Hardware ODE Solvers using Stochastic Circuits

Siting Liu and Jie Han
Department of Electrical and Computer Engineering
University of Alberta
9211-116 Street NW
Edmonton, Alberta T6G 1H9
{siting2,jhan8}@ualberta.ca

ABSTRACT

A novel ordinary differential equation (ODE) solver is proposed by using a stochastic integrator to implement the accumulative function of the Euler method. We show that a stochastic integrator is an unbiased estimator for a Euler numerical solution. Unlike in conventional stochastic circuits, in which long stochastic bit streams are required to produce a result with a high accuracy, the proposed stochastic ODE solver provides an estimate of the solution for every bit in the stochastic bit stream, thus significantly reducing the latency and energy consumption of the circuit. Complex ODE solvers are constructed for solving nonhomogeneous ODEs, systems of ODEs and higher-order ODEs. Experimental results show that the stochastic ODE solvers provide very accurate solutions compared to their binary counterparts, with on average an energy saving of 46% (up to 74%), 8× throughput per area (up to nearly 12×) and a runtime reduction of 72% (up to 82%).

CCS CONCEPTS

• **Hardware** → *Finite state machines*; *Hardware accelerators*; • **Mathematics of computing** → *Ordinary differential equations*; • **Theory of computation** → *Stochastic approximation*; *Numeric approximation algorithms*; Random walks and Markov chains;

KEYWORDS

stochastic integrator, ordinary differential equation, stochastic computing

ACM Reference format:
Siting Liu and Jie Han. 2017. Hardware ODE Solvers using Stochastic Circuits. In *Proceedings of the 54th Annual Design Automation Conference, Austin, TX, USA, June 18 - 22, 2017 (DAC'17),* 6 pages.
DOI: http://dx.doi.org/10.1145/3061639.3062258

1 INTRODUCTION

Ordinary differential equations (ODEs) are widely used in the modeling of natural processes in physics, chemistry and biology, as well as in solving problems in many engineering and social studies such as scientific computing and economics. Various algorithms have been developed for solving an ODE. However, most of the algorithms are computationally intensive, especially when solving problems for a large scale system. While a general-purpose processor is often used, acceleration has been achieved by using the massively parallel structure of graphics processing units (GPUs) [7, 9]. However, a GPU is not well suited for mobile and embedded applications because it is large and power hungry compared to a tailored ASIC design.

Conventional digital ODE solvers require the use of adders, multipliers, registers and complex control circuitry, such as in a digital differential analyzer (DDA) [14]. Although more efficient implementations have been proposed by using concurrent processing paths [2], a conventional binary design still requires a complex datapath and control circuitry.

Stochastic computing (SC) is a different computing paradigm that uses random binary bit streams to encode a real value and digital logic to perform computation [5]. In the unipolar representation of SC, a number s in $[0, 1]$ is compared with a uniformly distributed random number in $[0, 1]$ to obtain one bit in a bit stream. The probability of a single bit being "1" is then s. The digital implementation of a stochastic number generator (SNG) is shown in Figure 1, where both s and the random number are in n-bit binary format. The bipolar representation encodes a number s in $[-1, 1]$ by a linear mapping $(s + 1)/2$ to $[0, 1]$.

A number of SC applications have been proposed, including function generation [15], LDPC decoders [17], and learning machines [10]. An SC circuit has the unique features of simple hardware and high fault tolerance. However, it often requires long stochastic bit streams to achieve a high accuracy, which results in a long latency in SC [8]. Subsequently, this long latency has a negative impact on the energy efficiency of a stochastic circuit. Recently, low discrepancy (LD) sequences have been used to replace conventional linear feed back shift register (LFSR)-generated pseudorandom sequences for reducing the latency and energy of a stochastic circuit [1, 12].

A stochastic integrator was originally proposed in [5] as a sequential SC element. A stochastic integrator consists of two components: an n-bit or 2^n-state up/down counter and an SNG. The counter can be modeled by a finite state machine (FSM) of 2^n states and the SNG generates a stochastic bit stream encoding the probability

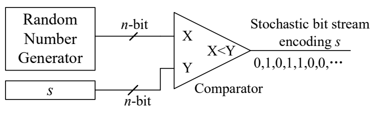

Figure 1: A stochastic number generator (SNG).

for the value stored in the counter. The stochastic bit stream is then employed to compute complex functions such as high-order polynomials, logarithm and trigonometric functions [13] and perform division [3]. Nevertheless, the up/down counter inside the stochastic integrator carries useful information, which is often overlooked. Gaines showed that a network of stochastic integrators with feedback, called adaptive digital elements or ADDIEs, can solve Laplace's equations and serve as a Bayesian estimator/predictor [5].

In this paper, a novel ODE solver is proposed by using SC circuits. Detailed formulation shows that a single stochastic ODE solver provides an unbiased estimate of the Euler numerical solution for an ODE. Three error reduction schemes are further proposed and verified by both theory and simulation. The approach to building more complex ODE solvers is demonstrated by constructing the circuits for solving three typical ODEs. With a high accuracy, the proposed stochastic ODE solvers show significant advantages in energy consumption, throughput and performance, compared with their binary counterparts.

2 PROPOSED STOCHASTIC ODE SOLVERS

2.1 Formulation

The circuit diagram and a symbol of the stochastic integrator are shown in Figure 2 [5]. A key component in a stochastic integrator is a 2^n-state up/down counter. A random number generator (RNG) and a comparator work as an SNG for generating a stochastic bit stream, Seq_{out}, to encode the value stored in the counter. The bit stream can be used as a feedback for itself or as an input bit stream for subsequent stochastic integrators. If not used, the RNG and the comparator can be removed to reduce hardware cost. The input signals A and B carry stochastic bit streams, which determine whether to increase, decrease or keep the value of the counter.

The counter is typically a 2^n-state counter with n-bit width, counting from 0 to $2^n - 1$. The initial value is determined by the input of the application. Let I denote the integer value stored in the counter. The probability to be encoded by the output stochastic bit stream is $I/2^n$ in the unipolar representation or $2 \times I/2^n - 1$ in the bipolar representation. Due to the space limitation, only the unipolar representation is considered in this paper; designs for the bipolar representation can similarly be derived by a linear mapping.

Let the two bits in the input streams A and B be a_i and b_i respectively at the ith clock cycle. The function of the up/down counter is then

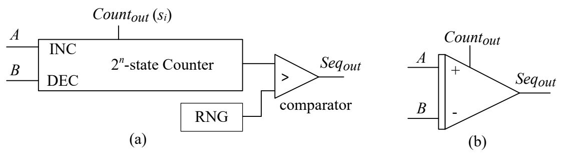

Figure 2: A stochastic integrator: (a) the circuit block diagram; (b) a symbol.

$$I_{i+1} = \begin{cases} I_i + 1 & \text{if } a_i = 1 \text{ and } b_i = 0 \\ I_i & \text{if } a_i = b_i \\ I_i - 1 & \text{if } a_i = 0 \text{ and } b_i = 1 \end{cases}, \quad (1)$$

where I_i and I_{i+1} are the integers stored in the counter at the ith and $(i+1)$th clock cycles. Equivalently, we have

$$I_{i+1} = I_i + a_i - b_i. \quad (2)$$

The expectation of I_{i+1}, $\mathbb{E}[I_{i+1}]$, is given by

$$\mathbb{E}[I_{i+1}] = \mathbb{E}[I_i + a_i - b_i] = \mathbb{E}[I_i] + \mathbb{E}[a_i] - \mathbb{E}[b_i]. \quad (3)$$

Let the probability that a_i is "1" at the ith clock cycle be $p_{a,i}$ and the probability that b_i is "1" be $p_{b,i}$, (3) becomes [16]:

$$\mathbb{E}[I_{i+1}] = \mathbb{E}[I_i] + p_{a,i} - p_{b,i}. \quad (4)$$

To convert an integer into a stochastic number in the unipolar representation, both sides of (4) are normalized by $\frac{1}{2^n}$. Hence, (4) is transformed to

$$\mathbb{E}[s_{i+1}] = \mathbb{E}[s_i] + \frac{1}{2^n}(p_{a,i} - p_{b,i}), \quad (5)$$

where $s_i = I_i/2^n$ and $s_{i+1} = I_{i+1}/2^n$. If the initial value in the counter is s_0, then by an iterative accumulation of (5), the expected value of s_k at the (arbitrary) kth clock cycle ($k = 1, 2, \dots$) is obtained as

$$\mathbb{E}[s_k] = s_0 + \frac{1}{2^n}\sum_{i=0}^{k-1}(p_{a,i} - p_{b,i}). \quad (6)$$

For an ODE $\frac{dy(t)}{dt} = f(t, y(t))$, the numerical solution for a given t can be estimated by considering the derivative of $y(t)$ at a discrete t_i as

$$\frac{\mathbf{d}y(t)}{\mathbf{d}t}|_{t=t_i} = \lim_{\Delta t \to 0} \frac{y(t_i + \Delta t) - y(t_i)}{\Delta t} \approx \frac{y(t_i + h) - y(t_i)}{h} \quad (7)$$

where h is a small value for the time interval Δt. Let $t_{i+1} = t_i + h$ ($t_i = h \cdot i$ when $t_0 = 0$ and h is a constant, $i = 0, 1, 2, \dots$), (7) leads to the solution by the Euler method [4]:

$$\hat{y}_{i+1} = y_i + hf(t_i, y_i), \quad (8)$$

where $\hat{y}_{i+1}$ is the numerical estimation of the function value of $y(t)$ at t_{i+1}, i.e., $y(t_{i+1})$, and h is the step size for the estimate. Let t start from $t = 0$, with $h = \frac{1}{2^n}$ and $f(t, y(t)) = p_a(t) - p_b(t)$, (8) is simplified to

$$\hat{y}_{i+1} = y_i + \frac{1}{2^n}[p_a(\frac{i}{2^n}) - p_b(\frac{i}{2^n})]. \quad (9)$$

If the initial condition of the ODE is y_0, the estimate of the solution at the kth step is given by an iterative accumulation of (9) over i, which leads to

$$\hat{y}_k = y_0 + \frac{1}{2^n}\sum_{i=0}^{k-1}[p_a(\frac{i}{2^n}) - p_b(\frac{i}{2^n})]. \quad (10)$$

Hence, for the ODE

$$\frac{\mathrm{d}y(t)}{\mathrm{d}t} = p_a(t) - p_b(t), \tag{11}$$

the Euler numerical solution $\hat{y}_k$ provides an estimated value of the function $y(t)$ at $t = k/2^n$, $k = 0, 1, 2, \ldots$, i.e.,

$$y(\frac{k}{2^n}) \approx \hat{y}_k. \tag{12}$$

Let the input sequences of the stochastic integrator encode the probabilities $p_a(t)$ and $p_b(t)$ at $t = i/2^n$, i.e., $p_{a,i} = p_a(\frac{i}{2^n})$ and $p_{b,i} = p_b(\frac{i}{2^n})$; as per (6) and (10), the normalized expected value of the up/down counter at the kth clock cycle, $\mathbb{E}[s_k]$, provides an unbiased estimate of the Euler solution at the kth time step, $\hat{y}_k$, for the same initial condition, i.e. $y_0 = s_0$. By (12), we obtain

$$y(\frac{k}{2^n}) \approx \mathbb{E}[s_k], \tag{13}$$

that is, $\mathbb{E}[s_k]$ ($k = 1, 2, \ldots$) provides an approximate solution of the ODE (11) with a step size of $1/2^n$.

The input bit streams only serve as the control signals for the counter, while the output of the counter provides the Euler estimate, one estimate at each time step or per clock cycle, thus achieving great efficiency.

2.2 Stochastic ODE Solver Designs

In this section, several designs are proposed as typical ODE solvers. Note that all the ODEs and parameters are chosen so that the solution lies in the range $[0, 1]$ of the stochastic unipolar representation; otherwise, it is considered that an overflow occurs. To evaluate accuracy, the analytical solutions are also obtained for comparison.

2.2.1 Nonhomogeneous ODEs. Nonhomogeneous ODEs refer to the type of ODEs that involves time, i.e., using t-related terms.

As per (11) and (13), if $p_a(t) = 1$ and $p_b(t) = 0$, a stochastic integrator solves the ODE

$$\frac{\mathrm{d}y(t)}{\mathrm{d}t} = 1 - 0, \tag{14}$$

with $y(0) = 0$, i.e., the counter is initialized to "0". The analytical solution for (14) is $y(t) = t$. It is produced by the output of the counter, as shown in the stochastic integrator at the first stage in Figure 3. If the output sequence from the stochastic integrator is connected to a subsequent stochastic integrator (with 0 as another input), the cascaded structure, as shown at the first two stages in Figure 3, solves the ODE

$$\frac{\mathrm{d}y(t)}{\mathrm{d}t} = t, \tag{15}$$

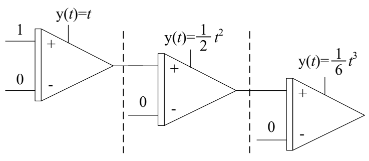

Figure 3: A stochastic ODE solver for (14), (15) and (16).

with $y(0) = 0$. The analytical solution for (15) is $y(t) = (1/2)t^2$, estimated in the circuit by the output of the second counter. Similarly, three stages of the cascaded structure solves

$$\frac{\mathrm{d}y(t)}{\mathrm{d}t} = \frac{1}{2}t^2, \tag{16}$$

with $y(0) = 0$. The analytical solution for (16) is $y(t) = (1/6)t^3$, estimated by the output of the third counter.

Hence, the cascaded stochastic integrators in Figure 3 are solvers for a set of nonhomogeneous ODEs. In the cascaded structure, the integrators in the earlier stages are used to generate the stochastic bit stream for the t-related terms in the ODE to be solved.

The cascading of the stochastic integrators can be continued for solving an ODE with a higher-order polynomial as its solution. Note that the output sequence of the integrator at the last stage is not connected to any other components, so the RNG and comparator in this stochastic integrator can be removed.

2.2.2 Systems of ODEs. A system of ODEs can be solved by using multiple stochastic integrators. For a system of ODEs such as

$$\begin{cases} \frac{\mathrm{d}y_1(t)}{\mathrm{d}t} = y_2(t) - 0.5, \\ \frac{\mathrm{d}y_2(t)}{\mathrm{d}t} = 0.5 - y_1(t), \end{cases} \tag{17}$$

with $y_1(0) = 0$ and $y_2(0) = 0.5$ as the initial values of the two counters. For this system, two cross-coupled stochastic integrators provide a solution by utilizing the output bit stream from one integrator as an input of the other integrator, as shown in Figure 4. In this design, the other inputs are set to 0.5, as determined by (17). The analytical solution for (17) is $y_1(t) = 0.5 - 0.5\cos(t)$ and $y_2(t) = 0.5 + 0.5\sin(t)$. Other values other than 0.5 can be used in (17) if they do not result in an overflow.

2.2.3 Higher-order ODEs. To solve an mth order ODE, at least m stochastic integrators are required since one stochastic integrator performs a single integration. For a second order ODE such as

$$\frac{\mathrm{d}^2y(t)}{\mathrm{d}t} + 2\frac{\mathrm{d}y(t)}{\mathrm{d}t} + y(t) = 0, \tag{18}$$

with $y(0) = 0$, $\frac{\mathrm{d}y(t)}{\mathrm{d}t}|_{t=0} = 1$, an additional function, $z(t) = \frac{\mathrm{d}y(t)}{\mathrm{d}t} + 2y(t)$, is introduced for the first two terms in (18) such that $\frac{\mathrm{d}z(t)}{\mathrm{d}t} = \frac{\mathrm{d}^2y(t)}{\mathrm{d}t} + 2\frac{\mathrm{d}y(t)}{\mathrm{d}t}$. By doing so, the order of the ODE is lowered and (18) is converted into two first-order ODEs

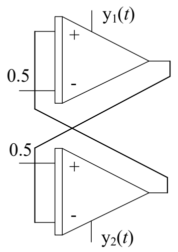

Figure 4: A stochastic ODE solver for (17).

$$\begin{cases} \frac{\mathrm{d}z(t)}{\mathrm{d}t} = -y(t), \\ \frac{\mathrm{d}y(t)}{\mathrm{d}t} = z(t) - 2y(t). \end{cases} \tag{19}$$

In Figure 5, the first stochastic integrator solves the first ODE in (19) by initializing the counter to $z(0) = \frac{\mathrm{d}y(t)}{\mathrm{d}t}|_{t=0} + 2y(0) = 1$. The inverter serves as a stochastic subtractor that computes the function $1 - y(t)$. The second stochastic integrator solves the second ODE by initializing the counter to $y(0) = 0$.

Due to the $\times 2$ factor for $y(t)$, the updating rule of the second stochastic integrator becomes: $I_{i+1} = I_i + a_i - 2b_i$, where a_i and b_i are two bits of the input stochastic bit streams at the ith clock cycle. The counter is modified to take multiple bits as inputs for the stochastic sequence of $2b_i$.The analytical solution for (18) is $y(t) = te^{-t}$; $z(t) = (t+1)e^{-t}$ is computed by the first stochastic integrator as an intermediate result.

3 ERROR REDUCTION SCHEMES

One major disadvantage in SC is the loss of accuracy [8]. Usually, it is believed that multiple independent RNGs need to be utilized for a higher accuracy, which inevitably increases hardware overhead. However, we show here that sharing RNGs can reduce the variance in the solution of a stochastic ODE solver, thus reducing error.

The error in the solution of a stochastic ODE solver is mainly caused by : (1) the Euler numerical method and (2) the randomness in SC.

In the Euler method, the error is measured by the local truncation error (LTE) and global truncation error (GTE). The LTE refers to the error introduced in a single step of estimation and the GTE refers to the error caused in multiple steps. The LTE and GTE are proportional to h^2 and h respectively, where h is the step size [4]. Thus a simple solution is to increase the bit width of the up/down counter to reduce the step size ($h = 1/2^n$, where n is the bit width of the counter), thereby reducing error due to the Euler method. Note that increasing the bit width of the counter leads to a better granularity in the final solution, but it does not significantly increase the latency of the stochastic circuit, unlike in conventional SC.

The error introduced by SC is related to the variance of the derivative term, $a_i - b_i$. When independent RNGs are used to generate a_i and b_i, the variance at a single step is given by:

$$\mathrm{Var}[a_i - b_i] = p_{a,i}(1 - p_{a,i}) + p_{b,i}(1 - p_{b,i}). \tag{20}$$

The total variance is the sum of variances at each step if each random number is independently generated, as is approximately the case for using an LFSR.

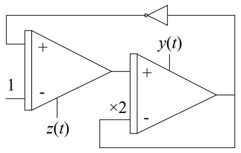

Figure 5: A stochastic ODE solver for (18).

However, if the same RNG is used to generate A and B, the probability distribution of $a_i - b_i$ is shown in Table 1 (assume $p_{a,i} \geq p_{b,i}$). In this case, the variance of $a_i - b_i$ can be derived as:

$$\begin{aligned} \mathrm{Var}_s[a_i - b_i] &= \mathbb{E}[(a_i - b_i - \mathbb{E}(a_i - b_i))^2] \\ &= (p_{a,i} - p_{b,i})(1 - p_{a,i} + p_{b,i}). \end{aligned} \tag{21}$$

Because $\mathrm{Var}_s[a_i - b_i] - \mathrm{Var}[a_i - b_i] = -2p_{b,i}(1 - p_{a,i}) \leq 0$, we obtain $\mathrm{Var}_s[a_i - b_i] \leq \mathrm{Var}[a_i - b_i]$ for any $i = 0, 1, 2, \ldots$. Therefore, sharing the use of RNGs to generate input stochastic bit streams improves the accuracy. The same conclusion can similarly be obtained for $p_{a,i} < p_{b,i}$.

Further improvement of the accuracy can be achieved by using LD sequences for a faster convergence and thus better progressive precision [1, 6].

4 EXPERIMENTS AND RESULTS

4.1 Validation of the Proposed Designs

In this section, the proposed stochastic ODE solvers are validated by hardware simulations using VHDL. The designs are synthesized by Synopsys Design Compiler and analyzed by Mentor Graphic ModelSim with the STM 28nm technology library. The same temperature and supply voltage are used in all simulations. The numerical solution is produced by using 8-bit counters with a step size of $1/2^8$.

Figure 6 shows the results produced by the circuit in Figure 3 for solving (14), (15) and (16), in comparison with the analytical results. Note that for $t > 1$, the result is not shown as it exceeds the range of the unipolar representation in SC. The simulation results are depicted in Figure 7 for two full periods along with the analytical solution for (17). A full period of the sine/cosine function can be generated within 1609 clock cycles ($\lceil 2 \times \pi \times 2^8 \rceil$), while a conventional SC function generation method requires 1024-bit sequences to produce a single result [13]. Thus, a stochastic ODE solver can be used as an efficient function generator. The simulation results produced by the circuit in Figure 5 are depicted in Figure 8, in comparison with the analytical results.

As seen from the results, the 8-bit stochastic ODE solvers produce very accurate solutions when compared to the analytical results. A quantitative evaluation of the results using the root mean square error (RMSE) is reported next.

4.2 Validation of the Error Reduction Schemes

The accuracy of the stochastic ODE solvers with different configurations is measured to verify the three error reduction schemes by: (1) increasing the bit width; (2) sharing the use of RNGs; (3) using LD Sobol sequences [12]. The proposed design in Figure 4

Table 1: Probability distribution of $a_i - b_i$ when the same RNG is used to generate A and B

$a_i - b_i$	Probability
-1	0
0	$1 - p_{a,i} + p_{b,i}$
1	$p_{a,i} - p_{b,i}$

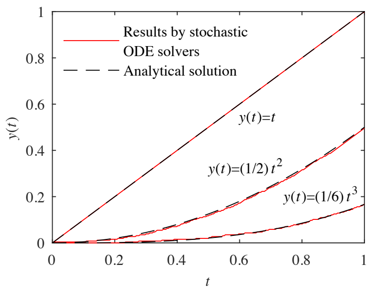

Figure 6: Analytical solution vs. hardware results produced by the stochastic ODE solvers for (14), (15) and (16).

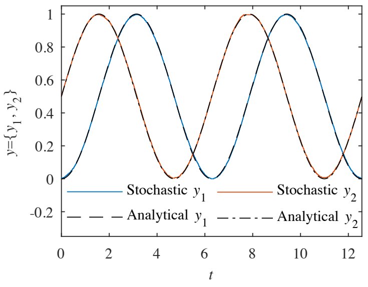

Figure 7: Analytical solution vs. hardware results produced by the stochastic ODE solver for (17).

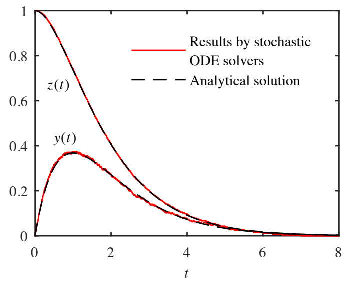

Figure 8: Analytical solution vs. hardware results produced by the stochastic ODE solver for (18). $z(t)$ is the intermediate result.

is considered for an RMSE analysis in the first full period of the functions $y_1(t)$ and $y_2(t)$. The results are shown in Figure 9.

As can be seen, the circuit with a larger width tends to have a lower RMSE. In general, the circuits using LD sequences produce more accurate results than those using pseudorandom (PR) sequences generated by LFSRs. For the same bit width, the circuits using PR sequences with shared RNGs provide more accurate numerical solutions than those using independent RNGs. When LD sequences are used, the RMSE is not significantly affected by using shared RNGs. Nevertheless, LD sequences are adopted with

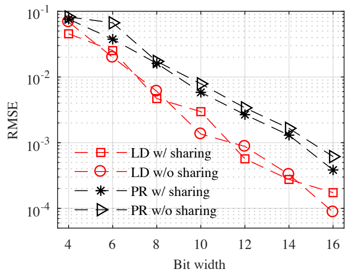

Figure 9: RMSE of $y_1(t)$ and $y_2(t)$ for the stochastic ODE solver in Figure 4 under different configurations.

the RNG-sharing scheme for reducing hardware cost and energy consumption [12].

4.3 Performance Evaluation

The accuracy and hardware cost of the stochastic ODE solvers are evaluated and compared with their binary counterparts. The binary circuits are implemented by using a second-order Runge Kutta (RK2) (midpoint) numerical method [4]. The RK2 method is also known as a modified Euler method with GTE in $O(h^2)$ and LTE in $O(h^3)$. The numerical solution of an ODE in the form of $\frac{dy(t)}{dt} = f(y(t), t)$ using the RK2 method is given by

$$\hat{y}_{i+1} = y_i + hf(t_i + \frac{h}{2}, y_i + \frac{h}{2}f(t_i, y_i)), \tag{22}$$

where h is the step size of the estimate. The bit width of the stochastic ODE solver is set to 8, so that h is $1/2^8$ for the stochastic ODE solver. The binary ODE solvers also employ 8-bit designs with a step size of $1/2^8$ for comparison. The RK2 method performs iterative additions and multiplications. For a step size of $1/2^8$, however, the multiplication can be simplified by shifting. Thus, a binary RK2 ODE solver can be implemented by iterative shifting and additions [11].

For the stochastic solvers, the RNG in a stochastic integrator is implemented by the simplest Sobol sequence generator, a reversely mapped counter [1]. The RNG is shared to reduce the hardware cost. As a result, only one RNG is required to generate stochastic bit streams for each design.

The accuracy of the hardware ODE solvers is measured by RMSE. The hardware efficiency is measured by energy per operation (EPO), throughput per area (TPA) and minimum runtime. The EPO is the total energy consumed for completing one estimate of the solution, obtained as the power multiplied by the clock period. The TPA is the maximum throughput per area and per time unit, given by $1/(\text{area} \times \text{critical path delay})$. The minimum runtime is obtained by multiplying the critical path delay and the number of clock cycles needed to obtain the solutions in Figures 6, 7 and 8 respectively. The power, area and critical path delay are first measured to compute the EPO, TPA and minimum runtime.

As shown in the simulation results in Table 2, all stochastic ODE solvers have a smaller EPO, minimum runtime and a larger TPA than their binary counterparts. The stochastic ODE solver achieves

Table 2: Hardware performance comparison

ODE	Metric	SC	RK2	Improvement
(15)	EPO (fJ)	144.49	201.05	28%
	TPA ($w/\mu s/\mu m^2$)	13.84	3.86	258%
	Runtime (ns)	104.96	263.68	60%
(16)	EPO (fJ)	186.10	253.05	26%
	TPA ($w/\mu s/\mu m^2$)	9.76	0.94	934%
	Runtime (ns)	104.96	586.24	82%
(17)	EPO (fJ)	201.21	466.00	56%
	TPA ($w/\mu s/\mu m^2$)	4.75	0.58	716%
	Runtime (ns)	2573.59	8557.20	70%
(18)	EPO (fJ)	156.04	591.62	74%
	TPA ($w/\mu s/\mu m^2$)	5.68	0.44	1184%
	Runtime (ns)	1597.44	6819.84	76%

up to 74% in energy saving, up to nearly 12× throughput enhancement and up to 82% drop in runtime. On average, the stochastic designs achieve an energy saving of 46%, 7× TPA improvement and a 72% reduction of runtime due to the smaller delay and area. Due to the random fluctuations in an SC circuit, the RMSE is larger for a stochastic ODE solver than for a binary ODE solver, as shown in Table 3. However, the RMSE is in the order of 10^{-3}, which indicates that the accuracy obtained by a stochastic ODE solver is quite high.

Figure 10 shows the accuracy and hardware efficiency of the design in Figure 4, compared to its binary counterpart with various bit widths and step sizes. In Figure 10(a), the stochastic design shows a smaller RMSE when the bit width is 4. However, the accuracy of a binary design becomes higher for a larger bit width. When a similar RMSE is achieved (e.g., for 8-bit binary and 10-bit stochastic designs), the stochastic design shows considerable advantages in EPO and TPA over the binary design, despite with a slightly longer runtime.

5 CONCLUSIONS AND FUTURE WORK

In this paper, a novel circuit design for solving an ODE is proposed by using a stochastic integrator to implement the Euler numerical method. The integrator provides an unbiased estimate of the Euler numerical solution. The stochastic ODE solver produces one estimate of the solution for every bit in the stochastic bit stream, so it is very efficient compared to conventional stochastic designs. The stochastic solvers are constructed for three types of ODEs and the solutions are validated. Sharing the use of RNGs is effective in reducing the error for pseudorandom sequences while it is less effective for deterministic LD sequences. The solutions for the proposed designs are currently limited to [0, 1] in the unipolar representation (or [-1, 1] in the bipolar representation). Extended range of representations for the solutions will be investigated in future work.

Table 3: Accuracy comparison (RMSE in 10^{-3})

ODE	SC	RK2
(15)	5.66	3.70
(16)	3.88	2.37
(17)	4.69	3.59
(18)	5.86	2.11

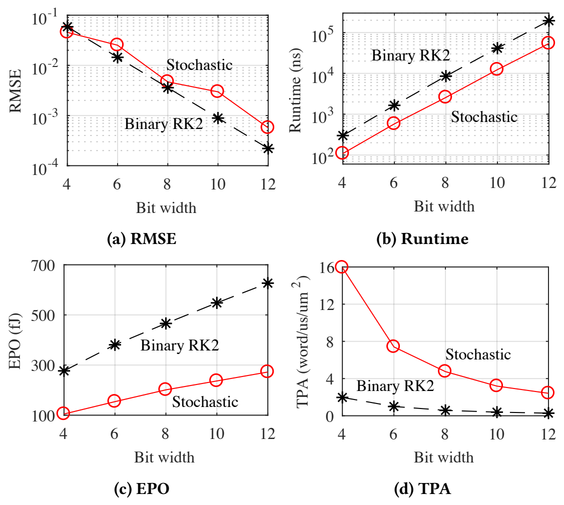

Figure 10: Comparison of stochastic and binary ODE solvers with different bit widths.

REFERENCES

[1] A. Alaghi and J. P. Hayes. 2014. Fast and Accurate Computation using Stochastic Circuits. In *DATE*.
[2] W.H.S. Bong. 2002. Digital Differential Analyzer. (April 23 2002). https://www.google.com/patents/US6377265 US Patent 6,377,265.
[3] B. D. Brown and H. C. Card. 2001. Stochastic Neural Computation. I. Computational Elements. *IEEE Trans. on Computers* 50, 9 (Sep 2001), 891–905.
[4] J. C. Butcher. 1987. *The Numerical Analysis of Ordinary Differential Equations: Runge-Kutta and General Linear Methods*. Wiley-Interscience.
[5] B. R. Gaines. 1969. *Stochastic Computing Systems*. Springer US, 37–172.
[6] M. Gerber and N. Chopin. 2015. Sequential Quasi Monte Carlo. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 77, 3, 509–579.
[7] M. Harris. 2004. Fast Fluid Dynamics Simulation on the GPU. *GPU gems* 1, 637–665.
[8] J. P. Hayes. 2015. Introduction to Stochastic Computing and Its Challenges. In *DAC*.
[9] M. Januszewski and M. Kostur. 2010. Accelerating Numerical Solution of Stochastic Differential Equations with CUDA. *Computer Physics Communications* 181, 1, 183 – 188.
[10] K. Kim, J. Kim, J. Yu, J. Seo, J. Lee, and K. Choi. 2016. Dynamic Energy-Accuracy Trade-off using Stochastic Computing in Deep Neural Networks. In *DAC*.
[11] B. R. Land. 2006. DDA on FPGA. http://people.ece.cornell.edu/land/courses/ece5760/DDA/index.htm. Accessed: 2016-10-12.
[12] S. Liu and J. Han. 2017. Energy Efficient Stochastic Computing with Sobol Sequences. In *DATE*.
[13] Y. Liu and K. K. Parhi. 2016. Computing Complex Functions Using Factorization in Unipolar Stochastic Logic. In *GLSVLSI*.
[14] K. Olynyk. 2003. System and Method for Improved Digital Differential Analyzer. https://www.google.ca/patents/US6510442 US Patent 6,510,442.
[15] W. Qian, X. Li, M. D. Riedel, K. Bazargan, and D. J. Lilja. 2011. An Architecture for Fault-tolerant Computation with Stochastic Logic. *IEEE Trans. on Computers* 60, 1, 93–105.
[16] N. Saraf, K. Bazargan, D. J. Lilja, and M. D. Riedel. 2014. IIR Filters using Stochastic Arithmetic. In *DATE*.
[17] S. S. Tehrani, S. Mannor, and W. J. Gross. 2008. Fully Parallel Stochastic LDPC Decoders. *IEEE Trans. on Signal Processing* 56, 11, 5692–5703.