Automated Synthesis of Stochastic Computational Elements using Decision Procedures

Amad Ul Hassen
Department of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida
Email: amad@knights.ucf.edu

Brigadesh Chandrasekar
Department of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida
Email: brigadesh@knights.ucf.edu

Sumit Kumar Jha
Department of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida
Email: jha@eecs.ucf.edu

***Abstract*—As integrated circuits move into the sub-10nm range, their reliability decreases due to reduced noise margin, radiation-induced errors and manufacturing variations. Stochastic circuits are inherently fault tolerant. Their instrinsic fault tolerance along with their area and power efficiency have made them competitive candidates for energy-hungry multimedia and pattern recogntion applications. We have proposed a new approach for the synthesis of stochastic circuits. We demonstrate the success of our approach by synthesizing polynomial, tanh and exponentiation functions. Our approach that employs decision procedures to effectively explore the space of linear finite state machines guarantees an upper bound on the maximum error between the synthesized function and its stochastic approximation. Our appraoch has resulted in 1.17 to 1.65 times smaller worst-case error as compared to the previous state-of-the-art.**

I. Introduction

Transistors in commercially available integrated circuits have shrunk to 14nm. Their size is projected to be 10nm in 2016-2017 and less than 7nm in 2020. At such a small scale, integrated circuits are ever more prone to process variations, radiation effects and temperature changes. Such uncertain behavior has driven the exploration of fault-tolerant computing paradigms. Due to their inherent fault tolerance, stochastic circuits are naturally more robust than deterministic circuits [1].

Stochastic computing has been around since 1960s. Von Neumann [2] introduced stochastic computing in his pioneering paper “Probabilistic Logics and Synthesis of Reliable Organisms from Unreliable Components”. Shannon described a method to build high reliability circuits from unreliable relays [3]. Gaines [4] defined stochastic computational element (SCE) as a simple circuit which performs computations on input Bernoulli streams and generates an output Bernoulli stream. Relationship between the probability of ‘1’ in input and output streams characterizes the computation performed. He described SCEs for elementary operations such as multiplication, scaled addition, squaring, inversion and integrator.

In 2001, Brown and Card [5] used finite state machine to approximate tanh and exponentiation functions using a stochastic bit stream. Li et al. [6] devised a scheme for the synthesis of SCE and demonstrated its effectiveness by synthesizing tanh, polynomial and exponentiation functions. The synthesized circuits were more compact and robust to errors as compared to their deterministic implementations. Their implementation also had better area efficiency as compared to circuits yielded by Bernstein polynomials [6].

TABLE I
Comparison of worst-case errors of our synthesized linear finite state machines with those synthesized using Li et al. [6]

Function	Li et al. Error	Our Error	Improvement
Polynomial	0.024	0.0145	1.65 times
tanh	0.065	0.04	1.625 times
exp	0.125	0.1119	1.17 times

Current state-of-the-art approach [6] synthesizes stochastic circuits by minimizing the mean square error between the target function and its estimate. This approach does not guarantee any upper bound on the maximum error between the target function and its estimate. To overcome this problem, we have proposed a new algorithm which minimizes absolute error between the target funtion and its estimate, thus guaranteeing an upper bound on the maximum error between the target function and its estimate over entire input range. Our algorithm employs decision procedures to search the space of linear finite state machines. We have demonstrated the superiority of our algorithm by synthesizing finite state machines for polynomial, tanh and exponentiation functions using same number of states as Li et al. [6]. As shown in Table 1, the circuits synthesized using our appraoch have 1.17 to 1.65 times smaller worst-case error than the previous state-of-the-art [6]. All of this improvement is achieved without any additional hardware or delay.

II. Linear Finite State Machines

Although simple elementary operations, such as multiplication and scaled addition, can be performed stochastically on small combinational circuits comprised of a few logic gates or multiplexers, more complex computations such as non-linear function approximation can also be performed easily using simple sequential circuits modeled by Finite State Machines (FSM). We are synthesizing single-input single-output linear Moore machines with probabilistic outputs for stochastic estimation of target functions.

Definition 1. A Moore machine with probabilistic outputs is defined as a 6-tuple (S, s_i, X, Y, T, G) where

(i) S is a finite set of states $\{s_0, s_1, s_2, ...s_n\}$,
(ii) s_i is the initial state,
(iii) X is a set of input alphabets, e.g. $\{0, 1\}$,
(iv) Y is a set of output alphabets, e.g. $\{0, 1\}$,
(v) $T \subseteq S \times X \to S$ is a transition function from current state $s_i \in S$ and the input $x \in X$ to the next state $s_i' \in S$, and
(vi) $G \subseteq S \to P(Y)$ is a probabilistic output function that assigns a probability $P(Y|s_i)$ of producing the output $y \in Y$ at state $s_i \in S$.

Definition 2. A Moore machine with probablistic outputs (S, s_i, X, Y, T, G) is a linear Moore machine for binary input x provided,

(i) if the input $x = 1$, the state s_i makes a transition to s_{i+1} if $i < n$; otherwise, the state s_i does not change.
(ii) if input $x = 0$, the state s_i makes a transition to s_{i-1} if $i > 0$; otherwise, the state s_i does not change.

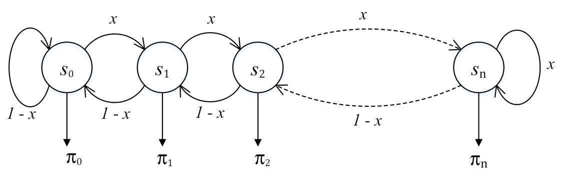

Fig. 1. The structure of a linear finite state machine.

A linear Moore machine is shown in Figure (1). Input to this state machine is a Bernoulli sequence with probability of '1' given by $P(x = 1)$. For compactness of notation, let us denote $P(x = 1)$ as $P(x)$. For a long enough Bernoulli sequence with fixed $P(x)$, the linear finite state machine will be in steady state. Since this steady state probability depends only on $P(x)$, let us denote it by $P(s_i|P(x))$.

When linear finite state machine is in equilibrium, the probability of leaving a state will be identical to the probability of entering that state

$$P(x)P(s_i|P(x)) = (1 - P(x))P(s_{i+1}|P(x)),$$

$$\Rightarrow P(s_i|P(x)) = \left(\frac{P(x)}{1 - P(x))}\right)^i P(s_0|P(x)). \qquad (1)$$

Now, since

$$P(s_0|P(x))+P(s_1|P(x))+P(s_2|P(x))+\cdots+P(s_n|P(x)) = 1,$$

By substituting $P(s_i|P(x))$ from equation (1) in the above equation, we get

$$P(s_0|P(x)) = \frac{1}{\sum\limits_{i=0}^{n}\left(\frac{P(x)}{1-P(x)}\right)^i}.$$

When machine is in state k, it outputs a Bernoulli stream in which probability of '1' is given by $P(Y = 1|s_k)$. Let us denote this output probability by π_k. Using total probability theorem, the overall probability of '1' in the output stream is given by

$$P(Y = 1|P(x)) = \sum_{i=0}^{n} \pi_i P(s_i|P(x)). \qquad (2)$$

III. Synthesis of Stochastic Computational Elements using Decision Procedures

Let $f(t)$ be our target function such that $f(t)\in[0, 1]$ where $t\in[0, 1]$. Since both $t\in[0, 1]$ and $P(x) \in [0, 1]$, therefore we can replace t by $P(x)$. Our objective is to find the output probabilities of linear finite state machine $\{\pi_0, \pi_1, \pi_2, ...\pi_n\}$ such that its steady state ouput $P(Y = 1|P(x))$ closely follows target function $f(P(x))$ for $0 < P(x) < 1$

$$P(Y = 1|P(x)) \approx f(P(x)).$$

Equation (2) is central to the synthesis of stochastic computational elements as described in algorithm (1). Let us define $\epsilon > 0$ as the maximum permitted error between $P(Y = 1|P(x))$ and the target function $f(P(x))$. Then, from equation (2),

$$f(P(x)) - \epsilon \leq \sum_{i=0}^{n} \pi_i P(s_i|P(x)) \leq f(P(x)) + \epsilon. \qquad (3)$$

Algorithm 1 Decision Procedure based Synthesis of Stochastic Computational Elements

```
(i)   Choose a small value of $\epsilon > 0$.
(ii)  Initialize a set of constraints, $C = \phi$.
(iii) Choose $N$ samples $\{P(x_1), P(x_2), P(x_3), \cdots P(x_N)\}$
      of $P(x)$ by sampling it uniformly between 0 and 1.
(iv)  $For\ each\ P(x_k)\{$
        • Compute $P(s_i|P(x_k))$ for all $s_i \in S$ using Eq. (1).
        • Compute $f(P(x_k))$.
        • Substitute $P(s_i|P(x_k))$ and $f(P(x_k))$ in
          (3) and create a constraint $c_k$ in terms of
          $\{\pi_0, \pi_1, \pi_2, ....\pi_n\}$.
        • Add $c_k$ to the set of constraints $C$
      }
(v)   Find the feasible region by solving the set of constraints
      $C$ using decision procedures .
(vi)  $if$ (feasible region is found) $then$
      $return\ \{\pi_0, \pi_1, \pi_2, ....\pi_n\}$,
      $else$
      Set $\epsilon := \frac{\epsilon}{0.99}$ and repeat all the steps from Step 2.
```

First step is the specification of error tolerance ϵ at sampled points. Next step is the creation of N samples of $P(x) \in [0, 1]$.

Substitution of each value of $P(x_k)$ in (3) gives a constraint in terms of output probabilities $\{\pi_0, \pi_1, \pi_2, ...\pi_n\}$. After creating N such constraints, we can solve them using decision procedures. The solution $\{\pi_0, \pi_1, \pi_2, ...\pi_n\}$ satisfying all the constraints is the set of output probabilities of linear finite state machine approximating the target function $f(t)$. If no such solution is found, we can relax our constraints by setting ϵ to slightly larger value and using decision procedures again for solving these new constraints. In next section, we show that the output probabilities obtained from decision procedures result in bounded error at unsampled values of $P(x)$.

Lemma 3. *(Boundedness of Error) The error between a target function and its estimate between sampled points can be made arbitrarily close to ϵ by increasing the sampling rate N.*

Proof: Error between the target function and its estimate is always less than ϵ at sampled points. If target function and its estimate are Lipschitz continuous, then the error between sampled points decreases as we increase the sampling rate. Let x be any of the N sampled points, let Δx be a small interval such that $\Delta x < \frac{1}{N}$, For compactness of notation, let us denote $P(Y=1|x)$ by $g(x)$, then we can rearrage inequality (3) as,

$$-\epsilon < f(x) - g(x) < \epsilon.$$

Since $\Delta x < \frac{1}{N}$, $x + \Delta x$ will be between sampled points. Let L_f be the absolute value of the Lipschitz constant of $f(x)$, then

$$f(x) - L_f \Delta x < f(x+\Delta x) < f(x) + L_f \Delta x, \tag{4}$$

Let L_p be the absolute value of Lipschitz constant of $P(Y=1|x) = g(x)$, then

$$g(x) - L_p \Delta x < g(x+\Delta x) < g(x) + L_p \Delta x, \tag{5}$$

Subtracting inequality (5) from inequality (4),

$$\begin{aligned} f(x) - g(x) - (L_p + L_f)\Delta x &< (f(x+\Delta x) - g(x+\Delta x)) \\ &< (f(x) - g(x)) + (L_p + L_f)\Delta x, \end{aligned}$$

By using the inequality (3),

$$-\epsilon - (L_p + L_f)\Delta x < f(x+\Delta x) - g(x+\Delta x) < \epsilon + (L_p + L_f)\Delta x,$$

Substituting $\Delta x < \frac{1}{N}$ in the above inequality, we get

$$-\epsilon - \frac{(L_p + L_f)}{N} < f(x+\Delta x) - g(x+\Delta x) < \epsilon + \frac{(L_p + L_f)}{N}. \tag{6}$$

It is clear from (6) that for a given ϵ, we can make the error arbitrarily small by increasing the sampling rate and making Δx arbitrarily small. ■

Lipschitzness of Estimated Function

The synthesized function $P(Y=1|P(x))$ given by equation (2) is Lipschitz continuous if steady state probabilities $P(s_i|P(x))$ of all states $s_i \in S$ are also Lipschitz continuous. Steady state probability of state s_i is given by,

$$P(s_i|P(x)) = \frac{\left(\frac{P(x)}{1-P(x))}\right)^i}{\sum\limits_{k=0}^{n} \left(\frac{P(x)}{1-P(x))}\right)^k} = \frac{r^i}{\sum\limits_{k=0}^{n} r^k},$$

where $r = \frac{P(x)}{1-P(x))}$ and it changes from 0 to ∞ when x changes from 0 to 1. Using chain rule, the derivative of $P(s_i|P(x))$ with respect to $P(x)$ is given by,

$$P'(s_i|P(x)) = \frac{r^{i-1}(r+1)^2 \sum\limits_{k=0}^{n}(i-k)r^k}{(\sum\limits_{k=0}^{n} r^k)^2}. \tag{7}$$

$P'(s_i|P(x))$ is bounded for $0 < r < \infty$, which means $P(s_i|P(x))$ is Lipschitz continuous for all i. Therefore the synthesized function $P(Y=1|P(x))$, which is the weighted sum of $P(s_i|P(x))$, is also Lipschitz continuous.

IV. Experimental Results

We have approximated polynomial, tanh and exponential functions by synthesizing linear finite state machines using algorithm (1). Each of these functions is sythesized by uniformly sampling 10^3 points of $P(x)$ in the range [0,1]. We have compared the accuracy of our machines with those of Li et al [6].

A. Polynomial Function

We have sythesized the third degree polynomial of equation (8) on a four state linear machine shown in figure (2). The output of this machine is shown in figure (3).

$$P(Y=1|P(x)) = \frac{1}{4} + \frac{9}{8}P(x) - \frac{15}{8}P^2(x) + \frac{5}{4}P^3(x). \tag{8}$$

For this function, Li's [6] four state linear machine has the worst-case error of ***0.024***, which is ***1.65*** times larger than our approach ***0.0145***.

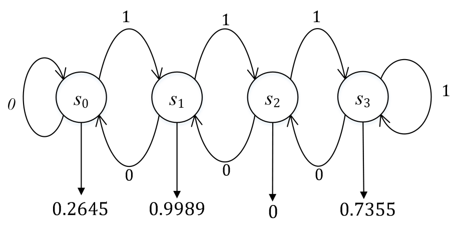

Fig. 2. A 4 state linear finite state machine approximating the polynomial function. Our design provides 1.65 times smaller worst-case error than the state-of-the-art.

B. Stochastic tanh function

Stochastic tanh function can be approximated by the following equation,

$$f(x) \approx \frac{e^{\frac{N}{2}x} - e^{-\frac{N}{2}x}}{e^{\frac{N}{2}x} + e^{-\frac{N}{2}x}}.$$

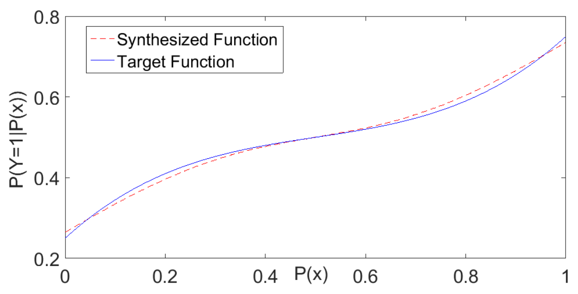

Fig. 3. Comparison of the stochastic approximation of polynomial function and target polynomial function. Our design provides 1.65 times smaller worst-case error than the state-of-the-art.

where 'N' is the number of states of linear finite state machine used for the synthesis of '$tanh$' function. Here $f(x)$ and 'x' are represented using the bipolar coding format [4]. To synthesize this function, we have converted it into unipolar coding format resulting in the folllowing equation,

$$f(x) = \frac{e^{8(2P(x)-1)}}{e^{8(2P(x)-1)}+1}. \tag{9}$$

This function was synthesized on the eight state linear machine shown in figure (4), the output of this machine is compared with equation (9) in figure (5).

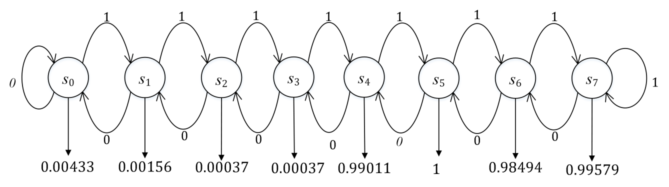

Fig. 4. A linear finite state machine approximating the tanh function using 8 states. Our design provides 1.625 times smaller worst-case error than the state-of-the-art

For this function, Peng Li's approach [6] resulted in worst case error of ***0.0065***, while worst case error using our approach is ***0.04***, which is ***1.625*** times smaller than their approach.

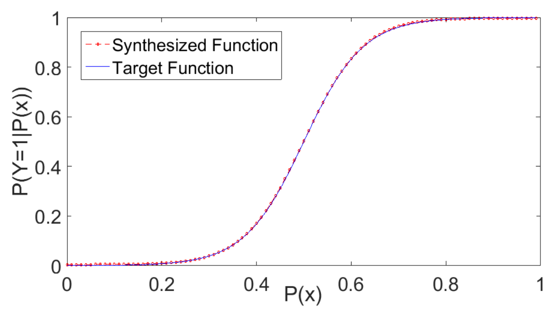

Fig. 5. Comparison of the stochastic approximation of tanh function and the target tanh. Our design provides 1.625 times smaller worst-case error than the state-of-the-art.

C. Exponentiation function

We have synthesized the exponentiation function of equation (10) on a 16-state linear finite state, the output of this machine is shown in figure (6).

$$P(Y=1|P(x)) = \begin{cases} 1, & 0 \leq P(x) \leq 0.5 \\ e^{-4(2P(x)-1)}, & 0.5 \leq P(x) \leq 1 \end{cases} \tag{10}$$

For this function, Li's 16-state machine had the worst case error of 0.125 [6]. Again, our approach produced smaller (***1.17*** times improvement) worst case error as compared to Li at el [6].

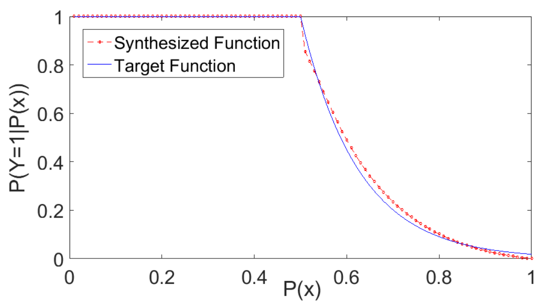

Fig. 6. Comparison of the our stochastic approximation of exponential function and the target exponential. Our design provides 1.17 times smaller worst-case error than the state-of-the-art.

V. Conclusion

We have proposed a decision procedures based algorithm for the synthesis of stochastic computational elements. We have tested it on three different classes of functions. Our algorithm guarantees an upper bound on the error. Using this approach we have synthesized approximations for polynomial, exponentiation and tanh functions using linear finite state machines. Our approach resulted in 1.17 to 1.65 times smaller errors for these functions as compared to the previous state-of-the-art approach [6].

References

[1] Chen, Te-Hsuan, Armin Alaghi, and John P. Hayes. "Behavior of stochastic circuits under severe error conditions", IT-Information Technology 56.4 (2014): 182-191.

[2] J. Von Neumann "Probabilistic logics and the synthesis of reliable organisms from unreliable components", Automata Studies, no. 34, pp.43 -99.

[3] E. F. Moore and C. E. Shannon "Reliable circuits using less reliable relays", J. Franklin Inst., vol. 262, pp.191 -297 1956.

[4] B.R. Gaines, "Stochastic Computing Systems", Advances in Information Systems Science, J.F. Tou, ed., vol. 2, chapter 2, pp. 37-172, New York: Plenum, 1969.

[5] Brown, B.D.; Card, H.C., "Stochastic neural computation. I. Computational elements", in Computers, IEEE Transactions on , vol.50, no.9, pp.891-905, Sep 2001 doi: 10.1109/12.954505.

[6] Peng Li; Weikang Qian; Riedel, M.D.; Bazargan, K.; Lilja, D.J., "The synthesis of linear Finite State Machine-based Stochastic Computational Elements", in Design Automation Conference (ASP-DAC), 2012 17th Asia and South Pacific , vol., no., pp.757-762, Jan. 30 2012-Feb. 2 2012 doi: 10.1109/ASPDAC.2012.6165056.

[7] Weikang Qian; Riedel, M.D., "The synthesis of robust polynomial arithmetic with stochastic logic", in Design Automation Conference, 2008. DAC 2008. 45th ACM/IEEE , vol., no., pp.648-653, 8-13 June 2008.