Error-Rate Prediction for Probabilistic Circuits with More General Structures

Mark S. K. Lau, Keck-Voon Ling, Arun Bhanu, and Vincent J. Mooney III*
School of Electrical and Electronic Engineering
Nanyang Technological University
50 Nanyang Avenue, Singapore 639798
{marklausk, ekvling, arunbhanu, vjmooney}@ntu.edu.sg

Abstract— A methodology has been proposed recently to predict error-rates of probabilistic circuits having a cascade structure. It was able to predict reasonably accurately for probabilistic ripple-carry and carry-skip adders. The objective of the present paper is twofold. First, the methodology is applied, for the first time in the literature, to a probabilistic carry-select adder, which has a more complex structure than the adders mentioned above. This is to provide additional evidence that the method is versatile and applicable to some non-trivial circuits. Second, the present paper shows that the methodology is also applicable to some seemingly non-cascade circuits. The key technique is to appropriately group circuit components into various blocks before applying the methodology. Such a preprocessing may potentially widen the scope of applicability of the methodology.

I. Introduction

As the feature size of CMOS continues to shrink, CMOS circuits become increasingly susceptible to various kinds of errors, failures, and process-induced variations. Instead of spending disproportionate efforts to avoid errors completely, probabilistic computing proposes to embrace occasional errors caused by erroneous chips [1]. Manifested in probabilistic computing are potential trade-offs between correctness of circuit operation and design parameters. A good trade-off can possibly result in significant power savings [2, 3] or increased circuit speeds [4].

For systematic design, performance evaluation, and resource allocation of probabilistic circuits, accurate prediction of error-rates is essential. A method was proposed in [5] to predict error-rates of probabilistic circuits that are cascades of some circuit blocks as shown in Fig. 1. Such a cascade structure can be found in, for examples, various adder circuits such as ripple-carry and carry-skip (CSAs) adders. In [5] the method was used to predict error-rates of a probabilistic CSA. The predicted error-rates were reasonably close to that obtained from HSPICE simulations.

A contribution of the present paper is to apply the method of [5] to a more complex circuit, namely, a probabilistic carry-select adder (PCSLA). The PCSLA has two multi-bit ripple-carry adders working in parallel, and has a more complex circuit structure (e.g., converging fan-outs) than the CSA studied in [5]. The results shown in the present paper can provide additional evidence that the method is versatile and applicable to non-trivial circuits.

*Also with Georgia Institute of Technology.

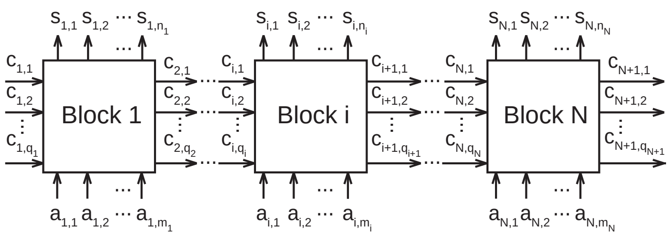

Fig. 1. A cascade circuit to which the method of [5] is applicable.

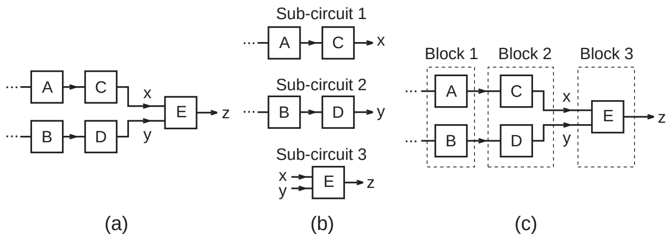

Fig. 2. Transforming converging fan-outs into a cascade structure.

In a recent study, we observed that some circuits of a directed-acyclic-graph structure can be re-configured into the same cascade structure as Fig. 1. It was done by properly grouping circuit components into blocks. We have seen in our study that this technique works for a Wallace tree multiplier. Motivated by the study, as the second contribution of the present paper, we suggest that the method of [5] is also applicable to some circuits with seemingly non-cascade structures; the key is a preprocessing step that can reconfigure components of a target circuit into blocks properly.

An example is shown in Fig. 2 (a), where the error-rate of z is to be determined. At first, it seems that this circuit does not have a cascade structure because there are two fan-out signals converging at E. One may be tempted to decompose the circuit into three cascade sub-circuits, as shown in Fig. 2 (b). The method of [5] is then applied to sub-circuits 1 and 2 separately to obtain probabilities of x and y, denoted as p_x and p_y. If x and y are statistically independent, a joint probability of x and y is simply the product $p_x p_y$. Having this joint probability, one can apply the method of [5] again to sub-circuit 3 to obtain the error-rate of z.

However, this technique may fail when x and y are correlated. Unless the correlation is known in advance, there may be insufficient information to estimate a joint probability of x and y since p_x and p_y were estimated separately.

This is now tackled by appropriately re-grouping the components into blocks, i.e., preprocessing, before applying the method. We group A and B as a block, C and D as a block, while E itself is another block, as shown in Fig. 2 (c). It is now clear that Fig. 2 (c) exhibits the same cascade structure as Fig. 1, and thus the method of [5] can be applied. Joint probabilities of x and y are now given directly by the method since they are outputs from the same block. Similar ideas were used for another error-rate predicting method in [6].

In view of this example, we are able to apply the method of [5] to circuits having cascade as well as some seemingly non-cascade structures. This technique is demonstrated in the present paper with a PCSLA as an example.

II. Carry-select Adders

We consider a carry-select adder (CSLA) which is a cascade of several identical multi-bit adders, as depicted in Fig. 3 (a). The internal structure of the ith multi-bit adder is shown in Fig. 4. For a detailed discussion of CSLAs, see for example pg. 192 of [7] and pg. 575 of [8].

The multi-bit adder shown in Fig. 4 accepts a carry-in $c_{i,1}$ and two segments of addend bits. The addend segments are added, and the resulting carry-out $c_{i+1,1}$ will serve as the carry-in of the next multi-bit adder.

Before a carry-in $c_{i,1}$ arrives, the multi-bit adder performs major amount of work required by the addition. The multi-bit adder performs the addition twice with two possible carry-in values (VSS and VDD); the additions are performed in parallel by two ripple-carry adders. Once an actual carry-in has arrived, the multiplexers (MUXes) will select the correct addition result based on the value of the actual carry-in. Then, the correct addition result will be ready for output almost instantly.

We decompose a CSLA in two different ways. The first way, which is perhaps more obvious, is to view a CSLA as a cascade of multi-bit adders, as shown in Fig. 3 (a). The CSLA then exhibits the same cascade structure of Fig. 1. We are allowed to apply the method of [5] directly, considering each multi-bit adder as a single block. This is a straightforward way of decomposing a CSLA. However, the method becomes less efficient for a block of a long bit-length [5], as the complexity of characterizing a block grows rapidly with its number of inputs/outputs.

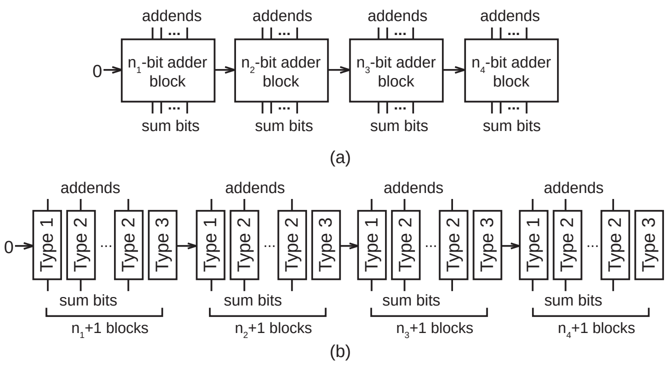

Fig. 3. Two representations of the same CSLA.

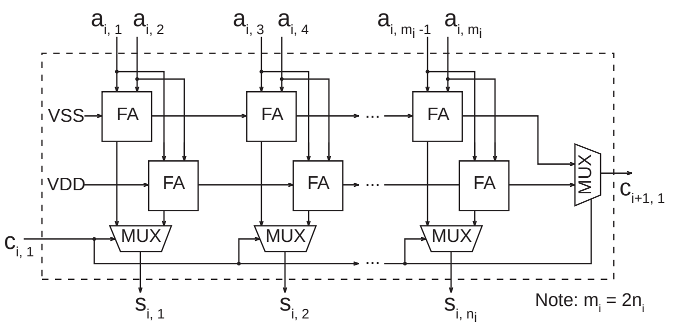

Fig. 4. A CSLA is a cascade of instances of this multi-bit adder.

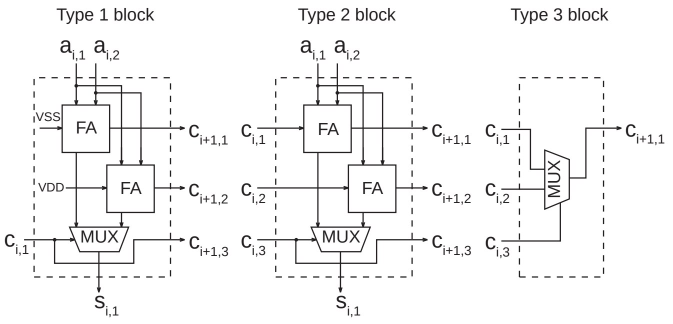

Fig. 5. The multi-bit adder of Fig. 4 is a cascade of three types of primitive blocks.

Such a scalability issue can be alleviated by further dividing a multi-bit adder into a cascade of sub-circuits, where each sub-circuit has only a small number of inputs/outputs. The multi-bit adder of Fig. 4, however, has no apparent cascade structure if we assign each full-adder (FA) or multiplexer (MUX) an individual block. It is then not clear how the multi-bit adder of Fig. 4 can be sub-divided into a cascade of smaller blocks.

In this paper, we observe that a cascade structure would immediately emerge if we group the FAs and MUXes into three types of blocks, as shown in Fig. 5. This results in another cascade representation of a CSLA, as depicted in Fig. 3 (b). This cascade structure can be handled by the method of [5]. The above-mentioned scalability issue does not exist any more. This is because each block of Fig. 5 corresponds to no more than a single bit of a CSLA, and hence the number of inputs/outputs of a block does not grow with the bit-length of a multi-bit adder.

The above discussions illustrate that, by appropriately grouping circuit components into blocks, circuits of seemingly non-cascade structure (e.g., the multi-bit adder of Fig. 4) can also be partitioned to form a cascade structure that can be handled by the method of [5].

III. A Framework for Error-rate Prediction

In this section, we briefly present the method of [5] and the associated equations for error-rate prediction. Discussions here are self-contained, and can be understood without referring to the original paper [5].

The method deals with a cascade of N circuit blocks shown in Fig. 1. Block i has $m_i + q_i$ inputs and $n_i + q_{i+1}$ outputs; the inputs $(c_{i,1}, \ldots, c_{i,q_i})$ and $(a_{i,1}, \ldots, a_{i,m_i})$ are statistically independent. Internal structures of the circuit blocks need not to be specified in advance, and there are no strong assumptions imposed.

A. Notation and Mathematical Model

We use the notation shown in Fig. 1. The first subscript refers to the block number and the second subscript refers to the bit position within a block. We shall sometimes use vector notation, which is denoted by bold symbols such as $\boldsymbol{a}_i$ and $\boldsymbol{c}_i$. The definitions of some symbols are given in Table I for the ease of reference. With vector notation, we present the ith block in a compact manner in Fig. 6 (a).

We introduce two functions, f^c_{ij} and f^s_{ij}, that model the input-output relations of the block in Fig. 6 (a). Namely,

$$c_{i+1,j} = f^c_{ij}(\boldsymbol{c}_i, \boldsymbol{a}_i), \quad s_{ij} = f^s_{ij}(\boldsymbol{c}_i, \boldsymbol{a}_i).$$

We also write in vector forms:

$$\begin{aligned}
\boldsymbol{c}_{i+1} &= (c_{i+1,1}, c_{i+1,2}, \ldots, c_{i+1,q_{i+1}}) \\
&= \boldsymbol{f}^c_i(\boldsymbol{c}_i, \boldsymbol{a}_i) \overset{\text{def}}{=} \big(f^c_{i,1}(\boldsymbol{c}_i, \boldsymbol{a}_i), \ldots, f^c_{i,q_{i+1}}(\boldsymbol{c}_i, \boldsymbol{a}_i)\big), \\
\boldsymbol{s}_i &= (s_{i,1}, s_{i,2}, \ldots, s_{i,n_i}) \\
&= \boldsymbol{f}^s_i(\boldsymbol{c}_i, \boldsymbol{a}_i) \overset{\text{def}}{=} \big(f^s_{i,1}(\boldsymbol{c}_i, \boldsymbol{a}_i), \ldots, f^s_{i,n_i}(\boldsymbol{c}_i, \boldsymbol{a}_i)\big).
\end{aligned}$$

Figure 6 (b) shows a probabilistic block with inputs and outputs $\boldsymbol{c}'_i$, $\boldsymbol{c}'_{i+1}$, $\boldsymbol{a}_i$ and $\boldsymbol{s}'_i$. We assume that Figs. 6 (a) and (b) have the same input $\boldsymbol{a}_i$. Hence, the probabilistic block's input $\boldsymbol{a}$ is always correct. However, $\boldsymbol{c}_i$ and $\boldsymbol{c}'_i$ may be different, and we use a prime to indicate the distinction. The prime notation here does not denote complementation, and hence variables such as $\boldsymbol{c}'_i$ and $\boldsymbol{c}_i$ are in general completely distinct.

Assume that errors at the outputs of a probabilistic block are caused by some inherent error sources. The error sources cause an error and subsequently flip an output bit. Variables e^c_{ij} and e^s_{ij} were introduced in [5] such that

$$c'_{i+1,j} = e^c_{ij} \oplus f^c_{ij}(\boldsymbol{c}'_i, \boldsymbol{a}_i), \quad s'_{i,j} = e^s_{ij} \oplus f^s_{ij}(\boldsymbol{c}'_i, \boldsymbol{a}_i).$$

TABLE I
Mathematical symbols for the ith block.

Types	Symbols for the ith block
Input	$\boldsymbol{a}_i = (a_{i,1}, a_{i,2}, \ldots, a_{i,m_i})$
Input	$\boldsymbol{c}_i = (c_{i,1}, c_{i,2}, \ldots, c_{i,q_i})$
Output	$\boldsymbol{c}_{i+1} = (c_{i+1,1}, \ldots, c_{i+1,q_{i+1}})$
Probabilistic input	$\boldsymbol{c}'_i = (c'_{i,1}, c'_{i,2}, \ldots, c'_{i,q_i})$
Probabilistic output	$\boldsymbol{c}'_{i+1} = (c'_{i+1,1}, \ldots, c'_{i+1,q_{i+1}})$
Output bit-flip indicator	$\boldsymbol{e}^c_i = (e^c_{i,1}, e^c_{i,2}, \ldots, e^c_{i,q_{i+1}})$
Output	$\boldsymbol{s}_i = (s_{i,1}, s_{i,2}, \ldots, s_{i,n_i})$
Probabilistic output	$\boldsymbol{s}'_i = (s'_{i,1}, s'_{i,2}, \ldots, s'_{i,n_i})$
Output bit-flip indicator	$\boldsymbol{e}^s_i = (e^s_{i,1}, e^s_{i,2}, \ldots, e^s_{i,n_i})$

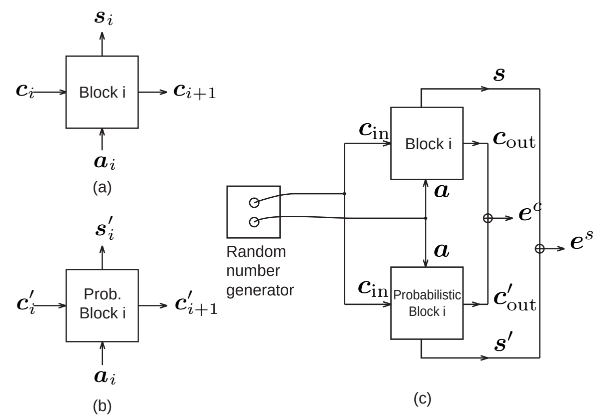

Fig. 6. Characterizing a probabilistic block.

Hence, for example, $e^c_{ij} = 1$ means that the ith probabilistic block has made an error at the jth carry output. The above equations can also be written in vector forms:

$$\boldsymbol{c}'_{i+1} = \boldsymbol{e}^c_i \oplus \boldsymbol{f}^c_i(\boldsymbol{c}'_i, \boldsymbol{a}_i), \quad \boldsymbol{s}'_i = \boldsymbol{e}^s_i \oplus \boldsymbol{f}^s_i(\boldsymbol{c}'_i, \boldsymbol{a}_i) \tag{1}$$

where $\boldsymbol{e}^c_i$ and $\boldsymbol{e}^s_i$ were defined in Table I, and $\oplus$ means component-wise XOR on two vectors of binary numbers.

Finally, two assumptions are required, as stated in [5]. First, $\boldsymbol{c}_i$ and $\boldsymbol{a}_i$ are statistically independent. Second, $\boldsymbol{e}^c_i$ is conditionally independent of $\boldsymbol{c}_i$ given $\boldsymbol{c}'_{i+1}$ and $\boldsymbol{a}_i$.

B. Equations for Error-rate Prediction

Suppose that one is interested in predicting the probability that $\boldsymbol{c}_{i+1} = \boldsymbol{\alpha}$, $\boldsymbol{c}'_{i+1} = \boldsymbol{\beta}$ for some vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ of binary numbers. Let $\boldsymbol{\gamma}^c$, $\boldsymbol{\delta}$, $\boldsymbol{\epsilon}$ and $\boldsymbol{\zeta}$ be vectors of binary numbers, and let $\mathcal{C}(\boldsymbol{\alpha}, \boldsymbol{\beta})$ be the set of all $(\boldsymbol{\gamma}^c, \boldsymbol{\delta}, \boldsymbol{\epsilon}, \boldsymbol{\zeta})$ such that $\boldsymbol{e}^c_i = \boldsymbol{\gamma}^c$, $\boldsymbol{c}_i = \boldsymbol{\delta}$, $\boldsymbol{c}'_i = \boldsymbol{\epsilon}$ and $\boldsymbol{a}_i = \boldsymbol{\zeta}$ results in

$$\begin{aligned}
\boldsymbol{c}_{i+1} &= \boldsymbol{f}^c_i(\boldsymbol{c}_i, \boldsymbol{a}_i) = \boldsymbol{f}^c_i(\boldsymbol{\delta}, \boldsymbol{\zeta}) = \boldsymbol{\alpha}, \\
\boldsymbol{c}'_{i+1} &= \boldsymbol{e}^c_i \oplus \boldsymbol{f}^c_i(\boldsymbol{c}'_i, \boldsymbol{a}_i) = \boldsymbol{\gamma}^c \oplus \boldsymbol{f}^c_i(\boldsymbol{\epsilon}, \boldsymbol{\zeta}) = \boldsymbol{\beta}.
\end{aligned}$$

The set $\mathcal{C}(\boldsymbol{\alpha}, \boldsymbol{\beta})$ can be obtained by, for example, exhausting all possible values of $\boldsymbol{e}^c_i$, $\boldsymbol{c}_i$, $\boldsymbol{c}'_i$ and $\boldsymbol{a}_i$.

Equation (2) in the next page was derived in [5]. It is recursive, namely, a probability for the $(i+1)$th block, i.e., $P(\boldsymbol{c}'_{i+1} = \boldsymbol{\alpha}, \boldsymbol{c}_{i+1} = \boldsymbol{\beta})$, is computed from probabilities

$$P(\boldsymbol{c}'_{i+1} = \boldsymbol{\alpha}, \boldsymbol{c}_{i+1} = \boldsymbol{\beta}) = \sum_{\mathcal{C}(\boldsymbol{\alpha},\boldsymbol{\beta})} P(\boldsymbol{e}^c_i = \boldsymbol{\gamma}^c \mid \boldsymbol{c}'_i = \boldsymbol{\epsilon}, \boldsymbol{a}_i = \boldsymbol{\zeta}) P(\boldsymbol{a}_i = \boldsymbol{\zeta}) P(\boldsymbol{c}'_i = \boldsymbol{\epsilon}, \boldsymbol{c}_i = \boldsymbol{\delta}). \tag{2}$$

$$P(s'_{ij} \neq s_{ij}) = \sum_{\mathcal{S}_{ij}} P(e^s_{ij} = \gamma^s \mid \boldsymbol{c}'_i = \boldsymbol{\epsilon}, \boldsymbol{a}_i = \boldsymbol{\zeta}) P(\boldsymbol{a}_i = \boldsymbol{\zeta}) P(\boldsymbol{c}'_i = \boldsymbol{\epsilon}, \boldsymbol{c}_i = \boldsymbol{\delta}). \tag{3}$$

of the previous block denoted as $P(\boldsymbol{c}'_i = \boldsymbol{\epsilon}, \boldsymbol{c}_i = \boldsymbol{\delta})$. If probabilities of the first block are known, (2) can be used iteratively to obtain probabilities of the other blocks. In (2), $P(\boldsymbol{a}_i = \boldsymbol{\zeta})$ is a probability of the data, which is given by the data statistics and is known in advance.

The conditional probability in (2) indeed characterizes the probabilistic behavior of the ith block. Recall from (1) that $\boldsymbol{e}^c_i$ indicates whether errors were made at the output $\boldsymbol{c}_i$ of the i block. Therefore, one may think of the conditional probability as a probability of error given the inputs $\boldsymbol{c}'_i = \boldsymbol{\epsilon}$ and $\boldsymbol{a}_i = \boldsymbol{\zeta}$. It characterizes the response of the ith probabilistic block to a particular input pattern.

The conditional probability in (2) can be estimated by experiments on the ith probabilistic block, where a large number of randomly generated inputs are fed to the block and the number of erroneous outputs are recorded subsequently [5]. This will be discussed in the next subsection.

Similarly, error-rates for $\boldsymbol{s}_i$ can be predicted by equation (3), whose derivation is found in [5]. Suppose that one wishes to estimate the probability that $s_{ij} \neq s'_{ij}$. Let γ^s be a binary number, and let $\boldsymbol{\delta}$, $\boldsymbol{\epsilon}$ and $\boldsymbol{\zeta}$ be vectors of binary numbers. Let $\mathcal{S}_{ij}$ be the set of all possible $(\gamma^s, \boldsymbol{\delta}, \boldsymbol{\epsilon}, \boldsymbol{\zeta})$ such that $s'_{ij} \neq s_{ij}$ if $e^s_{ij} = \gamma^s$, $\boldsymbol{c}_i = \boldsymbol{\delta}$, $\boldsymbol{c}'_i = \boldsymbol{\epsilon}$ and $\boldsymbol{a}_i = \boldsymbol{\zeta}$.

C. Characterizing Probabilistic Blocks

Recall that (2) and (3) require estimates of the following probabilities for the ith block:

$$P(\boldsymbol{e}^c_i = \boldsymbol{\gamma}^c \mid \boldsymbol{c}'_i = \boldsymbol{\epsilon}, \boldsymbol{a}_i = \boldsymbol{\zeta}),\ P(e^s_{ij} = \gamma^s \mid \boldsymbol{c}'_i = \boldsymbol{\epsilon}, \boldsymbol{a}_i = \boldsymbol{\zeta}). \tag{4}$$

It is done using the experiment illustrated in Fig. 6 (c). The lower block is the ith block of a probabilistic circuit, and the upper one is deterministic. This process is referred to as characterizing a probabilistic block [5, 9, 10].

The two blocks in Fig. 6 (c) have the same inputs generated by random-number generators of certain probability distributions. Independence between $\boldsymbol{c}_{\text{in}}$ and $\boldsymbol{a}$ is needed, due to the first assumption mentioned in the last paragraph of Section III-A. Outputs given by the blocks are compared using the XOR function to check for errors.

To estimate (4), we perform the experiment of Fig. 6 (c) with Θ sets of randomly generated inputs, where each set contains a vector of q_i bits for $\boldsymbol{c}_{\text{in}}$, and a vector of m_i bits for $\boldsymbol{a}$. Let $\Gamma(\Theta)$ be the number of randomly generated input sets such that $\boldsymbol{c}_{\text{in}} = \boldsymbol{\epsilon}$ and $\boldsymbol{a} = \boldsymbol{\zeta}$.

Out of those $\Gamma(\Theta)$ input sets, there are $\Lambda^c(\Theta)$ sets causing the output of the upper XOR function equal to $\boldsymbol{\gamma}^c$; and there are $\Lambda^s(\Theta)$ sets causing the jth bit of the output of the lower XOR function equal to γ^s. By the strong law

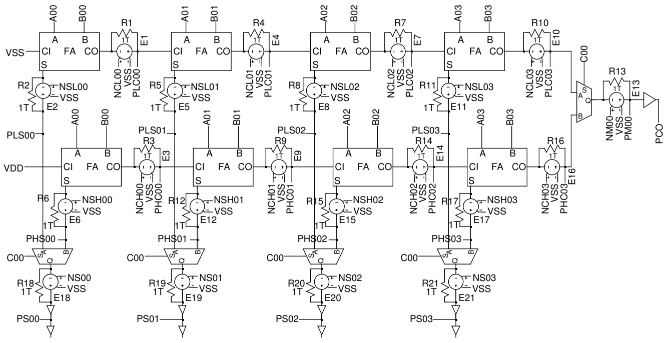

Fig. 7. A probabilistic block used in HSPICE simulations.

of large numbers (Theorem 4.1 of [11] and equation (4.1) thereafter), we do the following estimation:

$$P(\boldsymbol{e}^c_i = \boldsymbol{\gamma}^c \mid \boldsymbol{c}'_i = \boldsymbol{\epsilon},\, \boldsymbol{a}_i = \boldsymbol{\zeta}) \approx \lim_{\Theta \to \infty} \frac{\Lambda^c(\Theta)}{\Gamma(\Theta)}, \tag{5}$$

$$P(e^s_{ij} = \gamma^s \mid \boldsymbol{c}'_i = \boldsymbol{\epsilon},\, \boldsymbol{a}_i = \boldsymbol{\zeta}) \approx \lim_{\Theta \to \infty} \frac{\Lambda^s(\Theta)}{\Gamma(\Theta)}. \tag{6}$$

In practice, an infinitely large Θ is impossible. However, one has to make sure that Θ is sufficiently large to allow the estimated probabilities to converge and stabilize.

IV. Application to Probabilistic CSLA

The method of [5] described previously is now applied to a PCSLA. In particular, we use (2) and (3) to predict error-rates of a PCSLA, and then compare with error-rates obtained from HSPICE simulations. We also experiment with two decomposition methods proposed in Section II, and discuss the accuracy of their predictions.

A. Obtaining Error-rates by HSPICE Simulation

Following [2, 5, 10, 9, 12], we construct a probabilistic circuit by coupling noise sources to the output terminals of some deterministic components. This is seen from Fig. 7, which shows a diagram of a block in our simulations.

Each of the noise sources is generated independently, and is of Gaussian distribution with zero mean and non-zero root-mean-square (RMS) value. A noise source is simulated in HSPICE by a voltage-controlled voltage-source (VCVS) with a voltage gain equal to the desired noise RMS. In Fig. 7 VCVSes are labeled by E1, E2, ..., etc. Random numbers of the standard Gaussian distribution are generated by Matlab. A VCVS multiplies these random numbers by its voltage gain to get a Gaussian noise source of the desired RMS. Noise samples are added to the output terminals every 500 ps.

Our simulations use the Synopsys 90nm generic library. For this technology, the nominal voltage is 1.2 V. We follow [5, 9, 10, 12] to use a noise RMS of 0.2 V.

To construct a PCSLA of 16 bits, we cascade four identical copies of the block shown in Fig. 7. Two hundred thousand sets of input bits are randomly generated by Matlab. All input bits are independent of each other, and are of uniform distribution. The sampling period is 5 ns, which is longer than the worst-case propagation delays of the LSB-input-to-MSB-sum critical paths of the 16-bit PCSLA. That is, every 5 ns, a newly generated set of input bits is fed to the 16-bit PCSLA, and at the same time, the output bits (which are due to the past input vectors) are sampled. All blocks in the 16-bit PCSLA are identical, and have the same supply voltages chosen from the set $\{0.8, 0.9, \ldots, 1.2\}$.

A sampled output has the logical state zero (one) if the output voltage is lower (equal to or higher) than half of the supply voltage. We compare the sampled outputs with the correct results. In our simulations, instead of actually implementing a deterministic CSLA in HSPICE, the deterministic (correct) outputs are calculated by arithmetic functions. The simulated error-rate of the ith sum-bit of a PCSLA is defined as the number of errors observed at the ith sum-bit in the HSPICE simulation, normalized by the number of observed output samples, i.e., 200,000.

B. Predicting Error-rates

In the following, we are going to apply (2) and (3) to predict error-rates of a PCSLA, which are then compared with those error-rates obtained by HSPICE simulations.

As suggested in Section II, we may consider a PCSLA as a cascade of identical instances of the block in Fig. 4. In this case, the block has to be characterized by carrying out an experiment illustrated in Fig. 6 (c). Since all blocks are identical in the PCSLA considered here, the characterization has to be done only once. In this characterization experiment, 200,000 realizations of the 9-bit input vector $(\boldsymbol{c}_{\text{in}}, \boldsymbol{a})$ are randomly generated by Matlab. All input bits are of uniform distribution and independent of each other. Every 5 ns, a newly generated realization of $(\boldsymbol{c}_{\text{in}}, \boldsymbol{a})$ is fed to the blocks, and at the same time, their outputs are sampled. Finally, the required conditional probabilities are estimated using (5) and (6).

When characterizing a probabilistic block, noise filtering effect of logic gates has to be taken into account. It was observed in [12] that an output signal of a logic gate is often less noisy than the corresponding (noisy) input signal. The main reason for this appears to be the so-called noise filtering effect discussed in Section VI.B of [13]. In short, noise filtering occurs when the duration of a noise pulse is shorter than the propagation delay of a gate. It was observed in [12] that it has prominent effect on the accuracy of characterization of a probabilistic circuit.

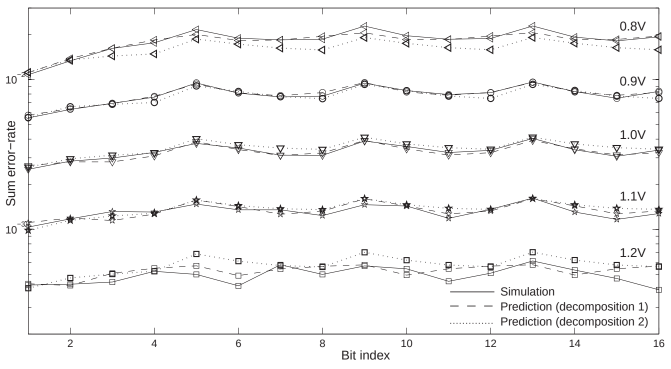

Fig. 8. Error-rates obtained by (2) and (3), and by HSPICE.

A practical way to deal with noise filtering effect is to use a buffer to approximate the output loads of an output terminal [5, 9, 10, 12]. The buffer is connected to a block's output, and output samples are taken after this buffer. A buffer should have a suitable size, in order to accurately model the actual loads that will be connected to the block when it is chained with other blocks to form a PCSLA.

We also observe that different supply voltages require slightly different buffer sizes. This is because the supply voltage of a logic gate has an effect on its propagation delay, which in turn affects the strength of noise filtering. However, for simplicity, in this paper we only use a single buffer size for all of the voltages 0.8-1.2 V.

The experiment results are shown in Fig. 8, which shows the predicted and simulated error-rates of the sum-bits. "Decomposition 1" in Fig. 8 refers to the PCSLA of Fig. 3 (a). We see that the predictions are fairly close to the simulated error-rates for various supply voltages. For clarity, the same results are also reproduced in Table II. The numbers of errors made by the sum-bits in the simulation are shown in the Sim rows. For example, for 0.8 V, 3,677 out of 200,000 samples taken at the seventh sum-bit (S7) are incorrect. The predicted number is 3,671, which is shown in the Pred-1 row. This number is given by multiplying the predicted error-rate by 200,000 and rounding to the nearest integer.

On average over all bit positions, a predicted error-rate is within 1-6% of the corresponding simulated value when the supply voltage is within 0.8-1.1 V. Whereas, for 1.2 V, the deviations are on average 11% of the simulated values. The relatively large deviation for 1.2 V is due to the fact that 1.2 V is the nominal voltage of the Synopsys 90nm technology and hence errors rarely occur when the probabilities in (4) were estimated in the characterization process. The estimated probabilities are therefore somewhat less accurate.

C. Prediction with Another Decomposition Approach

We now use another way to decompose a PCSLA. We decompose a PCSLA into three types of blocks shown in Fig. 5. Hence, besides Fig. 3 (a), the PCSLA can also be

TABLE II
Number of errors (out of 200,000 samples) predicted by the proposed equations and simulated by HSPICE.

V	—	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16
0.8	Sim	2150	2662	3226	3515	4311	3770	3677	3716	4543	3920	3709	3755	4550	3832	3622	3850
0.8	Pred-1	2218	2762	3234	3668	3994	3643	3671	3882	4099	3696	3697	3895	4106	3699	3698	3896
0.8	Pred-2	2233	2704	2871	2963	3719	3443	3239	3146	3809	3488	3262	3157	3815	3491	3263	3158
0.9	Sim	1108	1263	1387	1525	1895	1622	1529	1548	1890	1687	1587	1633	1925	1662	1498	1666
0.9	Pred-1	1157	1275	1383	1547	1866	1629	1559	1635	1909	1651	1570	1640	1912	1652	1571	1640
0.9	Pred-2	1117	1318	1366	1403	1809	1663	1539	1489	1852	1684	1549	1494	1855	1686	1550	1495
1.0	Sim	503	570	599	652	749	698	625	624	777	714	651	673	808	679	614	676
1.0	Pred-1	533	564	564	618	768	681	623	647	783	689	627	649	784	689	627	649
1.0	Pred-2	520	591	625	647	805	734	697	683	823	743	701	685	824	744	701	685
1.1	Sim	207	235	263	261	295	270	270	247	292	286	237	272	323	261	233	255
1.1	Pred-1	222	238	229	253	314	284	252	264	320	287	253	265	320	287	254	265
1.1	Pred-2	197	230	245	256	313	288	274	271	321	292	276	272	321	292	276	272
1.2	Sim	85	86	89	105	100	84	116	100	114	109	90	102	123	107	94	79
1.2	Pred-1	87	84	102	110	114	98	109	113	116	99	109	114	116	99	109	114
1.2	Pred-2	81	95	101	105	137	123	115	112	141	125	116	113	141	125	116	113

represented as the cascade circuit illustrated in Fig. 3 (b). Because there are three different types of blocks, three rounds of characterization are required, each round for each individual block. We select the same parameters as those used previously, i.e., 200,000 input realizations, uniformly distributed and independent input bits, 5 ns sampling time, and 0.2 V noise RMS.

The predictions by (2) and (3) are shown in Fig. 8 (decomposition 2). The results are also shown in the Pred-2 row of Table II. The predicted error-rates are, again, fair close to those simulated by HSPICE. Nevertheless, they are not as accurate as those obtained by the first decomposition method presented previously (Pred-1). On average over all bit positions, a predicted error-rate is within 3-7% of the corresponding simulated value when the supply voltage is within 0.9-1.1 V. Whereas, for 0.8 and 1.2 V, the deviations are on average 12% and 18%, respectively, of the simulated values.

A possible explanation is that the influence of supply voltage on the noise filtering effect of logic gates now becomes more significant. Since we have used only one buffer size for all supply voltages, the predictions would be accurate for some voltages (e.g., 0.9-1.1 V) but less accurate for other voltages. If our speculation is correct, then one can use different buffer sizes for different supply voltages in order to improve the accuracy.

V. Conclusion

The method of [5] has been used to predict error-rates of a probabilistic carry-select adder (PCSLA). Experimental results suggested that the predictions were close to error-rates simulated by HSPICE simulations. These results provided evidence that the error-rate prediction method is versatile and applicable to some non-trivial circuits.

We further suggest that, by appropriately grouping circuit components into various blocks, the error-rate prediction method can handle not only cascade circuits, but also circuits with more complex structure. This has been demonstrated with a PCSLA, in which a multi-bit adder circuit with a seemingly non-cascade structure was transformed into a cascade circuit, which was then handled successfully by the error-rate prediction method.

References

[1] K. V. Palem, "Energy aware computing through probabilistic switching: A study of limits," *IEEE Transactions on Computers*, vol. 54, no. 9, pp. 1123–1137, 2005.

[2] J. George, B. Marr, B. E. S. Akgul, and K. V. Palem, "Probabilistic arithmetic and energy efficient embedded signal processing," *Proceedings of 2006 CASES*, pp. 158–168.

[3] M. S. K. Lau, K. V. Ling, and Y. C. Chu, "Energy-aware probabilistic multiplier: Design and analysis," *Proceedings of 2009 CASES*, pp. 281–290.

[4] L. N. Chakrapani, K. K. Muntimadugu, A. Lingamneni, J. George, and K. V. Palem, "Highly energy and performance efficient embedded computing through approximately correct arithmetic: A mathematical foundation and preliminary experimental validation,"*Proceedings of 2008 CASES*, pp. 187–196.

[5] M. S. K. Lau, K. V. Ling, A. Bhanu, and V. J. Mooney III, "Error-rate prediction for a class of probabilistic circuits with applications to carry-skip adders," *submitted to the 16th ASP-DAC (obtainable from http://web.me.com/mark.lau).*

[6] S. Krishnaswamy, G. F. Viamontes, I. L. Markov, and J. P. Hayes, "Probabilistic transfer matrices in symbolic reliability analysis of logic circuits," *ACM Transactions on Design Automation of Electronic Systems*, vol. 13, no. 1, Article 8, 2008.

[7] A. Chandrakasan, W. J. Bowhill, and F. Fox, Eds., *Design for High-Performance Microprocessor Circuits.* IEEE Press, 2000.

[8] J. M. Rabaey, A. Chandrakasan, and B. Nikolić, *Digital Integrated Circuits: A Design Perspective*, 3rd ed. Prentice Hall, 2003.

[9] M. S. K. Lau, K. V. Ling, Y. C. Chu, and A. Bhanu, "Modeling of probabilistic ripple-carry adders," *Proceedings of 2010 DELTA*, pp. 201–206.

[10] ——, "A general mathematical model of probabilistic ripple-carry adders," in *Proceedings of 2010 DATE*, CD-ROM.

[11] S. Ross, *A First Course in Probability*, 7th ed. Prentice Hall, 2006.

[12] A. Bhanu, M. S. K. Lau, K. V. Ling, V. J. Mooney III, and A. Singh, "A more precise model of noise based CMOS errors," *Proceedings of 2010 DELTA*, pp. 99–102.

[13] P. Korkmaz, B. E. S. Akgul, and K. V. Palem, "Energy, performance, and probability tradeoffs for energy-efficient probabilistic CMOS circuits," *IEEE Transactions on Circuits and Systems I*, vol. 55, no. 8, pp. 2249–2262, 2008.