Stochastic Functions Using Sequential Logic

Naman Saraf, Kia Bazargan, David J. Lilja, and Marc D. Riedel
Department of Electrical and Computer Engineering
University of Minnesota, Twin Cities
Minneapolis, MN, USA
{saraf012, kia, lilja, mriedel}@umn.edu

***Abstract*—Stochastic computing is a novel approach to real arithmetic, offering better error tolerance and lower hardware costs over the conventional implementations. Stochastic modules are digital systems that process random bit streams representing real values in the unit interval. Stochastic modules based on finite state machines (FSMs) have been shown to realize complicated arithmetic functions much more efficiently than combinational stochastic modules. However, a general approach to synthesize FSMs for realizing arbitrary functions has been elusive. We describe a systematic procedure to design FSMs that implement arbitrary real-valued functions in the unit interval using the Taylor series approximation.**

Keywords—Stochastic computing; Reversible Markov chains; Finite state machines; Rational functions; Taylor series

I. INTRODUCTION

CMOS technology scaling has improved the performance of digital systems by lowering the power consumption and increasing the clock frequencies. However, concerns about the reliability of nanometer scale CMOS devices have initiated research efforts into novel computing techniques. *Stochastic computing* is an alternative paradigm that performs arithmetic on random bit streams. A real number $x \in [0,1]$ is represented by a sequence of random bits, where each bit has a probability x of being logic 1. The random bit streams are processed by digital logic gates to generate output random bit streams, where the input and output real values are given by the *probability* of the random bit streams.

The notion of performing arithmetic on random bit streams appeared in the early works of von Neumann [1] and Gaines [2]. Gaines presented a number of stochastic modules for basic arithmetic operations such as addition, multiplication, division and integration. Stochastic computing has been used recently to implement complicated functions in image processing [3] and communications [4].

Stochastic modules incur a much lower hardware cost than a conventional implementation as exemplified by the stochastic multiplier, which requires only one AND gate for any input size. By comparison, a 3-bit carry-save multiplier requires 30 gates. In addition to the lower area cost, stochastic modules are also extremely resilient to bit errors. Unlike the weighted positional binary radix encoding, the stochastic representation of data by random bit streams is unweighted and leads to smaller errors in the event of bit flips. The fault tolerance of stochastic computing over a conventional implementation was demonstrated for an image processing application in [5].

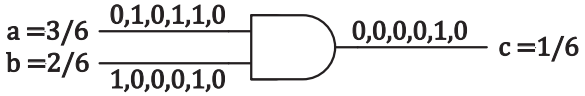

Fig.1 Stochastic multiplication, $c = ab$

Stochastic modules constructed from purely combinational logic have been used to implement polynomial functions of the input probability variables [6]. However, sequential logic based stochastic modules are more suited to realize complicated non-polynomial functions such as $exp(x)$ and $tanh(x)$ [7]. In spite of the tremendous potential of sequential logic based stochastic modules to realize complex functions, a systematic approach to their design has remained elusive. The authors in [7] presented specific examples of finite state machines (FSMs) realizing a few non-polynomial functions without providing a general synthesis strategy. More sophisticated functions were likewise described in [8].

The authors in [7] and [8] assume a fixed structure for the FSMs shown in Fig.2 and adjust the constants p_i (to be defined later) to minimize the error between the target function and the function realized by the FSM. The approach is not guaranteed to realize arbitrary functions since the structure of the FSMs is fixed. In contrast, we do not assume a specific structure for the FSMs and construct them individually for each target function. Our method transforms the target function to a Taylor series polynomial which itself is converted to a rational function and mapped to an FSM.

We begin by discussing Markov chains and introduce the special class of Reversible Markov chains. We present the prototype Reversible Markov chain (RMC) used in our designs and describe a systematic procedure to construct RMCs for realizing a given rational function. We next discuss the theory of function approximation to obtain a rational function from the Taylor series polynomial of a target function. We conclude with a comparison of the hardware costs of our approach with those previously reported in literature and outline the benefits of stochastic computing over deterministic processing.

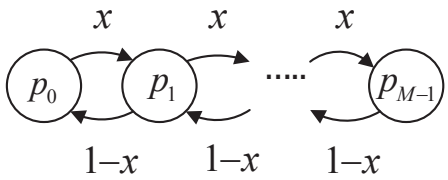

Fig.2 FSMs used in [7], [8] with x denoting the input probability variable

II. Reversible Markov Chains

Let us consider a discrete time stochastic system with M states and let S denote the state space of the system.

$$S = \{0,1,\dots,M-1\} \tag{1}$$

Let X_n denote the present state of the system. The stochastic system under consideration is a Markov chain if,

$$Prob\{X_{n+1} = j | X_n = i, X_{n-1} = i_{n-1}, \dots, X_0 = i_0\} = P_{ij} \tag{2}$$

for all states $i_{n-1}, i_{n-2}, \dots, i_0, i, j \in S$ and all $n \geq 0$.

Thus, the *transition probability* P_{ij} for a transition from the present state X_n of a Markov chain to the next state X_{n+1} does not depend on the previous history of the system.

A finite state Markov chain reaches equilibrium after a number of transitions from an initial state. At equilibrium, the probability of the Markov chain to be in state i becomes a constant, known as the *stationary probability* π_i such that,

$$\textstyle\sum_{i \in S} \pi_i = 1\,,\ \pi_i \geq 0,\ \forall\, i \in S \tag{3}$$

The stationary probabilities of a finite state Markov chain are unique if the Markov chain is ergodic. Ergodicity requires that any state of the Markov chain is accessible from any other state (irreducibility) at irregular time instants (aperiodicity). We will be using ergodic Markov chains in our designs.

The stationary probabilities of a general Markov chain are complicated functions of the transition probabilities. However, *Reversible* Markov chains (RMCs) satisfying the principle of detailed balance simplify the relationship to,

$$\pi_i P_{ij} = \pi_j P_{ji}, \qquad \forall\, i,j \in S, \qquad i \neq j$$
$$\implies \frac{\pi_i}{\pi_j} = \frac{P_{ji}}{P_{ij}} \triangleq r_{ij} \tag{4}$$

Thus, the stationary probabilities of an RMC are succinctly expressed in terms of transition ratios r_{ij}.

The prototype RMC used in our designs is constructed as a linear array of M states as shown in Fig.3. The stationary probabilities of the prototype RMC are expressed as,

$$\begin{aligned} \pi_1 &= r_{10}\pi_0 \\ \pi_2 &= r_{21}\pi_1 = r_{21}r_{10}\pi_0 \\ &\vdots \\ \pi_{M-1} &= r_{M-1\,M-2}\pi_{M-2} = r_{M-1\,M-2}r_{M-2\,M-3}\dots r_{10}\pi_0 \end{aligned} \tag{5}$$

π_0 is computed using (3) as,

$$\textstyle\sum_{i=0}^{M-1} \pi_i = 1$$
$$\pi_0[1 + (r_{10}) + \cdots + (r_{M-1\,M-2}r_{M-2\,M-3}\dots r_{10})] = 1$$
$$\implies \pi_0 = \frac{1}{[1 + (r_{10}) + \cdots + (r_{M-1\,M-2}r_{M-2\,M-3}\dots r_{10})]} \tag{6}$$

while the remaining stationary probabilities are calculated using (5) and (6). We next describe a method for constructing FSMs that implement a given rational function using the prototype RMC.

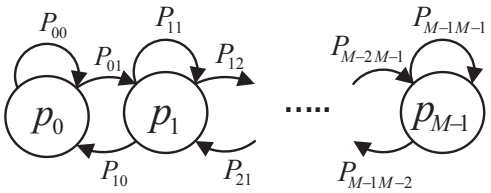

Fig.3 Prototype Reversible Markov chain with M states

III. Rational Functions Using Finite State Machines

Let us divide the state space S of the prototype RMC into S_{on} and S_{off} and generate an output bit from the RMC as,

$$\begin{aligned} Prob\{output\ bit = 1\} &= p_i,\ \ state \in S_{on} \\ Prob\{output\ bit = 1\} &= 0\,,\ \ state \in S_{off} \end{aligned} \tag{7}$$

The states of the RMC are mutually exclusive. Therefore, the probability of the output bit stream at equilibrium is given by,

$$P_{out} = \sum_{i \in S_{on}} p_i \pi_i \tag{8}$$

We interpret the non-negative probability as a real value Y,

$$\begin{aligned} Y &= 2P_{out} - 1 \\ 0 \leq P_{out} \leq 1 &\implies -1 \leq Y \leq 1 \end{aligned} \tag{9}$$

Let us consider the RMC in Fig.4. We do not annotate the transition probabilities P_{ii} on the state transition diagram for brevity. The transition ratios and the stationary probabilities of the RMC are given by,

$$r_{10} = \frac{x/2}{1/2} = x,\ r_{21} = \frac{1/2}{1/2} = 1,\ r_{32} = \frac{x/2}{1/2} = x$$
$$\pi_0 = 1/(1 + r_{10} + r_{21}r_{10} + r_{32}r_{21}r_{10}) = 1/(1 + 2x + x^2)$$
$$\pi_1 = r_{10}\pi_0 = x/(1 + 2x + x^2)$$
$$\pi_2 = r_{21}r_{10}\pi_0 = x/(1 + 2x + x^2)$$
$$\pi_3 = r_{32}r_{21}r_{10}\pi_0 = x^2/(1 + 2x + x^2) \tag{10}$$

where x is the input probability variable, $x \in [0,1]$.

The constants p_i of the RMC are indicated on the states in Fig.4. The probability function realized by the RMC becomes,

$$f_p(x) = \pi_2 + 0.25\pi_3 = (x + 0.25x^2)/(1 + 2x + x^2) \tag{11}$$

Using (9), the function implemented by the RMC is given by,

$$f(x) = 2f_p(x) - 1 = (-1 - 0.5x^2)/(1 + 2x + x^2) \tag{12}$$

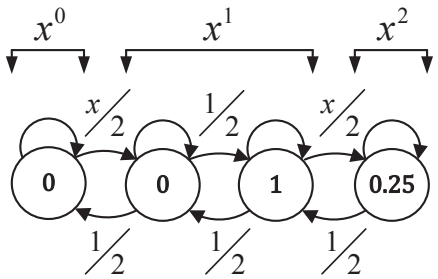

Fig.4 $f_p(x) = (x + 0.25x^2)/(1 + 2x + x^2)$

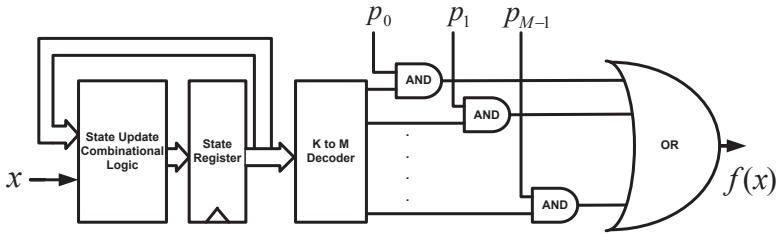

Fig.5 FSM implementation of the prototype Reversible Markov chain

A block diagram of the FSM implementing the prototype RMC is shown in Fig.5 The input random bit stream x updates the K-bit ($K = log_2 M$) state of the RMC every clock cycle. The K-bit state is decoded into M outputs such that the decoder output i represents the stationary probability π_i. The stationary probabilities are scaled by the constant random bit streams p_i using AND gates and ORed to produce the function value.

The states of the RMC in Fig.4 appear in distinct groups separated by the transition ratio x. The stationary probabilities of states in a group are equal ($\pi_1 = \pi_2$) while those in adjacent groups vary by a factor of x ($\pi_1 = x\pi_0,\ \pi_3 = x\pi_2$). We note that the coefficient of x^i in the denominator of $f_p(x)$ equals the total number of states in group i (numbered from left to right) while the numerator coefficient equals the number of states in group i with scaling. Therefore, by constructing an RMC using transition ratios from the set $\mathcal{R} = \{1, x\}$, we have simplified the analysis of the probability function from the transition diagram.

Let us now consider the inverse problem of constructing an RMC to realize a given rational function $g_p(x)$ using transition ratios from the set $\mathcal{R} = \{1, x\}$.

$$g_p(x) = \frac{a_0 + a_1 x + \cdots a_{M-1}x^{M-1}}{b_0 + b_1 x + \cdots b_{M-1}x^{M-1}} = \frac{\sum_{i=0}^{M-1}(a_i)x^i}{\sum_{i=0}^{M-1}(b_i)x^i} \tag{13}$$

$$0 \le g_p(x) \le 1 \tag{14}$$

$$b_i \in \mathbb{Z}^+, \qquad a_i \in [0, b_i], \qquad i = 0,1 \ldots M-1 \tag{15}$$

Equation (14) ensures that $g_p(x)$ equals a probability function. Equation (15) states that b_i must be a positive integer since it is equal to the total number of states in group i, while a_i cannot exceed b_i. The associated function $g(x)$ is given by,

$$g(x) = 2g_p(x) - 1$$

$$= \frac{\sum_{i=0}^{M-1}(2a_i - b_i)x^i}{\sum_{i=0}^{M-1}(b_i)x^i} = \frac{\sum_{i=0}^{M-1}(a_i')x^i}{\sum_{i=0}^{M-1}(b_i)x^i} \tag{16}$$

$$0 \le a_i \le b_i \Longrightarrow -b_i \le 2a_i - b_i \le b_i$$

$$\Longrightarrow |a_i'| \le b_i \tag{17}$$

In summary, given a rational function $g: [0,1] \to [-1,1]$,

$$g(x) = \frac{\sum_{i=0}^{M-1}(a_i')x^i}{\sum_{i=0}^{M-1}(b_i)x^i}, \quad b_i \in \mathbb{Z}^+, |a_i'| \le b_i \tag{18}$$

we can design an RMC to realize the function using transition ratios from the set $\mathcal{R} = \{1, x\}$. We next describe a method to map arbitrary functions into rational functions of the form in (18) using the theory of function approximation.

An order N Taylor approximation to a function is given by,

$$f_N(x) = c_0 + c_1 x + c_2 x^2 + \cdots c_N x^N + \mathcal{O}(x^{N+1}) \tag{19}$$

where the coefficients c_i are computed from the derivatives of the function $f(x)$ at a given point.

A *Pade`-type* approximant $g(x)$ of degree K to the function $f(x)$ is a rational function such that the Taylor series of $g(x)$ coincides with $f_N(x)$ upto K terms.

$$g(x) = \frac{\sum_{i=0}^{K}(a_i')x^i}{\sum_{i=0}^{K}(b_i)x^i} = f_N(x)$$

$$\Longrightarrow \sum_{i=0}^{K}(a_i')x^i = \sum_{i=0}^{K}(b_i)x^i \sum_{i=0}^{N}(c_i)x^i \tag{20}$$

The coefficients of $g(x)$ are determined from the coefficients c_i by equating the like powers of x. Therefore,

$$a_0' = b_0 c_0 \tag{21}$$

$$a_i' = b_i c_0 + \sum_{j=0}^{i-1} b_j c_{i-j}, 1 \le i \le K \tag{22}$$

$$-1 \le g(x) \le 1 \Leftrightarrow -1 \le f_N(x) \le 1, \qquad 0 \le x \le 1$$

$$\Longrightarrow -1 \le c_0 = f_N(0) \le 1 \Longrightarrow |c_0| \le 1 \tag{23}$$

$$\Longrightarrow b_0 = 1, \qquad a_0' = c_0 \tag{24}$$

Equation (22) is an underdetermined system with K equations in $2K$ unknowns which are solved sequentially starting with $i = 1$. Thus, each of the equations in (22) represents a line in the Cartesian plane of the unknown variables a_i' and b_i. Moreover, the constraints in (18) define a feasible region in the same plane as depicted in Fig.6. The optimal solution is obtained by the intersection of the line with the feasible region, having the smallest value of b_i as a smaller value of b_i implies an RMC with fewer states and a lower hardware cost.

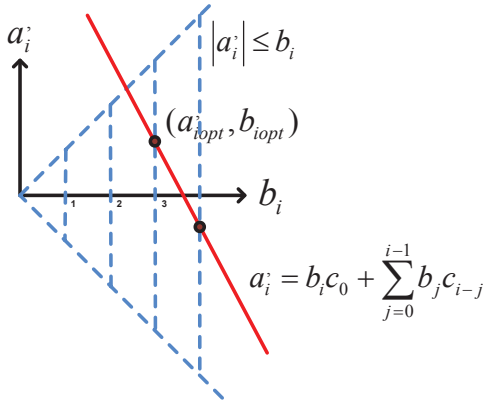

Fig.6 Optimal solution to $a_i' = b_i c_0 + \sum_{j=0}^{i-1} b_j c_{i-j}$, subject to $b_i \in \mathbb{Z}^+, |a_i'| \le b_i$

The probability function $g_p(x)$ is determined from $g(x)$ next,

$$g_p(x) = \frac{[g(x)+1]}{2} \tag{25}$$

$g_p(x)$ is mapped to an RMC using transition ratios from the set $\mathcal{R} = \{1, x\}$ which is subsequently implemented as an FSM.

IV. Experimental Results

We design FSMs to implement a number of complex non-polynomial functions (Fig.7-8) using our method and compare our results to those previously reported in [8] in Table I. The results show that our FSMs implement the functions using fewer states and hence reduce the overall cost of the system.

TABLE I. Comparison Of The Number Of States (Approximation Error < 10^{-3})

Function	Order of approximation K	Number of states in the FSM	
		This work	[8]
$\cos(x)$	7	8	-
$\sin(x)$	5	6	-
$\tanh(x)$	5	6	8
$\exp(-x)$	4	5	16
x^3	18	19	-
$\log(1+x)$	5	6	-

The performance metrics of an 8-bit CORDIC processor implementing trigonometric functions on a *Xilinx Virtex-5* FPGA are shown in Table II. The corresponding metrics of the FSMs designed using our approach are presented in Table III. In order to achieve the resolution of an M-bit binary system, a stochastic module requires a latency of 2^M clock cycles. Table IV compares the area-delay products of the designs and shows that stochastic modules offer superior performance over the conventional deterministic implementations.

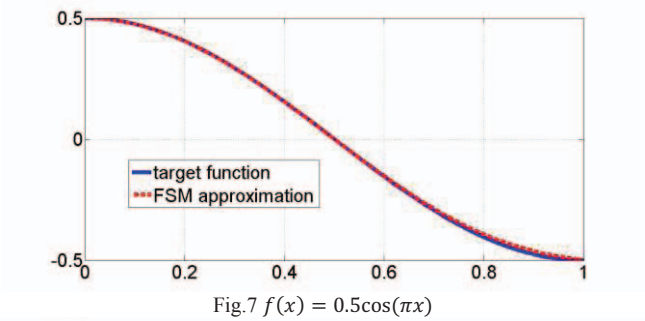

Fig.7 $f(x) = 0.5\cos(\pi x)$

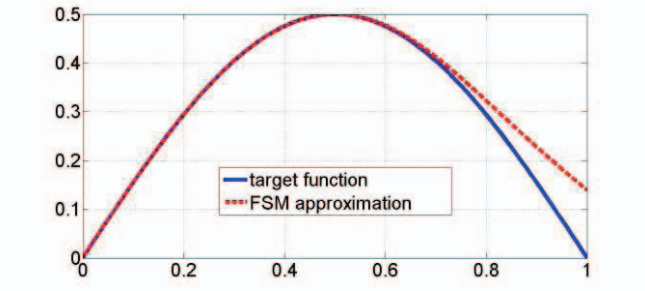

Fig.8 $f(x) = 0.5\sin(\pi x)$

TABLE II. Performance Metrics Of A Cordic Processor

Function	Area (Number of LUT-FF pairs)	Maximum Clock Frequency (MHz)	Latency (Clock cycles)
$\cos(x)$	265	237.8	12
$\sin(x)$	265	237.8	12

TABLE III. Performance Metrics Of The Stochastic Modules

Function	Area (Number of LUT-FF pairs)	Maximum Clock Frequency (MHz)	Latency (Clock cycles)
$\cos(x)$	11	640.6	256
$\sin(x)$	5	667.5	256

TABLE IV. Comparision Of The Area-Delay Product ($Area \times Latency/Clock\ Frequency$)

Function	CORDIC Processor	Stochastic Approximation	Stochastic / CORDIC
$\cos(x)$	13.4	4.4	0.33
$\sin(x)$	13.4	1.9	0.14

V. Conclusions

We have presented a general approach to design stochastic modules based on FSMs for implementing arbitrary functions in the unit interval. Based on the area-delay products, we conclude that stochastic computing offers better performance over deterministic processing.

References

[1] J. vonNeumann, "Probabilistic logics and the synthesis of reliable organisms from unreliable components," *Automata Studies*, Princeton University Press, pp.43-98, 1956.

[2] B.R. Gaines, "Stochastic computing systems," *Advances in Information Systems Science*, *Plenum*, vol.2, no.2, pp.37-172, 1969.

[3] P. Li, D.J. Lilja, "A low power fault-tolerant architecture for the kernel density estimation based image segmentation algorithm," *IEEE International Conference on Application-Specific Systems, Architectures and Processors*, pp.161-168, 2011.

[4] S.S. Tehrani, S. Mannor, W.J.Gross, "Fully parallel stochastic LDPC decoders," *IEEE Transactions on Signal Processing*, vol.56, no.11, pp. 5692-5703, 2008.

[5] W. Qian, X. Li, M.D. Riedel, K. Bazargan, and D.J. Lilja, "An architecture for fault-tolerant computation with stochastic logic," *IEEE Transactions on Computers*, vol.60, no.1, pp.93-105, 2011.

[6] W. Qian, M.D. Riedel, "The synthesis of robust polynomial arithmetic with stochastic logic," *Design Automation Conference*, pp.648-653, 2008.

[7] B.D. Card, H.C. Card, "Stochastic neural computation I: computational elements," *IEEE Transactions on Computers*, vol.50, no.9, pp.891-905, 2001.

[8] P. Li, D.J. Lilja, W. Qian, K. Bazargan, M. Riedel, "The synthesis of complex arithmetic computation on stochastic bit streams using sequential logic," *International Conference on Computer-Aided Design*, pp.480-487,2012.