Stochastic Circuit Design Based on 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.

ABSTRACT

Stochastic computing is an emerging paradigm that converts binary numbers into stochastic bitstreams to perform complex arithmetic computing using simple digital circuits. Stochastic computing is highly fault-tolerant and has been used in many applications. However, due to the large solution space of the problem, stochastic circuit synthesis is more complicated than the traditional logic synthesis. Previous methods use a heuristic method to synthesize a stochastic circuit. In this paper, a novel exact synthesis method using Boolean satisfiability (SAT) is proposed to obtain an optimal stochastic circuit represented by majority-inverter graphs (MIGs), which is a dedicated homogeneous logic network. The experimental results show that the proposed approach can achieve 21% area reduction, 4% delay improvement, with 3% mean absolute error trade-off.

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

INTRODUCTION

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

The work was supported by NSFC under Grant 61871242.

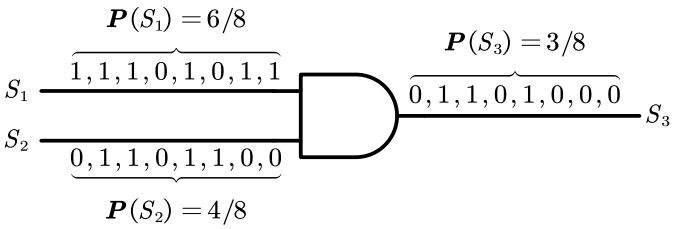

Fig. 1. AND gate performs multiplication by stochastic bitstreams.

In stochastic computing, many different Boolean functions can be implemented to compute the same function, which has large solution space for the synthesis of stochastic circuits. 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 represented by MIGs. The experimental results show that the exact synthesis approach achieves improvements in area and delay, compared with the heuristic approach.

BACKGROUND

A general form of the stochastic circuit is shown in Fig. 2, which is a combinational 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 $Y_1, \ldots, Y_m$ with probability 0.5. The stochastic bitstreams of these $n+m$ inputs are processed by a combination logic circuit.

For $i \in [0, n]$, let $G(i)$ represents the number of minterms $(X_1, \ldots, X_n, Y_1, \ldots, Y_m)$ that satisfying

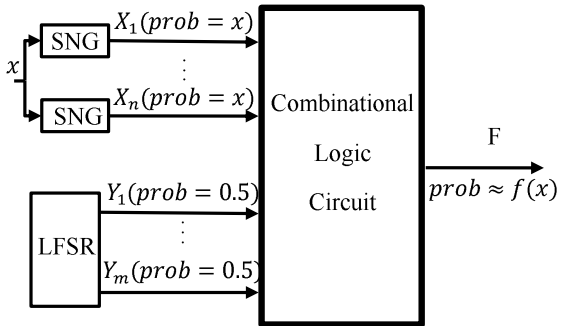

Fig. 2. A general form of stochastic computing circuits.

$F(X_1, \ldots, X_n, Y_1, \ldots, Y_m) = 1$ and $\sum_{j=1}^{n} X_j = i$. $(G(0), \ldots, G(n))$ is recorded as the problem vector [2]. For example, (0, 3, 2) is a problem vector using the truth table shown in Table I with $n = 2$ and $m = 2$. Its output probability realizes the 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 Table I is represented in a simplified sum-of-product (SOP) form

$$F = X_2\bar{Y_1} + X_1\bar{Y_1}\bar{Y_2}. \tag{2}$$

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

PROPOSED METHOD

Exact synthesis is the problem of finding optimal logic networks by giving a set of primitives. It is well known that the optimal size of a normal network is found via the SAT formula [3]. Based on the Knuth algorithm [4], the encoding method using majority-of-three operations is proposed [6].

Note that the traditional exact synthesis makes use of fixed truth table as input, the output is the optimal logic representations in size or depth. In this paper, since the stochastic circuit has problem vector, we add cardinality clauses to constraint the on-sets of logic function. In this way, the SAT solver can answer whether we can realize a stochastic problem vector using giving primitives.

For example, we know that $G(2) = 2$ according to the problem vector (0, 3, 2) in Table I. We restrict the sum of all minterms under columns "01" and "10" to be equal to $G(2)$. Let $f_{Y_1Y_2X_1X_2}$ be a minterm, then all minterms under columns "11" are f_{0011}, f_{0111}, f_{1011}, and f_{0011}. Thus the cardinality constraint is

$$f_{0011} + f_{0111} + f_{1011} + f_{0011} = 2. \tag{3}$$

Note that this constraint should be transformed into conjunction normal forms (CNFs) which are feasible for SAT solving.

Given a problem vector, our algorithm can find a solution to satisfy both problem vector cardinality constraints and given logical primitives. If a solution is found, it returns a MIG; otherwise, the algorithm will increase the number of gates, then restart encoding and solve until the upper limit is reached. This will ensure that the algorithm can find the MIGs network with the optimal number of gates.

EXPERIMENTAL RESULT

In this section, we apply our exact synthesis method to synthesize several common arithmetic functions by our open-source logic synthesis tool ALSO[1] using the command '`stochastic`'. Table. II shows the comparison results between the method in [2] and ours. 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 area and delay are reported by ABC[2]. Generally, we can achieve 21% reduction in area, and 4% improvement in delay. In terms of ADP, our method has 23% reduction on average. The MAE is slightly increased as a trade-off, which is affordable.

TABLE II
COMPARISONS OF SOME ARITHMETIC FUNCTIONS.

Functions	m	n	Method in [2]				Our Method			
			Area	Delay	ADP	MAE	Area	Delay	ADP	MAE
$\sin(x)$	2	4	15	4.5	67.5	0.0221	11	3	33	0.0364
$\cos(x)$	3	3	3	1.1	3.3	0.0080	3	1.1	3.3	0.0079
$\tanh(x)$	3	2	14	3.3	46.2	0.0074	7	3	21	0.0075
$\log_2(1+x)$	5	2	16	3.4	54.4	0.1810	14	4.5	63	0.1809
e^{-x}	3	2	8	3.1	24.8	0.0084	8	3.1	24.8	0.0082
$sin(\pi x)$	5	2	12	3.1	37.2	0.1829	11	3.1	34.1	0.1830
Average			1	1	1	1	0.79	0.96	0.77	1.03

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.

[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.

[6] M. Soeken, L. G. Amarù, P. Gaillardon and G. De Micheli, "Exact Synthesis of Majority-Inverter Graphs and Its Applications," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 36, no. 11, pp. 1842-1855, Nov. 2017.

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

[2]https://github.com/berkeley-abc/abc