{o}_{r8}
\end{aligned} \tag{3}$$

Further, clause $\bigvee_{w=1}^{8} o_{rw}$ ensure that each step should realize at least one of the 8 operations.

Given a problem vector, we use exact synthesis algorithm to find a solution by assuming $r = 1$ gate. If a solution is found, it returns MIG; otherwise, the algorithm will increase the number of gates r, then restart encoding and solve until the upper limit is reached. This will ensure that the algorithm can find the MIG network with the best number of gates.

EXPERIMENTAL RESULT

In this section, we apply our exact synthesis method to synthesize some common arithmetic functions by our open-source logic synthesis tool ALSO[1] command '`stochastic`'. Table. 2 show the comparions between method in [2] and our method. The two methods use the same precision m and degree of functions n, respectively. We also evaluate the mean absolute error (MAE) and area-delay product(ADP). The results of simulation show that our approach produces better circuits in terms of area and delay.

TABLE II
COMPARISONS OF SOME ARITHMETIC FUNCTIONS

Function	m	n	Method in [2]			Our method				
			Area	Delay	ADP	MAE	Area	Delay	ADP	MA
$\sin(x)$	2	4	15	4.5	67.5	0.0221	11	3	33	0.03
$\cos(x)$	3	3	3	1.1	3.3	0.0080	3	1.1	3.3	0.00
$\tanh(x)$	3	2	14	3.3	46.2	0.0074	7	3	21	0.00
$\log_2(1+x)$	5	2	16	3.4	54.4	0.1810	14	4.5	63	0.18
e^{-x}	3	2	8	3.1	24.8	0.0084	8	3.1	24.8	0.00
$sin(\pi x)$	5	2	12	3.1	37.2	0.1829	11	3.1	34.1	0.18

REFERENCES

[1] A. Alaghi, W. Qian and J. P. Hayes, "The Promise and Challenge of Stochastic Computing," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 37, no. 8, pp. 1515-1531, 2018.

[2] X. Peng and W. Qian, "Stochastic Circuit Synthesis by Cube Assignment," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 37, no. 12, pp. 3109-3122, 2018.

[3] Z. Chu, W. Haaswijk, M. Soeken, Y. Xia, L. Wang and G. De Micheli, "Exact Synthesis of Boolean Functions in Majority-of-Five Forms," 2019 IEEE International Symposium on Circuits and Systems, pp. 1-5, 2019.

[1] https://github.com/nbulsi/also

[4] D. E. Knuth, 'The Art of Computer Programming," Volume 4,Fascicle 6: Satisfiability. Addison-Wesley Professional, 2015.
[5] E. Testa, M. Soeken, O. Zografos, F. Catthoor, and G. De Micheli, "Exact synthesis for logic synthesis applications with complex constraints," International Workshop on Logic and Synthesis (IWLS), pp. 21–27, 2017.