An Energy Efficient Approximate Adder with Carry Skip for Error Resilient Neuromorphic VLSI Systems

Yongtae Kim, Yong Zhang and Peng Li
Department of Electrical and Computer Engineering
Texas A&M University, College Station, TX 77843 USA
{fcore4, zhangyong, pli}@tamu.edu

ABSTRACT

We propose a novel approximate adder design to significantly reduce energy consumption with a very moderate error rate. The significantly improved error rate and critical path delay stem from the employed carry prediction technique that leverages the information from less significant input bits in a parallel manner. An error magnitude reduction scheme is proposed to further reduce amount of error once detected with low cost. Implemented in a commercial 90 nm CMOS process, it is shown that the proposed adder is up to 2.4× faster and 43% more energy efficient over traditional adders while having an error rate of only 0.18%. The proposed adder has been adopted in a VLSI-based neuromorphic character recognition chip using unsupervised learning. The approximation errors of the proposed adder have been shown to have negligible impact on the training process. Moreover, the energy savings of up to 48.5% over traditional adders is achieved for the neuromorphic circuit with scaled supply level. Finally, we achieve error-free operations by including a low-overhead error correction logic.

1. INTRODUCTION

Modern VLSI systems integrate many high-performance functional modules, such as multi-media and communication processors, thanks to the aggressive CMOS technology scaling. However, today's circuit designers are facing increasing challenges in managing chip power consumption. Recently, a new design paradigm of approximate-/soft-computing has emerged as one promising solution to remedy the energy-efficiency challenge [6, 17, 2]. The key observation is that many applications, such as digital signal processing (DSP) and neuromorphic systems, have inherent error resilience, and hence 100% precision in computation is not required. This provides opportunities for energy saving by relaxing computation accuracy while achieving an acceptable processing quality.

Particularly, the core of many DSP and neuromorphic applications lies in processing specific kernel functions, which occupy a significant portion of silicon area and computation time [15, 7]. For instance, MPEG motion estimation heavily performs L1-norm arithmetic for sum of absolute difference (SAD) calculation [18] and spiking neural networks use leaky integrate-and-fire (LIF) operations to mimic neuron behavior [8]. Clearly, adders are the primary component for building these arithmetic kernel functions. In this regard, it is particularly attractive to design approximate adders for considerable energy saving. Lu proposes an approximate adder [12] that leverages a limited number of previous (less significant) input bits for carry speculation to increase the overall speed. The critical drawback of this approach is the use of a considerable number of carry generators, which gives rise to large area and high power dissipation. The ETAI [21] and LOA [13] are split into an accurate part for higher order bits and an inaccurate part, which utilizes a modified XOR (ETAI) and OR function (LOA) to approximately compute the remaining lower bits. A few transistors are eliminated from the traditional mirror adder to reduce power and area at the expense of accuracy degradation in [5]. Those two approaches are limited by high error rates. The segment based approximate adders are presented in [20] and [3] which are named ETAII and VLCSA-1, respectively. The carry for each k-bit segment is predicted from the lower k-bit inputs to reduce the delay of carry propagation. Similarly, the ACA [9] adopts a number of $2k$-bit sub-adders and leverages only k most significant bit (MSB) outputs of the sub-adders to achieve approximate additions. Unfortunately, these adders have high error rates for the carry generations, particularly for 2's complement signed additions of small numbers. In addition, the use of carry selection in VLCSA-1 and middle sub-adders in ACA result in power consumption and area overhead. The lack of an error magnitude reduction in ETAII degrades the quality of addition. In [14], the approximation errors for less significant bits are reduced by conditional bounding logic with dithering, which causes delay and area overhead.

In this paper, we propose a novel approximate adder with a carry skip scheme. While reducing the worst-case carry propagation delay, this carry skip scheme allows for highly accurate carry prediction, making it possible to either speed up addition operations, or reduce energy dissipation by lowering the supply level. The significantly improved error rate and critical path delay stem from the employed carry prediction technique that leverages the information from less significant input bits in a parallel manner. An error magnitude reduction scheme is proposed to further reduce amount of error once detected with low cost. Our adder design is rather flexible in the sense that a low-overhead error correction logic can be readily included to achieve error-free operations at the cost of one additional clock cycle. Implemented in a commercial 90 nm CMOS process, it is shown that the proposed adder is up to 2.4× faster and 43% more energy efficient over traditional adders while having an error rate of only 0.18%. To evaluate the performance of the proposed adder under neuromorphic applications, we develop a behavioral evaluation approach for a VLSI-based neuromorphic character recognition chip using unsupervised learning. The approximation errors of the proposed adder have been shown to have negligible impact on the training process while other approximate adders lead to unacceptable level of performance degradation. Furthermore, the proposed adder enables the energy reductions of up to 48.5% over traditional

978-1-4799-1071-7/13/$31.00 ©2013 IEEE

adders on the digital neuron circuit in the scaled supply.

2. PROPOSED APPROXIMATE ADDER

In the approximate adder design, our key contributions are (1) a significant reduction of the error rate by the carry skip scheme enabling carry speculation in a parallel manner, and (2) a very low-cost error magnitude reduction without additional clock cycle scheme.

2.1 Approximate Adder Architecture

Denote the two inputs of the adder A and B, and the (i)th least significant bits (LSBs) by a_i and b_i, respectively. In addition, the propagate (p_i), generate (g_i), kill (k_i), and carry (c_i) signals of the (i)th bit position are defined by

$$g_i = a_i b_i,\ \ k_i = \bar{a_i}\bar{b_i},\ \ p_i = a_i \oplus b_i$$
$$c_i = \begin{cases} 1 & if\ \ g_i = 1 \\ 0 & if\ \ k_i = 1 \\ c_{i-1} & if\ \ p_i = 1 \end{cases} \quad (1)$$

where c_{i-1} is the carry of the (i-1)th bit position. Briefly, the adder outputs the carry c_i when g_i=1 or k_i=1 independently of c_{i-1}, otherwise, it propagates c_{i-1} to c_i.

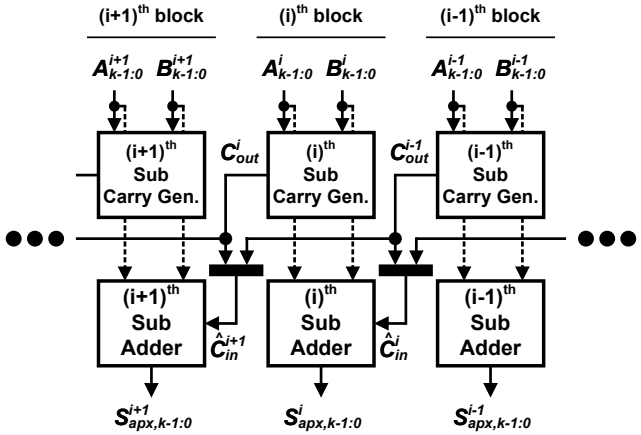

Figure 1: Block diagram of the proposed approximate adder.

Fig. 1 shows the block diagram of the proposed approximate n-bit adder, which is divided into several k-bit sized blocks. Each block contains a k-bit sub-adder and a k-bit sub-carry generator, which create a partial summation and a partial carry-out signal, respectively. The n-bit adder has $m = \lceil \frac{n}{k} \rceil$ blocks. Also, as in Fig. 1, the k-bit inputs of the (i)th block are represented by $A^i_{k-1:0}$ and $B^i_{k-1:0}$, and the partial summation result is indicated by $S^i_{apx,k-1:0}$. Note that the sub-adders could be implemented by any traditional accurate adders such as ripple-carry adder (RCA) and carry-lookahead adder (CLA). At the beginning of an addition operation, all the sub-carry generators simultaneously create the partial carry-out signals ($\cdots$, C^{i+1}_{out}, C^i_{out}, C^{i-1}_{out}, $\cdots$) using only their k-bit inputs. Then, the sub-adders' carry-in signals ($\cdots$, $\widehat{C}^{i+1}_{in}$, $\widehat{C}^i_{in}$, $\widehat{C}^{i-1}_{in}$, $\cdots$) are also concurrently speculated from the two preceding k-bit sub-carry generators with a multiplexer. Finally, the sub-adders work with the speculated carries and produce the partial summations ($\cdots$, $S^{i+1}_{apx,k-1:0}$, $S^i_{apx,k-1:0}$, $S^{i-1}_{apx,k-1:0}$, $\cdots$). Therefore, the critical path delay of the proposed approximate adder t_{apx} is derived by

$$t_{apx} = t_{sa} + t_{mux} + t_{scg} \quad (2)$$

where t_{sa}, t_{mux} and t_{scg} are the delays of the sub-adder, the multiplexer, and the sub-carry generator, respectively. Note that the multiplexer delay is negligible if k is large.

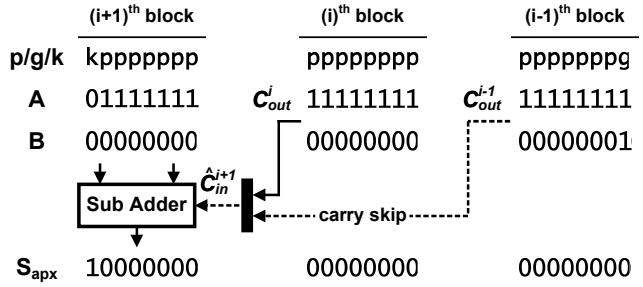

Figure 2: Proposed carry prediction using parallel carry skip.

The proposed carry prediction works as follows. When all the propagate signals of the (i)th block are true, the carry-out of the (i-1)th block is required for more accurate carry prediction. Thus, we utilize carry skip to speculate the carry as depicted in Fig. 2. This carry skip scheme is particularly more advantageous over the alternative approach of cascading two sub-carry generators, which could appreciably increase the critical path delay when k is large. In order to obtain the (i+1)th carry-in $\widehat{C}^{i+1}_{in}$, the multiplexer selects C^{i-1}_{out} if all the propagate signals of the (i)th block are true as in Fig. 2, otherwise, it chooses C^i_{out}. Hence, $\widehat{C}^{i+1}_{in}$ is expressed by

$$\widehat{C}^{i+1}_{in} = \overline{P^i_{k-1:0}} C^i_{out} + P^i_{k-1:0} C^{i-1}_{out}$$
$$where \quad P^i_{k-1:0} = \prod_{j=0}^{k-1} p^i_j \quad (3)$$

In (3), C^i_{out} and C^{i-1}_{out} are the carry-out signals of the (i)th and the (i-1)th blocks, respectively, and p^i_j is the propagate signal of the (j)th bit position of the (i)th block. Additionally, the carry-out of the (i)th block is given by

$$C^i_{out} = g^i_{k-1} + g^i_{k-2} p^i_{k-1} + ... + g^i_0 \prod_{j=1}^{k-1} p^i_j \triangleq G^i_{k-1:0} \quad (4)$$

where g^i_j is the generate signal at (j)th bit position of the (i)th block.

By adopting the carry skip scheme, the proposed adder is able to enhance the carry prediction accuracy at the cost of one multiplexer delay. It is important to note that this carry skip scheme is rather general. A larger number of preceding sub-carry generators can be connected in the same parallel way to further improve the accuracy of carry prediction, at a low cost of one multiplexer delay per each included generator. In the rest of this paper, we focus on the case of Fig. 2 since it already has a very low rate of error.

2.2 Error Analysis

The carry prediction error of the proposed adder occurs when a carry propagation chain has a length greater than $2k$. In other words, if all the propagate signals of more than two consecutive blocks are true and a carry is generated in the preceding block, then the carry prediction is incorrect. Assuming that the adder inputs A and B are bitwise independent, then the propagate and generate signals are bitwise independent as well. We denote the event that the carry-in

prediction of the (i)th sub-adder is mistaken due to a carry propagation path of a length between $2k$ and $3k-1$ by E^i_{cin}:

$$E^i_{cin} = P^{i-1}_{k-1:0} P^{i-2}_{k-1:0} G^{i-3}_{k-1:0} \tag{5}$$

where $P^i_{k-1:0}$ and $G^i_{k-1:0}$ are defined in (3) and (4), respectively and the probability of the event is given by

$$\begin{aligned}\mathbf{P}(E^i_{cin}) =& \mathbf{P}(P^{i-1}_{k-1:0} P^{i-2}_{k-1:0} G^{i-3}_{k-1:0}) \\ =& \mathbf{P}(P^{i-1}_{k-1:0}) \mathbf{P}(P^{i-2}_{k-1:0}) \mathbf{P}(G^{i-3}_{k-1:0})\end{aligned} \tag{6}$$

In (6), $\mathbf{P}(P^i_{k-1:0})$ [$= \mathbf{P}(P^{i-1}_{k-1:0}) = \cdots$] and $\mathbf{P}(G^i_{k-1:0})$ [$= \mathbf{P}(G^{i-1}_{k-1:0}) = \cdots$] are given by

$$\begin{aligned}\mathbf{P}(P^i_{k-1:0}) &= \mathbf{P}(\prod_{j=0}^{k-1} p^i_j) = \prod_{j=0}^{k-1} \mathbf{P}(p^i_j) = \frac{1}{2^k} \\ \mathbf{P}(G^i_{k-1:0}) &= \mathbf{P}(g^i_{k-1} + g^i_{k-2} p^i_{k-1} + \cdots + g^i_0 \prod_{j=1}^{k-1} p^i_j) \\ &= \mathbf{P}(g^i_{k-1}) + \mathbf{P}(g^i_{k-2} p^i_{k-1}) + \cdots + \mathbf{P}(g^i_0 \prod_{j=1}^{k-1} p^i_j) \\ &= \frac{1}{4} + \frac{1}{4} \cdot \frac{1}{2} + \cdots + \frac{1}{4} \cdot \frac{1}{2^{k-1}} = \frac{1}{2}\left(1 - \frac{1}{2^k}\right)\end{aligned} \tag{7}$$

where g^i_{k-1}, $g^i_{k-2} p^i_{k-1}$, $\cdots$, $g^i_0 \prod_{j=1}^{k-1} p^i_j$ are mutually exclusive. The proposed adder produces an error if any error event E^i_{cin} occurs for any of the sub-adders except for the three least significant ones. Note that the (0)th, (1)st and (2)nd sub-adders always have the correct carry-in signals in our design. Thus, the overall error rate of the proposed adder under random inputs is expressed by

$$\mathbf{P_{err}}(n,k) = \mathbf{P}(E^{\lceil \frac{n}{k} \rceil - 1}_{cin} + E^{\lceil \frac{n}{k} \rceil - 2}_{cin} + \cdots + E^4_{cin} + E^3_{cin}) \tag{8}$$

By the inclusion-exclusion principle [1], it is given by

$$\begin{aligned}\mathbf{P_{err}}(n,k) = & \sum_{3 \le i \le m} \mathbf{P}(E^i_{cin}) \\ & - \sum_{3 \le i_1 < i_2 \le m} \mathbf{P}(E^{i_2}_{cin} E^{i_1}_{cin}) \\ & + \sum_{3 \le i_1 < i_2 < i_3 \le m} \mathbf{P}(E^{i_3}_{cin} E^{i_2}_{cin} E^{i_1}_{cin}) \\ & - \cdots + (-1)^{m-1} \mathbf{P}(E^m_{cin} E^{m-1}_{cin} \cdots E^3_{cin}) \\ & \textit{where} \;\; m = \lceil n/k \rceil - 1\end{aligned} \tag{9}$$

Once E^i_{cin} occurs, neither E^{i-1}_{cin} nor E^{i-2}_{cin} can do. This is because that under this case the carry propagate chain lengths for the (i-1)th and (i-2)th sub-adders become less than $2k$ due to $P^{i-2}_{k-1:0} = G^{i-3}_{k-1:0} = 1$ and thus the carry speculations for these sub-adders are always correct. In short, $\mathbf{P}(E^{i_r}_{cin} \cdots E^{i_1}_{cin}) = 0$ if $\exists q : i_q - i_{q-1} < 3$ where $3 \le i_1 < \cdots < i_r \le \lceil \frac{n}{k} \rceil - 1$. Then, we can rewrite (9) to yield

$$\begin{aligned}\mathbf{P_{err}}(n,k) = & \sum_{r=1}^{m-2} (-1)^{r+1} \left(\sum_{\substack{3 \le i_1 < \cdots < i_r \le m, \\ \forall q : i_q - i_{q-1} \ge 3}} \mathbf{P}(E^{i_r}_{cin} \cdots E^{i_1}_{cin}) \right) \\ & \textit{where} \;\; m = \lceil n/k \rceil - 1\end{aligned} \tag{10}$$

$E^{i_r}_{cin}, E^{i_{r-1}}_{cin}, \cdots E^{i_1}_{cin}$ are independent if $\forall q : i_q - i_{q-1} \ge 3$. Therefore, by putting (6), (7) and (10) together, the overall error rate for the adder under random inputs is

$$\begin{aligned}\mathbf{P_{err}}(n,k) &= \sum_{r=1}^{m-2} (-1)^{r+1} \left(\sum_{\substack{3 \le i_1 < \cdots < i_r \le m, \\ \forall q : i_q - i_{q-1} \ge 3}} \mathbf{P}(E^{i_r}_{cin}) \cdots \mathbf{P}(E^{i_1}_{cin}) \right) \\ &= \sum_{r=1}^{m-2} (-1)^{r+1} \left(\sum_{\substack{3 \le i_1 < \cdots < i_r \le m, \\ \forall q : i_q - i_{q-1} \ge 3}} \left(\frac{1}{2^{2k+1}} \left(1 - \frac{1}{2^k}\right) \right)^r \right) \\ & \textit{where} \;\; m = \lceil n/k \rceil - 1\end{aligned} \tag{11}$$

Note that the error rate for cases where more than two preceding blocks are used in the carry skip scheme can be derived in a similar way, which is further improved.

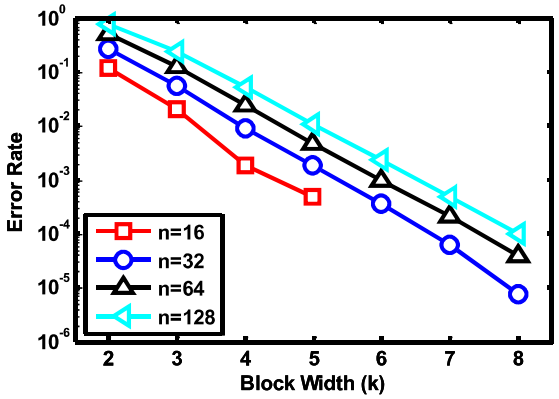

Figure 3: Error rates under different n and k.

Fig. 3 exhibits the error rates of the proposed adder with the different n and k. Under the case of n=16 and k=4, compared to the previously presented approximate adders [20, 3, 9], the proposed adder considerably reduce the error rate from 5.86% to 0.18% for random input patterns.

2.3 Error Magnitude Reduction

In addition to error rate, another important metric to evaluate approximate adders is error significance, which should be minimized and is defined by the ratio of the error magnitude to the correct summation result as follows [16]

$$\textit{error significance} = \left| \frac{S_{apx} - S_{cor}}{S_{cor}} \right| \tag{12}$$

where S_{apx} and S_{cor} are the approximate and correct outputs for given inputs.

Fig. 4 depicts the block diagram of the proposed error magnitude reduction and one example of its operation. In the example, since all the propagate signals of the (i)th and

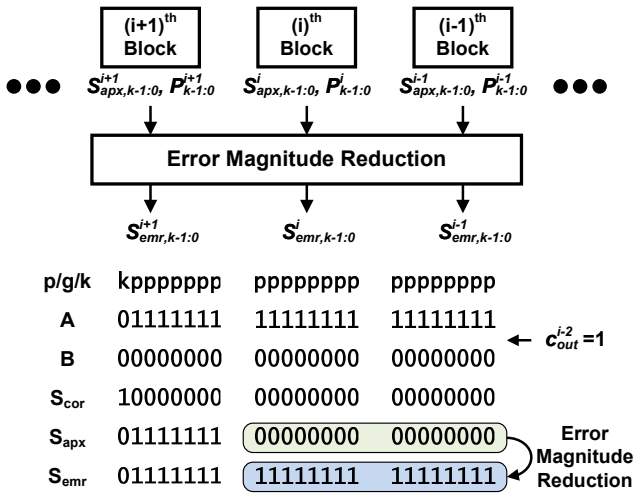

Figure 4: Block diagram of the error magnitude reduction and an example of its operation.

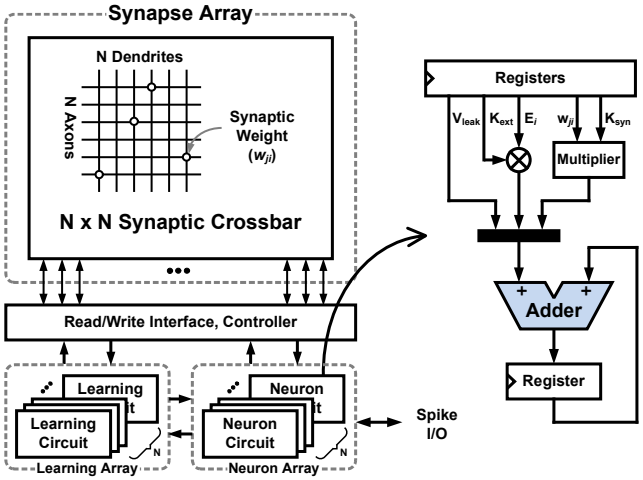

Figure 5: Block diagram of a digital neuromorphic VLSI system and implementation of LIF neurons.

(i-1)th blocks are true, the carry-in for the (i+1)th sub-adder is speculated to "0" although the correct one is "1" due to $C_{out}^{i-2} = 1$. Then, the error significance is $\frac{1}{2^7}$. Note that it could reach $\frac{1}{2}$ for the worst case inputs of $A_{7:0}^{i+1} = 00000001$ and $B_{7:0}^{i+1} = 00000000$. To reduce the amount of error, the proposed adder forces all the output bits of the (i)th and the (i-1)th sub-adders to "1" when $P_{k-1:0}^{i} = P_{k-1:0}^{i-1} = 1$. The reduction can be implemented by ORing each partial summation (i.e. $S_{apx,k-1:0}^{i}$ and $S_{apx,k-1:0}^{i-1}$) and the product of the propagate signals (i.e. $P_{k-1:0}^{i} P_{k-1:0}^{i-1}$). It allows the error significance to be reduced by $\frac{1}{2^{2k}}$. As a result of the reduction, the adder finally produces the error reduced output of $S_{emr,k-1:0}^{i} = S_{emr,k-1:0}^{i-1} = 11111111$ and the error significance decreases from $\frac{1}{2^7}$ to $\frac{1}{2^{23}}$. It is worth mentioning that the error magnitude reduction always produces the exact right results when $C_{out}^{i-2} = 0$. Consequently, the worst case error magnitude is reduced from 2^{n-k} to 2^{n-3k} through the use of error magnitude reduction.

3. NEUROMORPHIC VLSI SYSTEM

We briefly introduce the neuromorphic VLSI system that is our primary target application of this paper.

3.1 Overall Hardware Architecture

Fig. 5 depicts the block diagram of a general digital neuromorphic hardware architecture for spiking neural networks. It consists of three arrays of synapse, learning, and neuron circuits as well as a control and an interface circuits for them. The $N \times N$ crossbar array can represent a fully recurrent network topology and store N^2 possible synaptic weights among N neurons. It could be implemented with either traditional CMOS memory cells or emerging nanodevices such as memristors [8, 11]. An axon (output) and a dendrite (input) of a biological neuron correspond to a row and a column of the array. The connection between the (j)th row and the (i)th column keeps a synaptic weight between neuron j and i, which is represented by w_{ji}, on the crossbar. To mimic the behavior of a biological neuron, each neuron circuit, which can be either excitatory or inhibitory, emulates the neuron dynamics (e.g. according to LIF and Hodgkin-Huxley models) and generates a spike when it fires. The learning circuits cooperate with the respective neurons and update the synaptic weights according to a learning-rule, such as spike timing dependent plasticity (STDP).

3.2 Digital Leaky Integrate and Fire Neuron

Among the various neuron models, the LIF model is suitable for digital implementation with a few arithmetic components and widely used [8, 11]. Its dynamics can be indicated by [11]

$$V_i^{t+1} = V_i^t + K_{syn} \sum_{j=1}^{M} w_{ji} S_j^t + K_{ext} E_i^t - V_{leak}$$
$$S_i^{t+1} = \begin{cases} 1 & if \ V_i^{t+1} > V_{th} \\ 0 & otherwise \end{cases} \quad (13)$$

where V_i^t is the membrane potential of neuron i at time t, S_i^t is the spike bit that indicates whether neuron i fired at time t and is set to "1" when the membrane potential exceeds the given threshold voltage V_{th}, w_{ji} is the synaptic weight between neuron j and i, E_j^t is the spike bit for the external input for neuron i, M is the number of pre-synaptic neurons, V_{leak} is the leaky potential, and K_{syn} and K_{ext} are the weight parameters for synapses and external input spikes, respectively. A digital implementation of the LIF neuron is also shown in Fig. 5. It contains both a multiplier and an adder. The adder dominates the computation time because the multiplier is relatively small due to narrower bit-widths in multiplications. If each synaptic weight, model parameter and membrane potential are represented using 3, 5 and 16-bits, respectively, the addition operations contribute to 32% of the power and 73% of the processing time for the LIF computation with RCA. When implemented with CLA, the add operations consume 56% of both the power and processing time of the computation. Thus, it is important to reduce the delay and power of the adder to improve energy efficiency of neuromorphic computing.

3.3 Evaluation Environment

Evaluating the performance of proposed adder design by simulating the long training process of neuromorphic systems at the transistor level is computationally intractable. Instead, we develop a hardware-aware spiking neural network simulator for neuromorphic systems. The key network features and hardware design parameters including the

digital LIF neuron dynamics, the STDP learning rule, bit-widths used to represent various neuron model parameters are fully captured in the simulator. The proposed approximate adder is carefully characterized and its circuit profiles are extracted from HSPICE simulations [10]. To evaluate the approximate nature of our adder, we disable the error correction logic and inject the characterized input-specific adder error into each add operation in the behavioral simulator, providing a precise evaluation of the impacts of the adder error for neuromorphic computation.

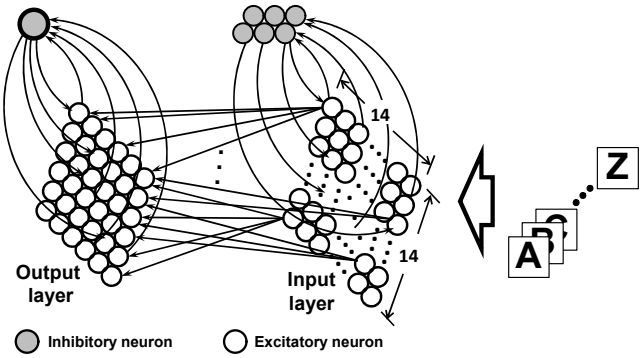

Figure 6: Network for character recognition.

We specifically consider the case where the neuromorphic hardware is configured to be a two-layer network for character recognition as illustrated in Fig. 6 [4]. The input and output layers have 196 and 36 excitatory neurons, respectively. Each excitatory input neuron receives a binary input representing the value of a pixel in a 14×14 pixel input pattern, and projects its output to all excitatory output neurons through plastic synapses. In the input layer, 6 inhibitory neurons are employed to modulate the firing frequencies of excitatory neurons by providing negative feedback. The inhibitory neuron in the output layer provides strong negative feedback to implement the winner-take-all (WTA) mechanism. To train the network, 26 input patterns of alphabets "A" – "Z" are applied one by one to the input layer. Before training, weights of all plastic synapses are random values. Thus, the input to each excitatory output neuron is the inner product of the signal vector representing the activity of excitatory input neurons and a random weight vector. The weight vector describes the receptive field, i.e. the area in which the presence of stimulus leads to excitation of the corresponding output neuron. During training, the network learns each alphabet by reshaping the receptive fields of certain excitatory output neurons such that these neurons receive strong inputs and generate spikes when the corresponding input pattern is applied. In this network, we use 3, 5 and 16-bits respectively to represent each synaptic weight, model parameter and the membrane potential for each neuron and employ a 16-bit adder for the LIF computation.

4. SIMULATION RESULTS

The proposed approximate adder was designed in Verilog HDL and synthesized with a commercial 90 nm CMOS technology and standard cell library. Also, the gate-level netlist was translated into transistor-level to perform HSPICE simulations. Each sub-adder was implemented using an RCA structure.

4.1 Performance of the Proposed Approximate Adder

First, we examine the implementations for the proposed adder with various values of n and k. Table 1 reports the results of area, delay, power, and error rate under the nominal supply of 1.2 V. Note that the error magnitude reduction circuit is included in all the proposed design implementations. The delay increases as k increases with a fixed n while the error rate and the power decrease. With a lower k for a given n, more carry prediction blocks are needed and the carry prediction with a smaller number of LSBs causes more errors. Meanwhile, under a given k, while the delay remains almost the same as n increases, the error rate increases slowly. The proposed 128-bit adder with k=4 has an error rate even slightly less than the error rate of 5.86% of other 16-bit approximate adders with the same k [20, 3, 9]. Hence, the proposed design can be equally well used for wide bit-width additions.

Table 1: Proposed adder with different n and k.

parameters (n, k)	*area* (μm^2)	*delay* (ps)	*power* (mW)	*error rate* (%)
(16, 2)	525	202	0.902	11.55
(16, 3)	533	297	0.708	2.05
(16, 4)	466	359	0.600	0.18
(16, 5)	509	431	0.591	0.05
(32, 4)	1147	345	1.682	0.91
(64, 4)	2389	344	3.039	2.36
(128, 4)	4873	339	5.827	5.19

4.2 Comparison with Six Other Approximate Adders

We also implemented two traditional accurate adders (RCA and CLA) and six previously presented approximate adders, which are Lu's Adder (LUA) [12], LOA [13], ETAI [21], ETAII [20], VLCSA-1 [3] and ACA [9], in the same commercial 90 nm CMOS technology, so as to compare with the proposed adder in area, delay and power aspects. The VLCSA-1 and ACA have their own error detection and correction mechanisms. The invoking of these modules, however, requires additional clock cycles, leading to timing overhead and potential architectural design complications needed for facilitating a given processing application. To examine the approximate natures of the different adder designs, to be fair, we exclude the error detection and correction modules and their timing, power and area overheads from this comparison. The same RCA structure is used for the sub-adders in ETAII, VLCSA-1, ACA and the proposed adder and the parameters of n=16 and k=4 are adopted in these adders as well as LUA. Moreover, we split LOA and ETAI to have both 8-bit sizes for the accurate and inaccurate parts and RCA structure is employed for the accurate parts. We denote these two adders by LOA (N-M) and ETAI (N-M), where N and M indicate the bit widths of the accurate and inaccurate parts, respectively. Table 2 summarizes the performance comparison under the regular supply of 1.2 V. The RCA exhibits the lowest power with the longest delay due to the bit-by-bit carry propagate chain and the CLA consumes the largest energy. The LUA is the fastest but occupies the second largest area due to the considerable number of carry generators. The error rate of EATI (8-8) reaches 90%, which may limit its practical use, due to lack of carry prediction for the accurate part (i.e. the carry is fixed to zero). On the other hand, thanks to the simple carry speculation scheme of LOA (8-8), which is achieved by ANDing two MSBs of each operand of the inaccurate part, the error rate is improved to 43.75%, which is still fairly large. The use of the simple OR operation for the inaccurate part allows it to be

Table 2: Comparison with other 16-bit adders.

	area (μm^2)	*delay* (ps)	*power* (mW)	*energy* (pJ)	*error rate* (%)
RCA	334	856	0.343	0.294	0.00
CLA	514	407	0.922	0.375	0.00
LUA	609	234	0.908	0.212	16.68
LOA (8-8)	200	450	0.420	0.189	43.75
ETAI (8-8)	234	435	0.470	0.204	90.00
ETAII	374	254	0.564	0.143	5.86
VLCSA-1[1]	673	277	1.337	0.370	5.86
ACA[1]	472	374	0.666	0.249	5.86
Proposed[2]	466	359	0.600	0.215	0.18

[1] without the error detection and correction
[2] with the error magnitude reduction

the most area efficient adder. Among the adders having the same error rate of 5.86%, the ETAII is the most efficient in area, delay, power and energy aspects. As a result of the use of carry selection in VLCSA-1, it dissipates the highest power, which is up to 3.9× more than the others. The carry skip scheme allows the proposed adder to have the lowest error rate of 0.18% among the approximate adders and to be 2.4× faster than RCA. Our design is comparable to ACA with respect to area, delay, power and energy while having much lower error rate thanks to carry skip.

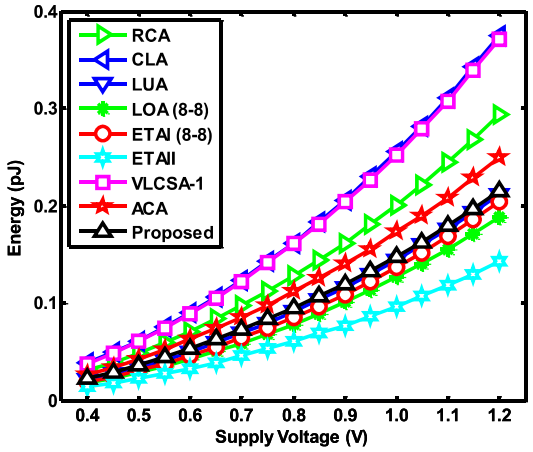

Figure 7: Energy comparison under supply scaling.

Fig. 7 plots the energy comparison under scaled voltages. The energy efficiency of VLCSA-1 is hindered by a high power dissipation in spite of its relatively fast speed. It takes almost the same amount of energy as CLA that consumes the highest energy, whereas the ETAII does the lowest among the adders. The good tradeoff between delay and power obtained by carry skip allows our adder to be more energy efficient than the two accurate adders, VLCSA-1 and ACA. Particularly, our design attains an energy saving of 27% and 43% compared to RCA and CLA, respectively. Besides, the proposed adder, LUA, LOA (8-8) and ETAI (8-8) show similar energy consumptions under the scaled voltages while our design enjoys the lowest error rate.

4.3 Comparison on Error-Free Operations

The main objective of this work is to develop an efficient approximate adder with low error rate for neuromorphic applications. For completeness, we also compare the error-free operations of various designs. We consider the error detection and correction schemes for VLCSA-1, ACA and the proposed adder. In each of these designs, the error detection is achieved by checking the propagate and generate signals in the approximate addition phase. Upon detecting an error, the error correction circuit reconstructs accurate results by leveraging propagate and generate signals [19, 3] or by adding "1" to sub-adder output [9], either of which requires an additional clock cycle. The VLCSA-1 and ACA exploit the prefix adder and incrementor, respectively, for error correction. For the proposed adder, the prefix adder based error correction circuit is implemented to produce error-free results [19, 3]. Note that the error magnitude reduction of our adder is not necessary here and thus removed in this implementation. Obviously, the error correction circuit is activated whenever errors are detected in the addition phase [9] and the effective energy E_{eff} can be expressed by

$$E_{eff} = P_{apx}t_{apx} + \mathbf{P_{err}} \cdot P_{ec}t_{ec} \quad (14)$$

where P_{apx}, t_{apx}, P_{ec} and t_{ec} are the power and delay of the approximate adder and those of the error correction circuit, respectively, and $\mathbf{P_{err}}$ is the error rate of the approximate adder. The implementations are summarized in Table 3. The error correction circuits have shorter delays than the respective approximate adders. The critical path delays of VLCSA-1 and ACA are slightly longer than the delays in Table 2 due to the additional error detection logic. Conversely, the proposed adder's critical path delay becomes shorter in spite of the error detection circuit since the error magnitude reduction block is eliminated. Our design occupies the lowest area and is the most efficient adder in terms of power and effective energy.

Table 3: Approximate adders with error detection and correction.

	area (μm^2)	*critical path delay* (ps)	*power* (mW)	*effective energy* (pJ)
VLCSA-1	899 (71.7%)[1]	296 (-7.0%)	2.094 (137.8%)	0.473 (142.1%)
ACA	580 (10.8%)	389 (22.1%)	1.269 (44.1%)	0.287 (46.9%)
Proposed	524	318	0.881	0.195

[1] (*) overheads against the proposed adder

4.4 Impacts of adder errors on the Neuromorphic Application

We use the neuromorphic application of Fig. 6 and the evaluation environment described in Section 3.3 to systematically examine the impacts of adder errors of several designs. First, we fix the supply level to 1.2 V so and clock the chip at the nominal clock rate of 100 MHz so that the errors produced are only due to the approximate natures of the adders since there is no timing failure. Fig. 8 shows the receptive fields of all excitatory output neurons after the training with the various adders and the corresponding error rates during the training process are depicted in Fig. 9 as well. The receptive fields with the accurate adders (RCA and CLA) as in Fig. 8(a) are trained well to respond to the inputs from "A" to "Z". This means that every letter appears once at least in the receptive fields. The results in Fig. 8(a) serves as a golden reference for the approximate adders.

The proposed adder with n=16 and k=4 has less than 1% error rate for the LIF computations during the training. Note that no error detection and correction is considered but the error magnitude reduction is. Fortunately, thanks to the error resilience of the neuromorphic system, Fig. 8(b) shows

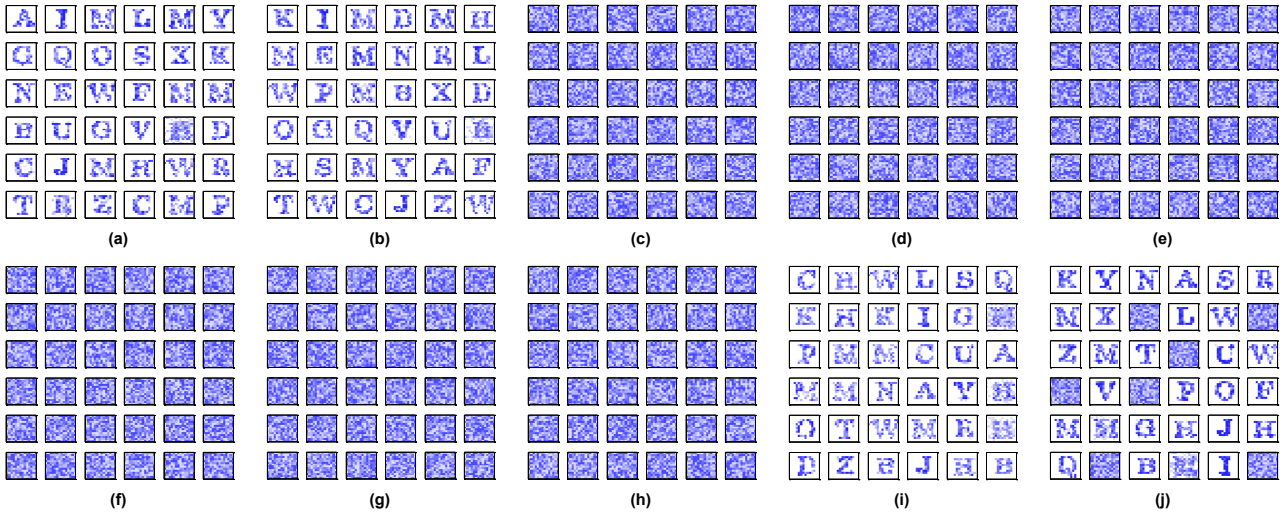

Figure 8: Receptive fields with 16-bit (a) RCA/CLA, (b) proposed approximate adder, (c) LUA, (d) LOA (8-8), (e) ETAI (8-8), (f) ETAII, (g) VLCSA-1, (h) ACA, (i) LOA (13-3) and (j) ETAI (15-1).

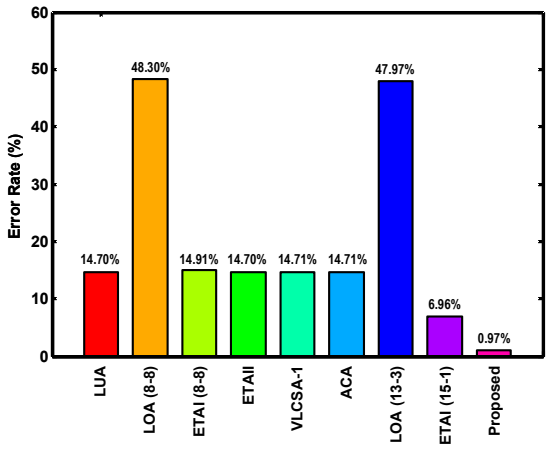

Figure 9: Error rates during training process.

that the receptive fields are trained successfully to recognize all the letters and the approximation errors have negligible effect on the training process of the character recognition system. We also test the other approximate adders with the network. For LOA and ETAI, the 8-bit accurate and inaccurate parts are used (i.e. LOA (8-8) and ETAI (8-8)). n=16 and k=4 are adopted and only the approximate adder part (i.e. without the error detection and correction) is utilized for VLCSA-1 and ACA. These adders have an error rate of more than 14%. Especially, the error rate of LOA (8-8) reaches 48.30% throughout the training. As seen in Fig. 8(c) – (h), the approximate adders produce a set of receptive fields with random synaptic weights. These high error rates give rise to failures in training the network since the approximation errors cause the neurons to either fire randomly or cease to fire. In particular, the 2's complement signed additions of small numbers frequently occur for the training. In this case, the LUA, ETAII, VLCSA-1 and ACA produce many wrong carry predictions and unacceptable performance degradation. This result suggests the carry speculation with only 4-bit of less significant inputs in these 16-bit adders might be insufficient for this application. To shed more light on this, we increase the accuracy of LOA and ETAI by expanding the accurate part of the adder at the cost of increased delay and energy dissipation. When the LOA and ETAI have 13 and 15-bits accurate parts, respectively, the network starts to perform better. Although the LOA (13-3) still has a relatively high error rate of 47.97%, the corresponding receptive fields as in Fig. 8(i) are trained such that all alphabets except for "R", "V" and "X" can be identified. Due to the expansion of the accurate part of the adder, the errors now concentrate more on LSBs with smaller error magnitudes. Similarly, the inclusion of a 15-bit accurate part in ETAI (15-1) allows the network to be trained for all letters except for "C", "D" and "E" as illustrated in Fig. 8(j). Clearly, our design outperforms all other approximate adders.

To show the energy efficiency of the these adders in the neuromorphic hardware, we scale down the supply voltage and obtain the energy dissipation in one LIF operation involving a multiplication of a synaptic weight w_{ji} with the weight parameter K_{syn} and an addition of the membrane potential V_i^t with the multiplier output $K_{syn}w_{ji}$, a key processing step in (13). The clock frequency is fixed at the maximum value such that neurons with RCA can operate without any error in the regular supply voltage of 1.2 V. For each adder design, we scale down the supply voltage with a 0.05 V step as long as there is no critical timing failure created in the neuron circuit. Fig. 10 plots the energy comparison with the neurons with the different adders under scaled power supply levels. The energies are normalized against the neurons with RCA. Neurons with LUA, ETAII or VLCSA-1 can operate at a supply voltage of 0.9 V. The ETAII is the most energy efficient design while having a much larger error rate than the proposed adder and leading to poor learning performance (see Fig. 8(f)). Regrettably, the high power consumption from the carry selection in VLCSA-1 is an obstacle to attain energy efficiency. The CLA, ACA and the proposed design can operate at the scaled supply voltage of 0.95 V. The proposed adder consumes about 26% and 8% less energy than CLA and ACA, respectively. Our design achieves the energy savings of up to 48.5% and 26.1% over RCA and CLA, respectively. It can be seen that our adder has the most competitive energy and error tradeoff among all these designs. Since a few hundreds of silicon neurons are integrated in the form of an array [8, 11], the total energy saving resulted from our design is remarkable for the neuromorphic chip.

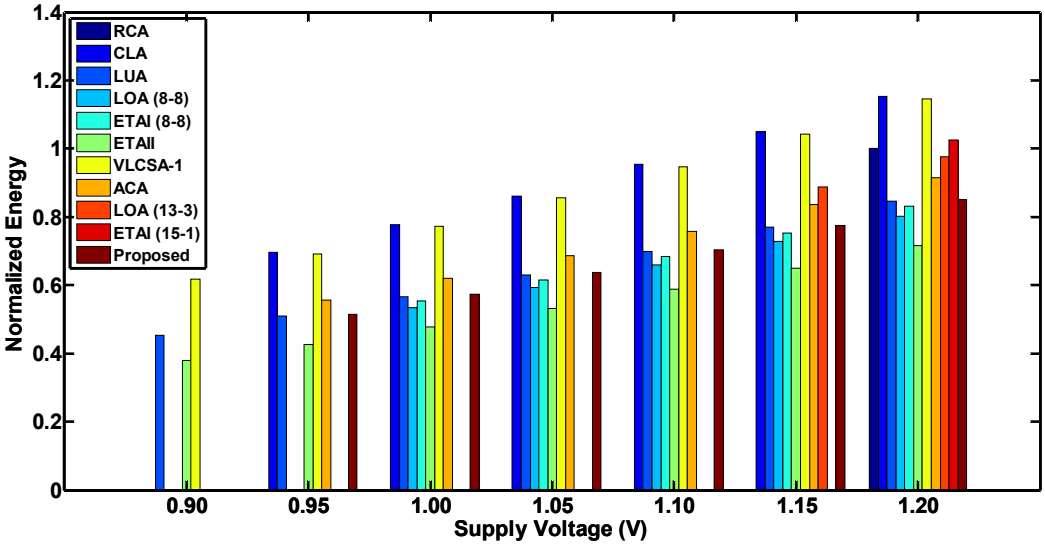

Figure 10: Normalized energies of the LIF neuron with various adders with supply voltage scaling.

5. CONCLUSION

A novel approximate adder design to considerably reduce energy consumption with a very moderate error rate has been presented for energy efficient neuromorphic VLSI systems. The proposed carry prediction with carry skip scheme significantly enhances the overall error rate and the critical path delay. Additionally, the error magnitude reduction technique reduces the amount of error further with low cost. Implemented in a commercial 90 nm CMOS process, the proposed adder is 2.4× faster and 43% energy efficient over traditional adders. We have demonstrated the performance of the adders under an unsupervised learning based VLSI neuromorphic character recognition chip by developing a hardware-aware simulation approach. The results have proven that the approximation errors of the proposed adder affect the training performance negligibly. Moreover, the adder allows for energy savings of up to 48.5% over traditional adders for digital LIF neurons with scaled supply voltage levels. Accordingly, the proposed design approach is applicable to energy efficient neuromorphic VLSI system designs.

6. ACKNOWLEDGMENT

This work was supported in part by an SRC gift under task 2364.001.

7. REFERENCES

[1] R. A. Brualdi. *Introductory combinatorics*. Prentice-Hall, 2010.

[2] V. Chippa, D. Mohapatra, A. Raghunathan, K. Roy, and S. Chakradhar. Scalable effort hardware design: Exploiting algorithmic resilience for energy efficiency. In *IEEE/ACM Design Automation Conf. (DAC)*, pages 555–560, Jun. 2010.

[3] K. Du, P. Varman, and K. Mohanram. High performance reliable variable latency carry select addition. In *Design, Automation Test in Europe (DATE)*, pages 1257–1262, Mar. 2012.

[4] S. Esser, A. Ndirango, and D. Modha. Binding sparse spatiotemporal patterns in spiking computation. In *Int. Joint Conf. on Neural Networks (IJCNN)*, pages 1–9, Jul. 2010.

[5] V. Gupta, D. Mohapatra, S. Park, A. Raghunathan, and K. Roy. Impact: Imprecise adders for low-power approximate computing. In *Int. Symp. on Low Power Electronics and Design (ISLPED)*, pages 409–414, Aug. 2011.

[6] R. Hegde and N. Shanbhag. Soft digital dignal processing. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 9(6):813–823, Dec. 2001.

[7] J. Huang, J. Lach, and G. Robins. A methodology for energy-quality tradeoff using imprecise hardware. In *IEEE/ACM Design Automation Conf. (DAC)*, pages 504–509, Jun. 2012.

[8] J.-S. Seo *et al.*. A 45nm cmos neuromorphic chip with a scalable architecture for learning in networks of spiking neurons. In *IEEE Custom Integrated Circuits Conf. (CICC)*, pages 1–4, Sep. 2011.

[9] A. Kahng and S. Kang. Accuracy-configurable adder for approximate arithmetic designs. In *IEEE/ACM Design Automation Conf. (DAC)*, pages 820–825, Jun. 2012.

[10] S. H. Kim, S. Mukhopadhyay, and M. Wolf. Modeling and analysis of image dependence and its implications for energy savings in error tolerant image processing. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 30(8):1163–1172, Aug. 2011.

[11] Y. Kim, Y. Zhang, and P. Li. A digital neuromorphic vlsi architecture with memristor crossbar synaptic array for machine learning. In *IEEE Int. System-on-Chip Conf. (SOCC)*, pages 328–333, Sep. 2012.

[12] S.-L. Lu. Speeding up processing with approximation circuits. *Computer*, 37(3):67–73, Mar. 2004.

[13] H. Mahdiani, A. Ahmadi, S. Fakhraie, and C. Lucas. Bio-inspired imprecise computational blocks for efficient vlsi implementation of soft-computing applications. *IEEE Trans. Circuits Syst. I*, 57(4):850–862, Apr. 2010.

[14] J. Miao, K. He, A. Gerstlauer, and M. Orshansky. Modeling and synthesis of quality-energy optimal approximate adders. In *IEEE/ACM Int. Conf. on Comp.-Aided Design (ICCAD)*, pages 728–735, Nov. 2012.

[15] D. Mohapatra, V. Chippa, A. Raghunathan, and K. Roy. Design of voltage-scalable meta-functions for approximate computing. In *Design, Automation Test in Europe (DATE)*, pages 1–6, Mar. 2011.

[16] Z. Pan and M. Breuer. Basing acceptable error-tolerant performance on significance-based error-rate (sber). In *IEEE VLSI Test Symp. (VTS)*, Apr. 2008.

[17] B. Shim, S. Sridhara, and N. Shanbhag. Reliable low-power digital signal processing via reduced precision redundancy. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 12(5):497–510, May 2004.

[18] J. Vanne, E. Aho, T. Hamalainen, and K. Kuusilinna. A high-performance sum of absolute difference implementation for motion estimation. *IEEE Trans. Circuits Syst. for Video Tech.*, 16(7):876–883, Jul. 2006.

[19] A. Verma, P. Brisk, and P. Ienne. Variable latency speculative addition: A new paradigm for arithmetic circuit design. In *Design, Automation Test in Europe (DATE)*, pages 1250–1255, Mar. 2008.

[20] N. Zhu, W. L. Goh, and K. S. Yeo. An enhanced low-power high-speed adder for error-tolerant application. In *Int. Symp. on Integrated Circuits (ISIC)*, pages 69–72, Dec. 2009.

[21] N. Zhu, W. L. Goh, W. Zhang, K. S. Yeo, and Z. H. Kong. Design of low-power high-speed truncation-error-tolerant adder and its application in digital signal processing. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 18(8):1225–1229, Aug. 2010.