On the Reliable Performance of Sequential Adders for Soft Computing

Jinghang Liang
ECE Department
University of Alberta
Edmonton, Canada
jinghang@ualberta.ca

Jie Han
ECE Department
University of Alberta
Edmonton, Canada
jhan8@ualberta.ca

Fabrizio Lombardi
ECE Department
Northeastern University
Boston, MA, U.S.A
lombardi@ece.neu.edu

***Abstract*— Addition is a significant operation in soft computing; several sequential adder designs have been proposed in the technical literature. These adders show different operational profiles; some of them are inspired by biological networks or the probabilistic nature of nanometric devices (such as the Lower-part OR Adder (LOA) and the Probabilistic Full Adder (PFA)). This paper deals with the reliability assessment and comparison of these sequential adder implementations. A new metric referred to as the mean error distance (MED) is proposed as a unified figure for evaluating the reliability of both probabilistic and deterministic adders. Reliability is analyzed using the so-called sequential probability transition matrices (S-PTMs) with respect also to error masking (as occurring due to the sequential nature of the addition process). A baseline sequential adder implementation, referred to as the Lower-bit Ignored Adder (LIA), is used as a benchmark for evaluating the other implementations. It is shown that compared with the LIA, the PFA has a better reliability at a small gate error rate, but at the cost of a larger overhead in area and therefore static power consumption. The LOA achieves a good tradeoff between reliability, area, power and delay compared to the LIA and PFA implementations.**

Keywords- Sequential adders, Soft computing, Reliability, Error masking, Approximate logic, Imprecise arithmetic, Mean error distance.

I. INTRODUCTION

Methodologies for soft computing rely on the feature that many applications can tolerate some loss of precision and the solution can be tailored to tolerate some degree of uncertainty. Deterministic, explicit, and precise models and algorithms are not always suitable to solve these types of applications [1]. However, soft computing applications are mostly implemented using digital binary logic circuits, thus operating with a high degree of predictability and precision. A framework based on a precise and specific implementation can still be used with a methodology that intrinsically has a lower degree of precision and an increasing uncertainty in operation. While this may be viewed as a potential conflict, such an approach tailors the significant advantage of soft computing (and its inherent tolerance to some imprecision and uncertainty) to a technology platform implemented by conventional digital logic and systems. The paradigm of soft computation relies on relaxing fully precise and completely deterministic building blocks (such as a full adder) when for example, implementing bio-inspired systems. This allows nature inspired computation to redirect the existing design process of digital circuits and systems by taking advantage of a decrease in complexity and cost with possibly a potential increase in speed and performance.

One of the fundamental arithmetic operations in many applications of soft computing is addition. Soft additions are generally based on the operation of *deterministic approximate logic* or *probabilistic imprecise arithmetic* (categorized in [2] as design-time and run-time techniques). Two recently proposed adder architectures are representatives of these types. The first is the bio-inspired lower-part OR adder (LOA) [3]; LOA is based on approximate logic, whose truth table is slightly different from the original truth table of a full adder. The use of a LOA based architecture results in approximations to the addition, making it deterministically different from the precise operational outcome. The other design is known as the probabilistic full adder (PFA) [4]; its implementation is based on probabilistic CMOS, a technology platform for modeling the behavior of future designs as well as reducing energy consumption.

The objective of this paper is to assess these two sequential adder designs as well as a baseline design with respect to reliability and performance for soft computing. A new metric referred to as mean error distance (MED) is proposed to characterize the reliability of a sequential adder implementation. The MED is used with sequential probability transition matrices (S-PTMs) and is able to evaluate the reliability of both probabilistic and deterministic adders in the presence of so-called error masking [5]. An adder implementation with reduced precision, referred to as the lower-bit ignored adder (LIA), is then investigated as a baseline for assessing the other adders. A detailed analysis and simulation results are presented to substantiate the practicality of these different sequential adders for soft computing.

II. REVIEW

A sequential adder (Fig. 1) is often used in applications such as neural computing. The sequential adder consists of a *k*-bit full adder concatenated with a *k*-bit register.

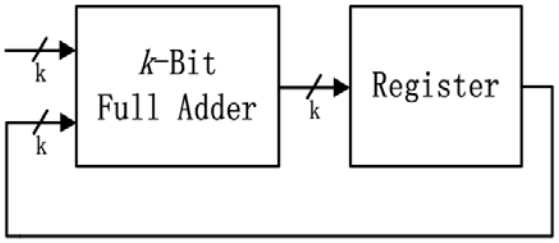


Fig. 1 A *k*-bit sequential adder.

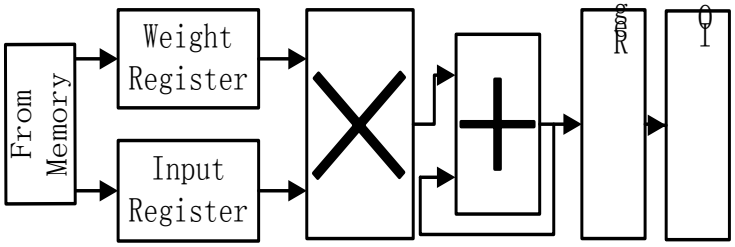
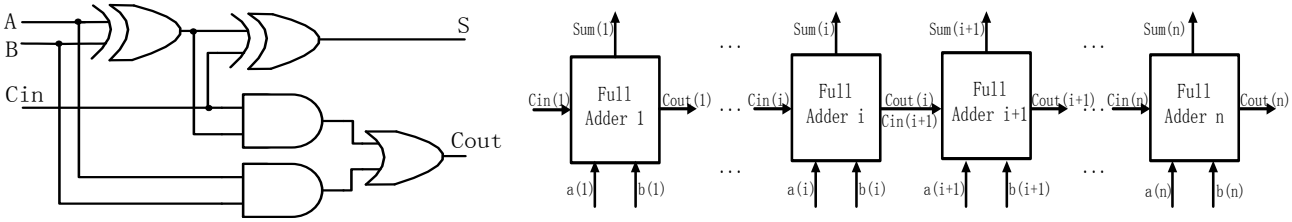


Fig. 2 Sequential architecture for VLSI implementation of a neural network [3].

Consider as an example the sequential adder proposed in [3]; Fig. 2 shows a sequential implementation of a face recognition neural network. In this implementation, a neuron is scheduled to perform on a sequential basis the computation of all neurons of the network. Next, different implementations of a full adder are reviewed; particular emphasis is devoted to features relevant to soft computing.

Conventional Full Adder (CFA): Fig. 3(a) shows a general (precise) one-bit full adder; the CFA is commonly connected in a ripple-carry implementation, i.e., by cascading the circuit of Fig. 3(a) in a linear array, as shown in Fig. 3(b).



(a) (b)

Fig. 3 (a) Implementation of a one-bit full adder, (b) Ripple carry adder implementation.

Lower-Part-OR Adder (LOA): Differently from conventional designs that strictly operate according to the exact function (as defined by its truth table), an approximate logic implementation ignores some entries in the truth table. This feature allows balancing precision with other performance metrics. The recently proposed LOA is based on such a design [3]. A LOA divides a *k*-bit addition into two smaller parts, i.e., two modules of *m*-bits and *n*–bits respectively. As shown in Fig. 4, the *m*-bit module of a LOA uses a smaller but precise adder (referred to as the *sub-adder*) to compute the exact values of the *m* most significant bits of the result (also referred to as the upper part). Additional OR gates are used to approximately compute the *n* least significant bits (also referred to as the lower part) of the sum by applying a bitwise OR operation on the respective input bits. An additional AND gate is used to generate the carry-in for the precise sub-adder when the most significant bits of both the lower-part inputs are "1." As this implementation ignores the "trivial" carries in the lower part of the LOA adder, it may result in a loss of precision. Albeit using different structures, the approximate adder designs in [6] and [7] belong to the same category.

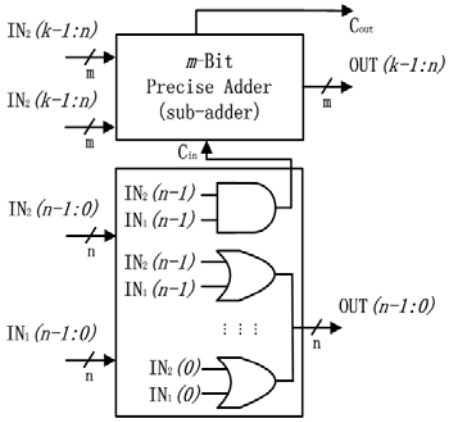


Fig. 4 Hardware structure of the lower-part-OR adder (LOA) [3].

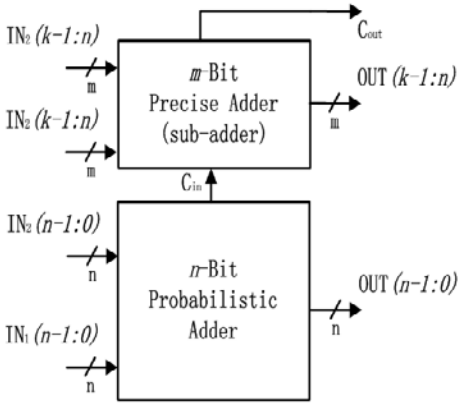


Fig. 5 Hardware structure of the probabilistic full adder (PFA).

Probabilistic Full Adder (PFA): Probabilistic CMOS (PCMOS) is considered as one of the many techniques to achieve savings in power consumption, while balancing performance at nanometric ranges. For a *k*-bit PFA, the most significant *m*-bit adders are implemented using perfectly reliable (and thus deterministic) gates, while the least significant *n* bits are implemented in PCMOS. The structure of this type of adder is shown in Fig. 5.

III. Sequential Probability Transition Matrices and Error Masking

Prior to presenting the analysis, several definitions need to be introduced for completeness. The *signal probability* is defined as the probability of a signal being logical "1." In a combinational circuit, the *output probability* for a specific input vector *I* is defined as the joint probability that all of the outputs are "1" when the input vector is *I*. The *output reliability* for input *I* is defined as the joint probability that all outputs are correct for *I*. Any output that deviates from the correct value, is considered an *unreliable* output. This definition of reliability considers the correlation among signals and is therefore used in this paper unless noted otherwise. *Circuit reliability* is defined as the average output reliability over all possible input vectors.

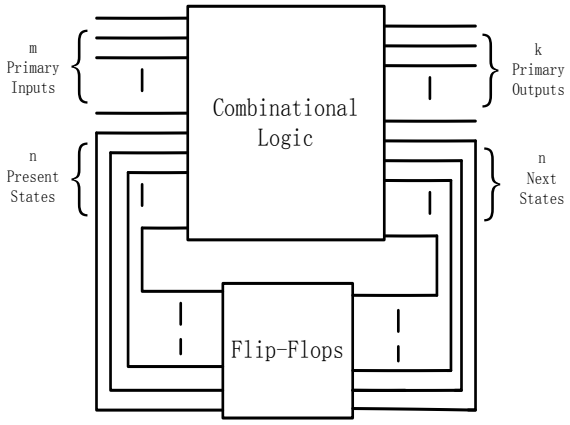


Fig. 6 Mealy model of a sequential circuit.

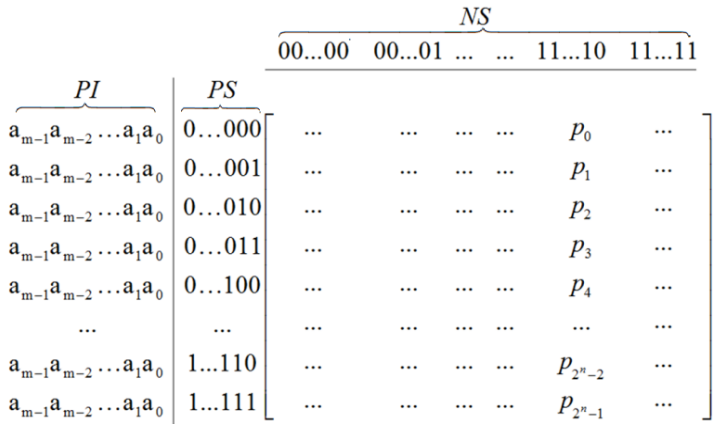


Fig. 7 A sub-matrix of the S-PTM Φ_{ns} used to obtain restoring inputs. "*PI*" represents primary inputs, "*PS*" represents present states. "*NS*" represents next states. "P_j" inside the matrix represents a large probability while "…" represents a small probability.

For sequential circuits the following definitions are also introduced. *Restoring inputs* are the primary inputs or a sequence of primary inputs that logically mask the feedback signals in a sequential circuit. *Error masking* in a sequential circuit refers to the scenario by which the feedback signals are logically masked by specific combinations of primary inputs (referred to as restoring inputs). Therefore a sequential circuit is said to be *reliable* if error masking frequently occurs; it is *unreliable* otherwise. In this paper the adders introduced previously are analytically assessed for *circuit reliability* using the so-called sequential probability transition matrix (S-PTM) technique. This technique is based on a Mealy model of a sequential circuit (Fig. 6). S-PTMs define several matrices mapping from the *m*+*n* inputs to the *k* primary outputs (Φ_{po}) and to the *n* next states (Φ_{ns}), also from the *n* next states to the *n* present states. By employing matrix operations, performance evaluation of a sequential circuit can be performed at each clock cycle (albeit this may incur a substantial computational overhead).

The process of error masking can be described using S-PTMs as follows. According to the values of the primary inputs, Φ_{ns} is divided into 2^m row blocks, each block consisting of 2^n row vectors for a specific combination of primary inputs. In this block, if the entries in a column (corresponding to the transition probabilities from the inputs to a specific output) have mostly large values, then the primary input combination for this block (denoted as $a_{m-1}a_{m-2}\ldots a_1a_0$ in Fig. 7) is defined as a set of so-called *restoring inputs*. A similar procedure can be applied to the analysis of Φ_{po}; it characterizes whether restoring inputs exist such that the effects of errors accumulated in the flip-flops can be taken into account. This scenario is generally referred to as *error masking.* Error masking can occur also over multiple steps (clock cycles); a detailed analysis of multiple-step error masking using state transition matrices and binary decision diagrams is provided in [5].

IV. Error Distance

In this section, a new metric for evaluating the reliability of an adder is proposed; subsequently, the performance of different implementations of sequential adders is assessed by using S-PTMs. This new metric relates performance to reliability for soft computing operation of the adders. Consider as an example the case in which the exact Sum output of an adder is "100101" and all other values are considered unreliable (for example, both "100100" and "110101" represent

unreliable values). However, these two output values have different implications when compared to the correct value: "100100" means the output is different by 1 (or at a distance of 1) from the correct value, while "110101" is different by 16 (or at a distance of 16) from the correct value. So, the output could take different (and erroneous) values that are substantially different from the correct Sum value. This is dependent on the error and its effect on the addition, i.e., a lower bit error has less impact on the output Sum.

Under these circumstances, the metric of circuit reliability has limited usefulness in assessing the performance of an adder, because it considers only the presence of an error, but not the error's implication on the addition. A new metric referred to as *error distance* (ED) is therefore proposed. In general, the ED between two binary numbers, *a* (erroneous) and *b* (correct, i.e., golden), is given by:

$$ED(\mathrm{a},\mathrm{b}) = |\mathrm{a}-\mathrm{b}| = \left|\sum_i \mathrm{a}[i]*2^{\mathrm{i}} - \sum_j \mathrm{b}[\mathrm{j}]*2^{\mathrm{j}}\right|, \quad (1)$$

where *i* and *j* are the indices for the bits in *a* and *b*, respectively. In the previous example, the two erroneous values "100100" and "110101" have an ED of 1 and 16 to the golden "100101," respectively.

For a non-deterministic implementation, the output is probabilistic and usually follows a distribution for a given input. In this case, the ED of the output (denoted by *d*) is defined as the weighted average of EDs of all possible outputs to the nominal output. Assume that for a given input, the output has a nominal value *b*, but it can take any value given in a set of vectors $(\mathrm{b}_1, \mathrm{b}_2, \ldots \mathrm{b}_\mathrm{N})$; the ED of the output is then given by:

$$\mathrm{d} = \sum_i \mathrm{ED}(\mathrm{b_i}, \mathrm{b}) * \mathrm{p_i}, \quad (2)$$

where $1 \le i \le N$ and $\mathrm{p_i}$ is the output probability of b_i.

When the primary inputs to a circuit are non-deterministic and thus each input occurs at a specific probability, the *mean error distance* (MED) of the circuit (denoted by d_m) is defined as the mean value of the EDs of all possible outputs for each input. Assume that the input is given by a set of vectors $(\mathrm{a}_1, \mathrm{a}_2, \ldots \mathrm{a}_\mathrm{M})$; each vector occurs with a probability given by a corresponding value in $(\mathrm{q}_1, \mathrm{q}_2 \ldots, \mathrm{q}_M)$. Then, the MED of the circuit is given by:

$$d_m = \sum_j \mathrm{d_j} * \mathrm{q_j}, \quad (3)$$

where $1 \le j \le M$ and $\mathrm{d_j}$ is the ED of the output for the input $\mathrm{a_j}$, which can be computed by (2).

TABLE 1. MEAN ERROR DISTANCE FOR FOUR CLOCK CYCLES WITH RANDOM INPUTS

Experiment	Architecture	Clk1	Clk2	Clk3	Clk4
No. 1	**CFA**	*0*	*0*	*0*	*0*
	PFA	*0.7400*	*1.1920*	*1.4620*	*2.1900*
	LOA	*0*	*2*	*1*	*1*
No. 2	**CFA**	*0*	*0*	*0*	*0*
	PFA	*0.7220*	*1.0260*	*1.5940*	*1.9640*
	LOA	*0*	*1*	*1*	*1*
No. 3	**CFA**	*0*	*0*	*0*	*0*
	PFA	*0.7820*	*1.2100*	*1.5180*	*2.2340*
	LOA	*0*	*2*	*1*	*0*
No. 4	**CFA**	*0*	*0*	*0*	*0*
	PFA	*0.6480*	*1.2860*	*1.5680*	*1.5980*
	LOA	*1*	*1*	*2*	*1*

For simplicity, uniformly-distributed random inputs are considered hereafter, i.e., each input occurs with the same probability. Consider as an example a 3-bit adder. In Fig. 4 and Fig. 5, let $k = 3$, $m = 1$ and $n = 2$. A 3-bit CFA is used to calculate the correct value for comparison. A gate error rate of 0.028 is used for PFA; this value is selected such that the MED is the same as for the LOA (as shown in subsequent sections). Table 1 shows the results of four experiments, each of which consists of four consecutive clock cycles. As can be seen, the MED of the LOA can be reduced or reset. For example, the MED has a value of 0 at both Clk1 and Clk4 in Experiment No. 3; this is due to the effect of error masking, as discussed in more detail next. However, the MED for the PFA is in most cases significantly greater than its LOA counterpart. As expected, the CFA has an MED of 0 throughout the four clock cycles in all experiments.

For uniformly distributed inputs, the 16 input values occur with the same probability of 1/16. Based on the S-PTM model for this 3-bit adder, the transition matrix Φ_{ns} is given in Fig. 8. Only the matrices for the lower two bits are shown as the higher bits are accurate and the same for these different implementations. Given the transition matrices, the ED of an output can be computed for an input against the corresponding output of the CFA (taken as the golden value). Given Φ_{LOA} in Fig. 8(b), for example, the MED of the LOA is:

$$d_{m,\mathrm{LOA}} = \frac{1}{16} * (0+0+0+0+0+1+0+1+0+0+2+2+0+1+2+1) = 0.625. \quad (4)$$

Similarly, the MED of the PFA is calculated as:

$$\mathrm{d_{m,PFA}} = \frac{1}{16} * \sum_j \mathrm{d_j} = 0.6167. \quad (5)$$

Consider next an implementation using only the higher two bits as the 3-bit adder, i.e., an approximate implementation with a reduced precision of one bit. The transition matrix is shown in Fig. 8(d) and its MED is given by:

$$d_{m(2-\text{bit Adder})} = \frac{1}{16} * \sum_j \mathrm{d_j} = \frac{1}{16} * (0+1+0+1+1+2+1+2+0+1+0+1+1+2+1+2) = 1. \quad (6)$$

This shows that the 2-bit adder has the largest MED, while the MED of the PFA (0.6167) is only slightly different from that of the LOA; therefore, 0.028 is used as error rate of the PFA in the previous simulation. However, from Table 1 the cumulated error distances are quite different. This is mainly due to a result of error masking. For the CFA (as well as the PFA), no adjacent 1's exist in the columns of $\boldsymbol{\Phi_{CFA}}$, i.e., there is no restoring input for CFA and PFA. Therefore, errors have a cumulative effect and cannot be logically masked. However, for the LOA several restoring inputs are found in $\boldsymbol{\Phi}_{\mathrm{LOA}}$ (indicated by the neighboring 1's in the columns), i.e., errors can be logically masked by specific inputs. This can be explained as follows. If there is a "1" at the primary input of an LOA, the OR gate masks all cumulative errors. This is not the case for the PFA because it is always affected by errors; therefore, errors accumulate rather than being masked.

	000	001	010	011	100	101	110	111
0000	1							
0001		1						
0010			1					
0011				1				
0100		1						
0101			1					
0110				1				
Φ_{CFA} = 0111					1			
1000			1					
1001				1				
1010					1			
1011						1		
1100				1				
1101					1			
1110						1		
1111							1	

(a)

	000	001	010	011	100	101	110	111
0000	1							
0001		1						
0010			1					
0011				1				
0100		1						
0101		1						
0110				1				
Φ_{LOA} = 0111				1				
1000			1					
1001				1				
1010							1	
1011								1
1100				1				
1101				1				
1110								1
1111								1

(b)

	000	001	010	011	100	101	110	111
0000	0.722	0.044	0.126	0.004	0.084	0.005	0.014	0.001
0001	0.044	0.722	0.004	0.126	0.005	0.084	0.001	0.014
0010	0.044	0.003	0.718	0.044	0.096	0.002	0.088	0.005
0011	0.003	0.044	0.044	0.718	0.002	0.096	0.005	0.088
0100	0.044	0.722	0.004	0.126	0.005	0.084	0.001	0.014
0101	0.083	0.004	0.765	0.044	0.012	0.001	0.086	0.005
0110	0.003	0.044	0.044	0.718	0.002	0.096	0.005	0.088
Φ_{PFA} = 0111	0.045	0.002	0.044	0.001	0.806	0.047	0.051	0.004
1000	0.043	0.003	0.718	0.044	0.097	0.002	0.088	0.005
1001	0.003	0.043	0.044	0.718	0.002	0.097	0.005	0.088
1010	0.043	0.002	0.002	0	0.763	0.047	0.138	0.005
1011	0.002	0.043	0	0.002	0.047	0.763	0.005	0.138
1100	0.003	0.043	0.044	0.718	0.002	0.097	0.005	0.088
1101	0.047	0.003	0.044	0.001	0.804	0.046	0.051	0.004
1110	0.002	0.043	0	0.002	0.047	0.763	0.005	0.138
1111	0.004	0.001	0.025	0.002	0.091	0.004	0.826	0.047

(c)

	000	001	010	011	100	101	110	111
0000	1							
0001	1							
0010			1					
0011			1					
0100	1							
0101	1							
0110			1					
Φ_{2-bit} = 0111			1					
1000			1					
1001			1					
1010					1			
1011					1			
1100			1					
1101			1					
1110					1			
1111					1			

(d)

Fig. 8. Transition matrices for the two lower bits of the 3-bit adder: (a) CFA (b) LOA (c) PFA and (d) 2-bit adder. Each row corresponds to an input, which consists of two bits from the primary inputs and two bits from the feedbacks; each column corresponds to a 3-bit output of the 2-bit addition. So, an entry in a matrix indicates a transition (probability) from an input to an output. In (d), "0" is used as the third output in default of the last bit.

V. Mean Error Distance (MED) Evaluation

As discussed previously, soft computing is confronted with many similarities and often contradictory features that can also be found in bio-inspired systems, i.e., systems made of a large number of unreliable modules. These systems utilize extensive networks to circumvent the unreliable nature of the computational modules while still retaining low power/energy consumption. The adder configurations presented previously draw significant resemblance to this type of system. The LOA resorts to approximate logic to target reliable modules and PFA resorts to characterizing probabilistic behavior of nanoscale modules. However, one of the issues that must be addressed for soft computing is that probabilistic implementations may likely tradeoff too much accuracy for little saving in area and power. In this section, this tradeoff is evaluated against a baseline design of a reduced precision adder, i.e., the lower-bit ignored adder (LIA). In this implementation, instead of using *n*-bit unreliable adders for computation, the lower significant bits are ignored.

Consider next the general case of a 32-bit adder (k=32); the MED is plotted in Fig. 9 by varying the number of lower bits, n. By ignoring all lower bits, as shown in Fig. 9, an n-bit ignored LIA has better performance than an n-bit PFA at a gate error rate of 0.2 for the 32-bit adder. Moreover, as shown in Fig. 9, the expected mean error distances of both the LOA and the PFA increase exponentially by increasing the number of lower bits. For the LOA, error masking occurs when the carry bit is ignored - there is no carry for the lower bits. Therefore, errors in different bits are operationally isolated. In the LOA by changing the truth table with approximate logic, error masking occurs in the combinational circuits. Based on this feature, errors in the lower bits cannot be propagated to the higher bits, thus preventing their cumulative effect.

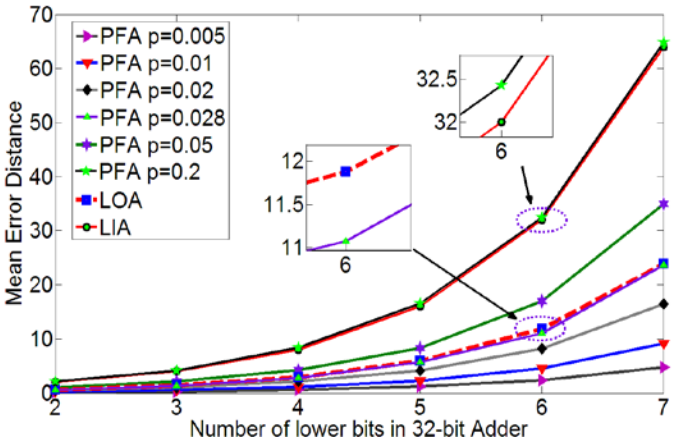


Fig. 9 Mean error distance versus the number of lower bits in different adder implementations.

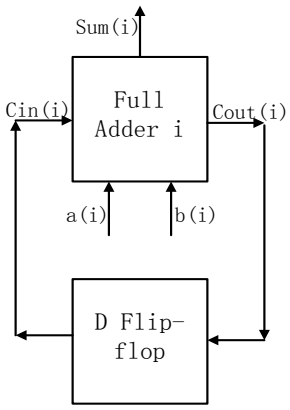


Fig. 10 A sequential adder implementation.

The simulation results of Fig. 9 show the effectiveness of the proposed evaluation technique. It is revealed that, although there is no cumulative error for the LOA, an error due to the approximate logic makes its MED to be not as optimistic as expected, especially for an increased number of lower bits. This is due to the fact that an error can occur in the most significant lower bit. The PFA shows a different scenario. Errors in the lower bits are likely to be propagated to the higher bits. So, its MED is rather large even for a small error rate, due to the cumulative errors from the lower bits (even though the error that results from the most significant bit is not that large).

LOA, PFA and LIA are based on an n-bit combinational adder. An n-bit ripple carry adder can be equivalently implemented by a sequential circuit based on a single-bit full adder, as shown in Fig. 10. In this circuit, Cout (Carry-out) in a lower significant bit is used as Cin (Carry-in) for a higher bit. So if occurring in the lower significant bits, errors can propagate to the higher significant bits, i.e., the cumulative effects of errors may make the higher significant bits to be less reliable than expected. However, the errors in Cout can be masked by primary inputs with values “00” and “11.” This is an effect of error masking. In a complex adder design (such as the LOA, PFA or LIA), error masking exists and its analysis must consider the multiple feedback signals. An efficient approach using stochastic bit streams can be used for such an analysis [10].

VI. Static Power Consumption and Delay

Consider next the static power consumption of the adders. As suggested in [4], the energy (E) consumed by a probabilistic inverter increases *exponentially* with the probability of its correct functioning, p, if the noise magnitude remains constant. For simplicity, here we assume that the energy consumption of any binary gate increases *exponentially* with p. It is further assumed that the static power of a 1-bit CFA is normalized to 1, i.e.,

$$E_{1-\mathrm{CFA}} = 1. \tag{7}$$

The static power of a 1-bit PFA is then:

$$E_{1-\mathrm{PFA}} = E_{1-\mathrm{CFA}} * \frac{e^p}{e^1} = e^{p-1}. \tag{8}$$

For a *k*-bit CFA, its static power is given by:

$$E_{\mathrm{CFA}} = k * E_{1-\mathrm{CFA}} = k. \tag{9}$$

Let *n* denote the number of lower bits; so, for the PFA, its static power is:

$$E_{\mathrm{PFA}} = m * E_{1-\mathrm{CFA}} + n * E_{1-\mathrm{PFA}} = m + n * e^{p-1}. \tag{10}$$

In a LOA, there is only one OR gate instead of five gates as in a conventional adder; so as a first-order estimation, the power consumption of the lower bits in a LOA is 1/5 of that in a CFA, i.e.,

$$E_{\mathrm{LOA}} = m * E_{1-\mathrm{CFA}} + n * E_{\mathrm{OR}} = m + 0.2 * n. \tag{11}$$

For the LIA, its power consumption is simply:

$$E_{\mathrm{LIA}} = m. \tag{12}$$

The delay in this paper is measured through the *critical path*, i.e., the longest path in a specific implementation of a sequential adder. It should be noted that due to the use of approximate logic, the length of the critical path is also changed from an exact implementation (as related to the rippling of the carry). The delays for a *k*-bit full adder implemented by CFA, PFA, LOA and LIA are given in Table 2 (the delay and the static power consumption are normalized to those of a single CFA). For comparison purposes, the reliable performance of an implementation can then be assessed at a given budget of static power consumption. Consider again a 32-bit adder (*k*=32); assume that the maximum static power is normalized to the single conventional adder and is given by a constant value of 28. Table 3 shows the MED of the different implementations. It can be seen that the MED of the PFA drastically increases with the number of lower unreliable bits, while for the same power consumption, the LIA has a significantly lower MED and the LOA performs the best with the smallest MED.

TABLE 2. COMPARISON OF DELAY AND STATIC POWER CONSUMPTION FOR DIFFERENT IMPLEMENTATIONS OF SEQUENTIAL ADDER

Implementation	Normalized Delay	Normalized Static Power Consumption
CFA	*k (=m+n)*	*k(=m+n)*
PFA	*k (=m+n)*	$m+n*e^{p-1}$
LOA	*m*	*m+0.2* n*
LIA	*m*	*m*

TABLE 3. MED OF DIFFERENT IMPLEMENTATIONS OF A 32-BIT ADDER WITH THE SAME STATIC POWER

Implementation	Features	MED
PFA	*p=0.1335: m=0, n=32*	$\sim 10^9$
	p=0.2 : m=10 , n=22	$\sim 10^6$
	p=0.3: m=17 , n= 15	$\sim 10^4$
	p=0.4: m=20 , n= 12	$\sim 10^3$
	p=0.5: m=22, n=10	*~500*
LOA	*m=27, n=5*	*5.875*
LIA	*n=4*	*8*

VII. CONCLUSION

This paper has assessed several sequential adder designs with respect to reliable performance for soft computing. A new metric referred to as mean error distance (MED) has been proposed to characterize the reliability of an approximate or probabilistic implementation. The MED has been used with sequential probability transition matrices (S-PTMs) in the presence of so-called error masking in the operation of different adder implementations under various input distributions. This paper has also dealt with the comparison of two representative implementations, the LOA and PFA. Based on the notion of MED, simulation results indicate that the reliable performance of PFA shows little advantage over the LOA, especially for large gate error rates, and that the LOA provides a better reliability-area tradeoff, due to the use of approximate logic and the presence of error masking. By ignoring or approximating the lower bits, the stringent timing feature of the critical path has been lessened, thus the LIA and LOA can be useful for higher speed computing. A detailed analysis and simulation results of these sequential adders have been presented to substantiate the practicality of these various implementations for soft computing.

REFERENCES

[1] Y. Dote, and S.J. Ovaska, "Industrial Applications of Soft Computing: a Review," Proc. IEEE, Vol. 89, no. 9, pp. 1243-1265, 2001.

[2] J. Huang and J. Lach, "Exploring the Fidelity-Efficiency Design Space using Imprecise Arithmetic," ASPDAC, 2011.

[3] H.R. Mahdiani, A. Ahmadi, S.M. Fakhraie, C. Lucas, "Bio-Inspired Imprecise Computational Blocks for Efficient VLSI Implementation of Soft-Computing Applications," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 57, no. 4, pp. 850-862, April 2010.

[4] S. Cheemalavagu, P. Korkmaz, K.V. Palem, B.E.S. Akgul, and L.N. Chakrapani, "A probabilistic CMOS switch and its realization by exploiting noise," in Proc. of In IFIP-VLSI SoC, Perth, Western Australia, Oct. 2005.

[5] J. Liang, J. Han and F. Lombardi, "Reliable Operation and Error Masking in Sequential Circuits," Internal Report, University of Alberta, 2011.

[6] A. K. Verma, P. Brisk, P. Ienne, "Variable latency speculative addition: A new paradigm for arithmetic circuit design," DATE, pp. 1250-1255, 2008.

[7] N. Zhu, W.L. Goh, K.S. Yeo, "An enhanced low-power high-speed adder for error tolerant application," ISIC, pp. 69-72, 2009.

[8] I. Polian, J.P. Hayes, S.M. Reddy and B. Becker, "Modeling and mitigating transient errors in logic circuits," IEEE Trans. on Dependable and Secure Computing, preprint, 2010.

[9] M. S. K. Lau, K. V. Ling, Y. C. Chu, A. Bhanu, "A General mathematical Model of Probabilistic Ripple-Carry Adders", in Proc. Des. Autom. Test Eur., 2010, pp. 1100–1105.

[10] H. Chen and J. Han, "Stochastic computational models for accurate reliability evaluation of logic circuits," in GLSVLSI'10, Proc. of the 20th Great Lakes Symposium on VLSI, Providence, Rhode Island, USA, pp. 61–66, 2010.