Stochastic Circuit Design Based on Exact Synthesis

Xiang He, and Zhufei Chu[*]

EECS, Ningbo University, Ningbo 315211, China

[*] Email: chuzhufei@nbu.edu.cn

ABSTRACT

Stochastic computing enables computationally complex arithmetic using binary numbers converted to stochastic bitstreams. A large number of applications have used stochastic computing due to its fault-tolerant nature. However, stochastic circuit synthesis presents a larger solution space when compared to classical logic synthesis. Previous methods synthesize stochastic circuits using a heuristic method. 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). The experimental results suggest that the proposed approach can achieve 21% area reduction, 4% delay improvement, with 3% mean absolute error trade-off.

INTRODUCTION

Many constraints limit modern circuit development, 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 (SC) has received increasing attention as an unconventional computing method for solving these problems. SC is unique in that it represents and processes information in the probabilistic form and is highly fault-tolerant for bit flips. SC converts numbers $x \in [0, 1]$ into stochastic bitstreams consisting only 0 and 1. Each bit has a probability x of being a one and probability $1 - x$ of being a zero in the bitstream. The values are represented by the probability of the one in the bitstream. For example, (0,1,0,0) and (0,0,0,1) are potential representations of the value 0.25 as the digit '1' presents once in the bitstream with length four. Numbers are normally stored in long bitstreams where a few bits are flipped without a significant difference in value. By converting numbers into random bitstreams, stochastic computing transforms a complex computing unit into a simple circuit with gates. Thus many arithmetic operations can be implemented with very simple logic circuits. For example, multiplication in stochastic computing can be implemented with an AND gate if the two input stochastic bitstreams are independent, as shown in Figure 1. Stochastic computing has been applied to many applications [1] as image processing, neural networks, decoding LDPC code, and filter design.

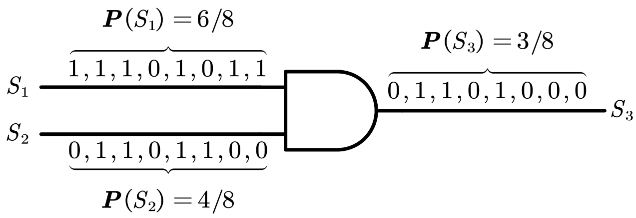

Figure 1: AND gate performs multiplication by stochastic bitstreams.

Multiple distinct Boolean functions can be used to compute the same function, which has a large solution space for the synthesize of stochastic circuits. Researchers have recently proposed several approaches for synthesizing functions in SC. A method that is currently being implemented [2] is based on assigning cubes (i.e., product terms) to the on-set of the Boolean function. The heuristics, however, do not yield optimal solutions. Exact synthesis is a new approach to finding Boolean networks that represent Boolean functions and respect given constraints. Exact synthesis allows one to find optimum networks, e.g., in size or depth. In this paper, we employ an exact synthesis method to obtain an optimal SC circuit represented by MIGs. Experimental results have shown that the exact synthesis approach outperforms the heuristic approach in terms of area and delay.

BACKGROUND

Stochastic Computing Circuits

A general form of the stochastic circuit is shown in Figure 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 $F(X_1, \ldots, X_n, Y_1, \ldots, Y_m) = 1$ 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

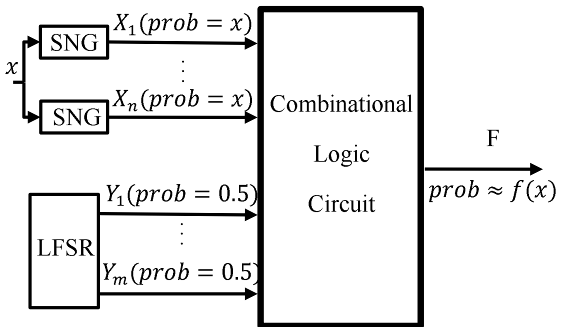

Figure 2: A general form of stochastic computing circuits.

Karnaugh map shown in Table I with $n = 2$ and $m = 2$. Its output probability realizes the function

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

TABLE I. The Karnaugh map of a Boolean function with problem vector $(0, 3, 2)$.

$\mathrm{Y_1Y_2 \backslash X_1X_2}$	00	01	11	10
00	0	1	1	1
01	0	1	1	0
11	0	0	0	0
10	0	0	0	0

The Boolean function shown in Table I is represented in a simplified sum-of-product (SOP) form

$$F = X_2\bar{Y_1} + X_1\bar{Y_1}\bar{Y_2}. \quad (2)$$

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. The problem vector $(0, 3, 2)$ indicates $G(0) = 0$, $G(1) = 3$, and $G(2) = 2$. As shown in Table I, the column '00' represents $G(0)$. Since $G(0) = 0$, then all the entries under this column must be all zero. Similarly, the column '11' represents $G(2)$, then we should assign two (i.e. $G(2) = 2$) '1's into 4 entries. Thus, there are $\binom{4}{2} = 6$ potential assignments. Furthermore, the columns '01' and '10' represents $G(1)$, we should assign three '1's into 8 entries, which results in $\binom{8}{3} = 56$ possibilities. Therefore, Table I shows just one of the $\binom{4}{2} \times \binom{8}{3} = 336$ Boolean functions of the problem vector $(0, 3, 2)$.

Sat-based Exact Synthesis

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].

1) Variables: For $1 \le h \le m$, $n < i \le n + r$, and $0 < t < 2^n$, the variables used in the SAT formulation are defined in the following:

$$\begin{aligned} x_{it} &: \quad t^{\text{th}} \text{ bit of } x_i \text{ 's truth table} \\ g_{hi} &: [g_h = x_i] \\ s_{ijk} &: [x_i = x_j \circ_i x_k] \ \text{ for } 1 \le j < k < i \\ f_{ipq} &: \circ_i (p, q) \text{ for } 0 \le p, q \le 1, p + q > 0 \end{aligned} \quad (3)$$

If function g_h is represented by gate x_i, variable g_{hi} is true. The select variables s_{ijk} is true if the inputs of gate x_i are x_j and x_k. The variable f_{ipq} is true, if the operation of gate x_i is true for the input assignment (p, q).

2) Clauses: Intuitively, if the gate x_i must operate as $b \circ_i c = a$. The main clauses to represent the operation constraints can be written as conjunction normal forms (CNFs), that is

$$(\bar{s}_{ijk} \vee (x_{it} \oplus a) \vee (x_{jt} \oplus b) \vee (x_{kt} \oplus c)) \vee (f_{ibc} \oplus \bar{a})) \quad (4)$$

Let $(t_1, \ldots, t_n)_2$ be the binary encoding of t, then the clauses

$$(\bar{g}_{hi} \vee (\bar{x}_{it} \oplus g_h (t_1, \ldots, t_n))) \quad (5)$$

constrain the output values to the gates they point to. Moreover, additional constraints can help to reduce the search space for the SAT solver [4].

PROPOSED DESIGNS

In this section, we will demonstrate the optimized design of stochastic circuit based on exact synthesis which can find optimal logic networks by giving a set of primitives.

Encoding

MIGs are used as underlying logic primitives for the synthesis of Boolean functions and containing AND/OR-inverter graphs. The variables to encode the truth table and the output gate are the same with Knuth's method. But the s_{ijk} and f_{ipq} need to be reexamined. In MIGs networks, each node has 3 children. The select variables s_{ijkl} is true if the operands of gate x_i are x_j, x_k, and x_l. The variable f_{ipqu} is true if the operation of gate x_i is true for the input assignment (p, q, u).

We use symbolic encoding method to represent all 8 3-input Majority Gates. The operation variable for step r is encoded as $O_{r1}, ..., O_{r8}$, we need add additional clauses to make the algorithm work.

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 $f_{Y_1 Y_2 X_1 X_2}$ be a minterm, then all minterms under columns "11" are f_{0011}, f_{0111}, f_{1011}, and f_{0011}. Thus the cardinality constraint is

$$f_{0011} + f_{0111} + f_{1011} + f_{0011} = 2. \qquad (6)$$

Note that this constraint should be transformed into CNFs which are feasible for SAT solving.

Due to the encoder works for normal function, we should also try the unnormal function, just redefine the problem vector, if $m = 1$, $n = 2$, the problem vector is $(1, 3, 2)$, then the inveterd problem vector is $(2 - 1 = 1, 4 - 3 = 1, 2 - 2 = 0)$.

Algorithms

Based on the encoding methods described above, given a problem vector, our algorithm can find a solution to satisfy both problem vector cardinality constraints and given logical primitives. The initial number of gates is set as 0. 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`', such as trigonometric, exponential, and logarithmic functions. The method in [2] is used to compare with our method. It is a state-of-the-art method in synthesizing these commonly used functions, using a heuristic breadth-first search algorithm.

Table II shows the comparison results between the method in [2] and ours. For a fair comparison, the two methods use the same precision m and degree of functions n, respectively. For each function $f(x)$, we chose nineteen input points $x = 0.05, 0.1, ..., 0.95$ for simulation. We chose the length of the stochastic bitstreams as 10240. 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, the result shows that the two methods have almost the same accuracy. But we can achieve 21% reduction in area, and 4% improvement in delay. In terms of ADP, our method has 23% reduction on average.

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

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

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

SUMMARY

In this paper, we proposed a method based on exact synthesis to synthesize general stochastic circuits. In stochastic computing, many different Boolean functions can implement the same function computation, which brings great design space for synthesis of stochastic circuits. Comparing with traditional heuristic approaches, our exact synthesis algorithm can easily solve complex constraint problems, and apply MIGs to implement function computation in stochastic circuits. The algorithm considers all different constraints and demonstrates that classical heuristic logic synthesis tools may lead to solutions that do not satisfy all requirements. The experimental results show that the proposed approach can achieve 21% area reduction, 4% delay improvement, with 3% mean absolute error trade-off.

In this paper, small circuits and functions of a single variable are considered, and future work will focus on the study of larger circuits and functions of multiple variables.

ACKNOWLEDGEMENT

The work was supported by NSFC under Grant 61871242.

REFERENCES

[1] A. Alaghi, W. Qian and J. P. Hayes, 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, 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, 2019 IEEE International Symposium on Circuits and Systems, pp. 1-5, 2019.

[4] D. E. Knuth, Volume 4,Fascicle 6: Satisfiability. Addison-Wesley Professional, 2015.

[5] E. Testa, M. Soeken, O. Zografos, F. Catthoor, and G. De Micheli, International Workshop on Logic and Synthesis (IWLS), pp. 21–27, 2017.

[6] M. Soeken, L. G. Amarù, P. Gaillardon and G. De Micheli, IEEE Transactions on Computer-Aided Design

of Integrated Circuits and Systems, vol. 36, no. 11, pp. 1842-1855, Nov. 2017.