An Approach to Energy-Error Tradeoffs in Approximate Ripple Carry Adders

Zvi M. Kedem*, Vincent J. Mooney†‡¶, Kirthi Krishna Muntimadugu§¶ and Krishna V. Palem§¶
*Courant Institute of Mathematical Sciences
New York University, New York, USA, Email: kedem@nyu.edu
†School of Electrical and Computer Engineering
Georgia Institute of Technology, Atlanta, USA, Email: mooney@gatech.edu
‡School of Electrical & Electronic Engineering and School of Computer Engineering
Nanyang Technological University, Singapore, Email: vjmooney@ntu.edu.sg
§Department of Electrical and Computer Engineering
Rice University, Houston, Texas, USA, Email: (kirthi.krishna, palem)@rice.edu
¶NTU-Rice Institute of Sustainable and Applied InfoDynamics (ISAID)
Nanyang Technological University, Singapore

***Abstract*—Given a 16-bit or 32-bit overclocked ripple-carry adder, we minimize error by allocating multiple supply voltages to the gates. We solve the error minimization problem for a fixed energy budget using a binned geometric program solution (BGPS). A solution found via BGPS outperforms the two best prior approaches, uniform voltage scaling and biased voltage scaling, reducing error by as much as a factor of 2.58X and by a median of 1.58X in 90nm transistor technology.**

***Index Terms*—Approximate Adders, Voltage Scaling, Low Energy Circuits, Geometric Programming**

I. Introduction

In this paper we introduce a novel approach to minimizing error in a Ripple Carry Adder (RCA) due to overclocking. We call our approach the Binned Geometric Program Solution (BGPS). We assume that an energy budget is fixed and that some small errors can be tolerated; thus, our goal is to minimize the errors. We have freedom to assign up to four distinct supply voltages to the gates of the RCA. We do not address physical design and floorplanning in this paper, but instead leave these issues for future work.

II. Technology Background and Prior Work

This paper presents a contribution to minimization of energy and error in so-called "approximate arithmetic" [1]. Very briefly, Chakrapani et al. [1] proposed reducing the supply voltage of VLSI arithmetic circuits beyond the point where the critical path is guaranteed to not be violated. This could result in an erroneous output of the circuit but, as shown by Chakrapani et al., can also be used in many applications which do not require strict 100% accuracy of computed values such as in audio and video signal processing. Such circuits which are "overclocked," being operated at a frequency higher than required to guarantee 100% accuracy, we call "approximate circuits."

There are other techniques to reduce switching power of a circuit. Techniques such as the ones by Blaauw et al. [2] and Broderson et al. [3] are *adaptive* which means that the throughput of the circuit is based on the workload. Non-adaptive techniques typically operate the circuit at multiple voltages which might rely on circuit implementation techniques like transistor sizing for energy efficiency [4]. Manzak and Chaktrabarti [5] as well as Yeh et al. [6], [7] present techniques that are also non-adaptive but which operate the critical paths of the circuit at higher voltages than the non-critical paths and also use transistor sizing.

Among the ones that use overclocking, one of them is broadly know as the "Razor" approach [8], [9] – championed by researchers at the University of Michigan – which allows errors at the circuit level, but only temporarily (for a few clock cycles). In this case, errors are corrected by inserting delay in order to continue from a previous known "correct" logic state. It is assumed that prior approaches to handle flip-flops and meta-stability issues [8], [9] can be used.

George et al. [10] presented a biased voltage scaling (BIVOS) for probabilistic ripple carry adders under the assumption that error sources (e.g., thermal noise) are uniformly random in time and space. The claim was that a geometric scaling of the probability of error at each bit position across an adder results in lower average error for the same energy consumption when compared to a uniform scaling of probability of error. The modeling and analysis in [10] built on the result shown by Cheemalavagu et al. [11] which described a probabilistic CMOS switch and the direct relationship between the supply voltage (energy consumption) and the probability of error. In such a probabilistic gate, an error at the output of the gate occurs based only on its supply voltage. Therefore, to establish a supply voltage allocation scheme, a geometric scaling of probability was introduced by utilizing the exact relationship between the probability of error and the supply voltage of a building block (in their case it was a full adder). This scheme cannot be directly applied to approximate circuits because there is no straightforward relationship between probability of error at a given bit position and its supply voltage. Therefore, it is not trivial to determine a supply voltage scheme from a required scaling of probability of errors across bit positions.

The primary distinctions between the previous approaches and our target problem in this paper are that we (i) present a rigorous mathematical model for the output error and energy consumption of an approximate RCA and also (ii) present a circuit level optimization methodology for multiple supply voltages whereas the prior approaches have either been ad-hoc circuit level approaches or algorithmic level optimizations (altering the algorithm being executed).

In this paper, we propose a methodology to find an assignment of multiple voltages to an approximate ripple carry adder circuit to minimize error for a given energy consumption. The first design constraint that arises out of this design methodology is the number of distinct voltages that could be used in practice in a fabricated chip. Circuit designers are using an increasing number of different voltages in their architectures. Typical high end chips seem to have four different voltages [12], [13]. To benefit from having multiple voltages on the die, the circuit designer has to overcome the challenge

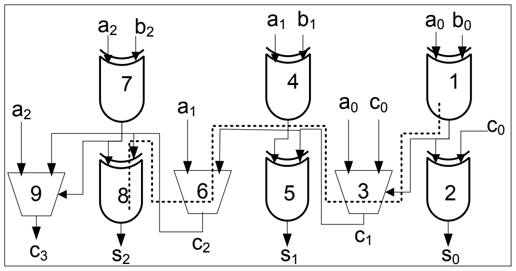

Fig. 1. Gate level diagram of a 3-bit ripple carry adder

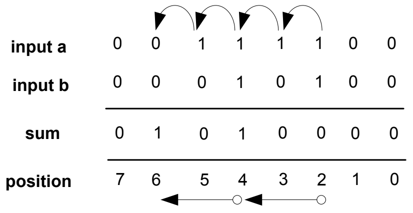

Fig. 2. An example of two contiguous carry chains in a binary addition using an RCA

of creating power distribution networks that feed from the voltage regulator modules that supply all the devices using the fewest number of interconnect layers. But the important point to note here is that the number of different voltages is the bottleneck here and not the actual magnitude of each voltage. The circuit designer has the freedom, albeit at design level, to choose the number of voltage levels and the exact values of the different voltages. With the freedom of using multiple voltage levels the use of voltage shifters becomes a necessity at least in some cases such as when a circuit with lower supply voltage is driving a circuit with higher supply voltage (and the difference in the supply voltages is not negligible) and when the output of a circuit is being stored in a register. It has been shown by Chang et al. [14] that the area/delay overhead of level shifters for using multiple supply voltages can be relatively small. Hence in this paper for the sake of simplicity of our mathematical model and experimental methodology we do not consider the overhead of voltage level shifters.

III. Approximate RCA Error and Energy Models

We will consider an n-bit ripple-carry adder (RCA). Binary numbers $\mathbf{a} = a_{n-1} \dots a_0$ and $\mathbf{b} = b_{n-1} \dots b_0$ to be added are unsigned in the range $0, \dots, 2^n - 1$ and have the standard binary representation. The addition of $\mathbf{a}$ and $\mathbf{b}$ results in an $(n+1)$-bit number $\mathbf{s} = s_n \dots s_0$. Thus, sum $\mathbf{s}$ is computed in an RCA as $s_k = a_k \oplus b_k \oplus c_k$ for $0 \le k \le n-1$. The sequence of carries $\mathbf{c}$ is computed as follows. c_0 is input (and is zero for addition which is the only case considered in this paper) while $c_{k+1} = (a_k \oplus b_k) \cdot c_k + \overline{(a_k \oplus b_k)} \cdot a_k$ for $0 \le k \le n-1$. Note that we have yet to define s_n; in fact, the last sum bit s_n is equal to the last carry bit c_n.

We use two XOR gates and a MUX for our full adder (FA) design in this paper, as can be seen in Fig. 1; while not necessarily the fastest possible or the least area, for simplicity we nonetheless use two XOR gates and a MUX for FA implementation in this paper. Each gate in Fig. 1 has an index $\ell \in \{1, 2, \dots, 9\}$; for each gate ℓ with supply voltage v_ℓ, we use $\epsilon_\ell(v_\ell)$ to denote the worst case delay of the gate at the specified voltage.

A. Carry chains in ripple carry adders

Given an n-bit RCA and two specific n-bit binary numbers $\mathbf{a} = a_{n-1}a_{n-2} \dots a_0$ and $\mathbf{b} = b_{n-1}b_{n-2} \dots b_0$ as inputs to the RCA, define $\mathbf{a}^{\mathrm{x}} = a^{\mathrm{x}}_n a^{\mathrm{x}}_{n-1} a^{\mathrm{x}}_{n-2} \dots a^{\mathrm{x}}_0$ and $\mathbf{b}^{\mathrm{x}} = b^{\mathrm{x}}_n b^{\mathrm{x}}_{n-1} b^{\mathrm{x}}_{n-2} \dots b^{\mathrm{x}}_0$ where $a^{\mathrm{x}}_i = a_i$, $b^{\mathrm{x}}_i = b_i$ for $0 \le i \le n-1$ and $a^{\mathrm{x}}_n = b^{\mathrm{x}}_n = 0$. A carry chain is said to be present in the n-bit RCA with inputs $\mathbf{a}$ and $\mathbf{b}$ from position i to position j if and only if

- $a^{\mathrm{x}}_i = b^{\mathrm{x}}_i = 1$. This case is referred to as the generation of a carry.
- $a^{\mathrm{x}}_w \neq b^{\mathrm{x}}_w$. This case is referred to as the propagation of a carry.
- $a^{\mathrm{x}}_j = b^{\mathrm{x}}_j$. If $a^{\mathrm{x}}_j = 0$, the carry is said to be killed and if $a^{\mathrm{x}}_j = 1$ another carry is said to be generated. In both the cases the carry chain that was generated at position i ends at position j.

where $0 \le i < w < j \le n$ (or, if $j = i+1$, $0 \le i < j \le n$ and $i \neq n-1$).

If there is a carry chain from i to j, we will set a boolean variable C_{ij} to 1. Also, for a carry chain from position i to position j, we have $s_i = 0 \oplus c_i$, $c_k = 1$ and $s_k = 0$, for $k \in \{i+1, \dots, j-1\}$; and $c_j = 1$, $s_j = 1$, and $c_{j+1} = a_j$ $(= b_j)$. Thus, if we know that there is a carry chain from position i to position j, we know that $s_{i+1} = s_{i+2} = \dots = s_{j-1} = 0$ and $s_j = 1$ while s_i depends on c_i.

While there can be more than one carry chain in a single addition, it turns out that multiple carry chains cannot overlap [15]. Fig. 2 shows an example where a carry chain starts at bit position 2 and is killed at position 4, thus $C_{24} = 1$; another carry chain starts at position 4 and is killed at position 6, $C_{46} = 1$.

B. Modeling the error at the output of an approximate RCA

We assume that the clock cycle time (D) of an adder is never lower than the worst-case propagation delay of a single full adder. We further assume that at the start of each addition, all carry bits have been reset to zero. Considering an approximate RCA, the result of these two assumptions is that there is a possibility of error at the output of an RCA only if there is propagation of carry: if there is no propagation of carry, the clock cycle time is sufficient for the full adders to compute the sum outputs of the RCA. Stated in a different way, there may be an error at the output only if there are carry chains in the approximate RCA. Hence in developing an error model for an approximate RCA, we will consider the behavior of error at the output of an RCA in the presence of carry chains.

We call the longest delay path with respect to a particular sum bit s_k as the "sum path of s_k." In an RCA, the sum path of s_k is the series of gates in the RCA which constitutes the longest delay path, assuming worst-case gate delays. This path is essential in computing whether there is enough time to always compute correctly a particular sum bit s_k. Of course, the true critical path for sum s_k depends on inputs $\mathbf{a}$ and $\mathbf{b}$; however, the result is that any true critical path – whose delay exceeds that of a single FA – comes from a prior bit position: for s_k, then, the true critical path given inputs $\mathbf{a}$ and $\mathbf{b}$ will come from bit position i where $i < k$. To capture all such possibilities, we define d_{ik} to be the time between the correct computation of sum bit s_k and the time when the inputs are provided to the RCA circuit thus triggering a true critical path for s_k starting from bit i. Inputs $\mathbf{a}$ and $\mathbf{b}$ are provided to the RCA at some time t_{in}. Let t_k be the time when the correct value of s_k is generated (assuming worst-case delays of all gates). Then $d_{ik} = t_k - t_{\mathrm{in}}$. $\mathbf{d}$ denotes the $(n+1) \times (n+1)$ matrix of all d_{ik}, $0 \le i < k \le n$. Properly speaking, $\mathbf{d}$ for a particular RCA in a particular technology is a function of the critical paths of the sum bits of the RCA, v_ℓ for each gate ℓ in any critical path of any sum bit, and $\epsilon_\ell(v_\ell)$; however, for brevity, in this paper we will simply refer to $\mathbf{d}$ without specifying all the input values on which $\mathbf{d}$ depends. For additional details please refer to [15].

TABLE I
MAXIMUM AND MINIMUM PROPAGATION DELAYS OF THE XOR GATE AND THE MUX IN 90NM TECHNOLOGY

$Gate$	$\epsilon_\ell(1.2V)$ (pico-sec)	$\epsilon_\ell(0.8V)$ (pico-sec)
XOR	33.3	55.2
MUX	30.5	51.2

We will determine d_{ik} as a sum of worst case propagation delays of the gates in an RCA. In conventional digital circuit design, intermediate outputs of a circuit are ignored as long as the final outputs are correct, but in overclocked circuits intermediate results could be construed as the final outputs and hence cannot be ignored during modeling. This warrants a consideration of worst-case vs. average-case vs. best-case propagation delays. In this paper, we do not consider spatial or temporal parameter variations. We only model the effect of different supply voltages. We start with an analysis of approximate RCAs assuming worst-case delays. While we do not have a formal proof to show that consideration of average and best case delays will result in lower error rates, we have found empirically for an RCA fed inputs having a uniform statistical distribution, increasing the delay increases the errors in the adder at least 98% of the time [15].

Example 1. Consider the 3-bit ripple carry adder shown in Fig. 1. Assume that the inputs are such that there is a carry chain from position 0 to position 2. One instance of such inputs are $\mathbf{a} = 011$, $\mathbf{b} = 001$ and $c_0 = 0$. For this instance, a carry bit of 1 is generated at position 0, is propagated through position 1 and is killed at position 2. For this scenario, $d_{02} = t_2 - t_{\text{in}}$, where t_2 is the time when the correct s_2 is generated. Assuming worst-case gate delays, d_{02} would be equal to the sum of worst-case propagation delays of the critical path shown in Fig. 1 as a dotted line from bit position 0 to the sum output in bit position 2. That means in the worst case, $d_{02} = \epsilon_1(v_1) + \epsilon_3(v_3) + \epsilon_6(v_6) + \epsilon_8(v_8)$. For our target technology of 90nm (Table I shows the corresponding delay values), considering maximum supply voltages for all the gates, i.e., $v_1 = v_3 = v_6 = v_8 = 1.2V$, we find that $d_{02} = 33.3$ ps $+30.5$ ps $+30.5$ ps $+33.3$ ps $= 127.6$ ps.

With calculations similar to d_{02}, we find the following:

$$\mathbf{d} = \begin{pmatrix} - & 97.1\text{ ps} & 127.6\text{ ps} & 124.8\text{ ps} \\ - & - & 97.1\text{ ps} & 94.3\text{ ps} \\ - & - & - & - \\ - & - & - & - \end{pmatrix}$$

□

Note that multiple carry chains can be handled by $\mathbf{d}$. For example, for Fig. 2, entries d_{24} and d_{46} would be used for sum bits s_4 and s_6, respectively.

C. An error model for an approximate RCA based on carry chain analysis

In this subsection, we will develop a function for the error at the output of an overclocked RCA. Note that we assume all bits in the RCA are initially set to zero [15]. We define the error at the output of the RCA as a function of $\mathbf{a}$, $\mathbf{b}$, $\mathbf{d}$ and D. We define $\mathbf{s}^{\text{a}}$ to be the sum actually read out; due to overclocking, $\mathbf{s}^{\text{a}}$ may be different from $\mathbf{s} = \mathbf{a}+\mathbf{b}$. We now proceed to characterize the absolute magnitude of the error $|\ \mathbf{s}^{\text{a}} - \mathbf{s}\ |$. We define an RCA's *indicator function* as follows: $I_k(\mathbf{a}, \mathbf{b}, \mathbf{d}, D) =$

$$\begin{cases} 1 & \text{if } \exists\ i < k < j \text{ such that } C_{ij} = 1 \text{ and } d_{ik} > D \\ -1 & \text{if } \exists\ i < k = j \text{ such that } C_{ij} = 1 \text{ and } d_{ik} > D \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

The indicator function is now used to develop a function to compute the error at the output of an approximate RCA. We define $Er(\mathbf{a}, \mathbf{b}, \mathbf{d}, D) = \mid \sum_{k=0}^{n} (s_k^{\text{a}} - s_k)2^k \mid$ to be the error introduced during the computation, assuming non-varying deterministic worst-case delays.

Theorem 1.

$$Er(\mathbf{a}, \mathbf{b}, \mathbf{d}, D) = \sum_{k=0}^{n} I_k(\mathbf{a}, \mathbf{b}, \mathbf{d}, D)2^k. \quad (2)$$

Proof: For a detailed proof please refer to [15]. ■

Theorem 1 (Eq. 2) gives $Er(\mathbf{a}, \mathbf{b}, \mathbf{d}, D)$ which is the error at the output of the target RCA for two specific inputs. The average of this error over all possible inputs is

$$Er_{\text{avg}}(D, \mathbf{d}) = \underset{0 \le \mathbf{a}, \mathbf{b} \le 2^n - 1}{\text{avg}} Er(\mathbf{a}, \mathbf{b}, \mathbf{d}, D). \quad (3)$$

This is a sum of 2^{2n} terms, which is not feasible to compute in a straightforward manner for large n. We will now transform the expression in Eq. 3 into a form that can be computed in O(n^2).

Recall that, given our assumptions, an error can occur only if there is a carry chain in the computation. We then note that the total error in a computation is the sum of the errors (if any) in the individual carry chains. The error introduced by a carry chain from i to j is

$$\mathbf{Er}_{\text{cc}}(D, i, j, \mathbf{d}) = \sum_{k=i+1}^{j} (s_k^{\text{a}} - s_k)2^k = \sum_{k=i+1}^{j} I_k^{\text{cc}}(D, i, j, \mathbf{d})2^k. \quad (4)$$

where

$$I_k^{\text{cc}}(D, i, j, \mathbf{d}) = \begin{cases} 1 & \text{if } i < k < j \text{ and } d_{ik} > D \\ -1 & \text{if } i < k = j \text{ and } d_{ik} > D \\ 0 & \text{otherwise.} \end{cases} \quad (5)$$

It can be shown that:

$$Er(\mathbf{a}, \mathbf{b}, \mathbf{d}, D) = \sum_{\text{all } i,j \text{ for which } C_{ij}=1} \mathbf{Er}_{\text{cc}}(D, i, j, \mathbf{d})$$

A proof of the above is available in [15]. We omit the proof for brevity.

One way to compute the average total error at the output of an adder is by summing the errors of all possible carry chains weighted by the probability of their occurrence:

$$Er_{\text{avg}}(D, \mathbf{d}) = \sum_{0 \le i < j \le n} p_{ij} \mathbf{Er}_{\text{cc}}(D, i, j, \mathbf{d}), \quad (6)$$

where p_{ij} is the probability that there exists a carry chain from i to j. Thus, the average total error is evaluated by computing and adding $n(n+1)/2$ terms only [15].

The probabilities p_{ij} can be computed given the distributions of the inputs a and b. By assuming a uniform distribution of inputs it is easy to conclude that $p_{ij} = \left(\frac{1}{2}\right)^{j-i+2}$ (when $j \neq i+1$) and $p_{ij} = \left(\frac{1}{2}\right)^3$ (when $j = i+1$); a proof is available in [15]. In real world applications, this might not be true. Therefore, if the knowledge about the probability distribution of the actual inputs is known then that could be used instead of using $P(a_i = 0) = P(b_i = 0) = P(a_i = 1) = P(b_i = 1) = \frac{1}{2}$. If the case is such that instead of the probability distribution we have a candidate input benchmark, then the probability distribution could be computed using the benchmark. In this paper, we leave the case of non-uniform input bits for future work.

IV. Energy Model for an Approximate RCA

The total energy consumption in an RCA consists of two separate components, the dynamic and static energy consumption. Our estimate of the dynamic energy consumption at the gate level of a CMOS circuit of an RCA is $E^{\text{dyn}} = \sum_{\ell=1}^{N} E_{\ell}^{\text{dyn}}(\nu_\ell) w_\ell$ where $E_{\ell}^{\text{dyn}}(\nu_\ell)$ is the dynamic energy consumption of the ℓ^{th} gate being operated at supply voltage ν_ℓ, and w_ℓ is the average switching activity of the ℓ^{th} gate in a single clock cycle (assuming a non-pipelined adder). w_ℓ for gates in the case of an RCA is approximately estimated as the ratio of the number of logic changes of gate ℓ to the total number of additions $\mathcal{A}$ simulated. Our estimate of static energy consumption is $E^{\text{stat}} = \sum_{\ell=1}^{N} P_{\ell}^{\text{stat}}(\nu_\ell) D$ where $P_{\ell}^{\text{stat}}(\nu_\ell)$ is the static power consumption of the ℓ^{th} gate being operated at supply voltage ν_ℓ and D is the clock cycle time of the circuit. Therefore, the total energy consumption is

$$E = \sum_{\ell=1}^{N} \left(E_{\ell}^{\text{dyn}}(\nu_\ell) w_\ell + P_{\ell}^{\text{stat}}(\nu_\ell) D \right) \tag{7}$$

where N is the total number of gates in the adder.

For an approximate RCA, Eq. 7 may be used if we find the switching activities for the gates when overclocking. Due to overclocking, the sum actually read might be different from the correct sum. The fact of whether at a given bit position the correct sum bit was computed in time or not is modeled using the indicator function in Eq. 1 in Subsection III-C. We use a similar model of an indicator function to check if a particular gate in the RCA had a logic change within the clock cycle time and, based on that, re-evaluate (reduce) the switching activity, denoted as w_ℓ^{a}, to reflect this [15].

Based on the revised estimates of the switching activities, the total energy consumption of an approximate RCA is as follows

$$E^{\text{a}} = \sum_{\ell=1}^{N} \left(E_{\ell}^{\text{D}}(\nu_\ell) w_\ell^{\text{a}} + P_{\ell}^{\text{S}}(\nu_\ell) D \right) \tag{8}$$

From Section III, $\epsilon_\ell(\nu_\ell)$ denotes the worst-case propagation delay of the ℓ^{th} gate when its supply voltage is ν_ℓ. We compute the average dynamic energy consumption and worst case propagation delays of all the gates in our process technology through simulations. It is known that the dynamic energy consumption of a gate is proportional to the square of the input supply voltage as well as that the propagation delay of a gate is inversely proportional to its supply voltage. To represent $E_{\ell}^{\text{dyn}}(\nu_\ell)$ in terms of $\epsilon_\ell(\nu_\ell)$, we will use the curve-fit that the average dynamic energy consumption of a gate is proportional to the inverse square of its worst case propagation delay, i.e.,

$$E_{\ell}^{\text{dyn}}(\nu_\ell) \propto \frac{1}{\epsilon_\ell^2(\nu_\ell)} = \gamma_\ell \frac{1}{\epsilon_\ell^2(\nu_\ell)} \tag{9}$$

where γ_ℓ is the proportionality constant for the ℓ^{th} gate. Thus γ_ℓ is computed separately for each type of gate [15]. For example in the design of the RCA that we consider there are two types of gates, XOR and MUX.

Substituting the relationship between average dynamic energy consumption and worst-case propagation delays from Eq. 9 into Eq. 8, we derive the following:

$$E^{\text{a}} = \sum_{\ell=1}^{N} \left(\gamma_\ell \frac{1}{\epsilon_\ell^2(\nu_\ell)} w_\ell^{\text{a}} + P_{\ell}^{\text{stat}}(\nu_\ell) D \right) \tag{10}$$

V. Minimizing Average Error of an Approximate RCA Using Geometric Programming

In this section we describe our procedure to formulate our target problem, which is minimizing average error of an approximate RCA under a given energy budget, as a geometric program [16]. Then we present our approach to perform supply voltage binning on the solution obtained from the geometric program.

We form an optimization problem consisting of an objective function and one or more constraint functions. The objective function is the average error of an approximate RCA as given in Eq. 6 in Subsection III-C. The average error as shown in Eq. 6 is a function of **a**, **b**, **d** and D. The clock cycle time D is an independent variable, and **a** and **b** are inputs, but **d** is a matrix whose elements are a function of the adder topology, resulting critical path delays and gate supply voltages. We do not alter the adder topology but instead vary the adder supply voltages which directly alters **d**. We found that a formulation of error optimization in terms of **d** – represented in terms of $\boldsymbol{\epsilon}(\boldsymbol{\nu})$ – to be much simpler than a direct formulation in terms of $\boldsymbol{\nu}$, where $\boldsymbol{\nu}$ is the vector of the voltages of all the gates in the RCA [15].

Therefore we consider the gate propagation delays as the decision variables. The RCA under consideration consists of N gates. We need to compute an optimized *supply voltage allocation scheme*, which is the exact assignment of supply voltages to the individual gates. To do that we will compute delays $\boldsymbol{\epsilon}(\boldsymbol{\nu}) = \{\epsilon_1(\nu_1), \epsilon_2(\nu_2), \ldots, \epsilon_N(\nu_N)\}$ for which the average error is minimized under the constraint that the total energy consumption is below the total energy budget. These gate delays will in turn determine the supply voltage allocation scheme.

The optimization problem is to minimize Eq. 6 which is

$$Er_{\text{avg}}(D, \mathbf{d}) = \sum_{0 \le i < j \le n-1} p_{ij} \mathbf{Er}_{\text{cc}}(D, i, j, \mathbf{d}), \tag{11}$$

subject to the following two constraints and assumptions.

1) min-delay$_\ell$ $\le \epsilon_\ell(\nu_\ell) \le$ max-delay$_\ell$, where min-delay$_\ell$ and max-delay$_\ell$ depend on the transistor technology while the delay additionally depends on the type of component and fanout.
2) The total energy consumption of all the gates is bounded from above by the given energy budget. Thus,

$$E^{\text{a}} = \sum_{\ell=1}^{N} \left(\gamma_\ell \frac{1}{\epsilon_\ell^2(\nu_\ell)} w_\ell^{\text{a}} + P_{\ell}^{\text{stat}}(\nu_\ell) D \right) \le \begin{array}{c}\text{Energy}\\ \text{Budget}\end{array}$$

In general the full class of optimization problems could be classified into two categories, linear optimization problems (LP) and nonlinear optimization problems (NLP). The objective function of our optimization problem in this paper, which is shown in Eq. 11, is not a linear function. Therefore, our solution is to formulate the problem of minimizing Eq. 11 subject to the constraints above as a geometric program [16] and then solve it. To model our target problem as a geometric program, the objective function and all the constraints should be in the form of a posynomial [16]. Our objective function is not a posynomial. So we will compute a posynomial approximation of our objective function based on the methodology given in Section 8.2 of [16]. The approach of computing a posynomial approximation of a given function and then using geometric programming to solve it is referred to as signomial programming, discussed in detail in [16]. The following is a mathematical description of the approximations and redefinitions that we use. We will redefine the indicator function I_k^{cc} (given in Equation 5) as:

$$\mathbf{I}_k(D, i, j, \mathbf{d}) = \begin{cases} 1 & \text{if } d_{ik} > D,\ C_{ij} = 1,\ i < k \le j \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

Using this definition, $\mathbf{Er}_{\text{cc}}(D, i, j, \mathbf{d})$ is transformed as follows

$$\mathbf{Er}_{\text{cc}}(D, i, j, \mathbf{d}) = \sum_{k=i+1}^{j} I_k^{\text{cc}} 2^k = \sum_{k=i+1}^{j-1} \mathbf{I}_k 2^k - \mathbf{I}_j 2^j \tag{13}$$

where I_k^{cc} is shown in Equation 5. Because the new indicator function $\mathbf{I}_k$ is a non-negative function, the negative sign appears in the definition of $\mathbf{Er}_{\text{cc}}(D, i, j, \mathbf{d})$. Thus the combination of the indicator function in Eq. 5 and the error function in Eq. 4 in Section III-C results in the same value as the redefined indicator function in Eq. 12 and transformed error function in Eq. 13.

We approximate $\mathbf{I}_k$ by $1/(1 + e^{-2\kappa(d_{ik}-D)})$ [15]. For $\kappa \gg 0$, $\text{sgn}(x) \approx \tanh(\kappa x)$, and we use $\kappa = 200$.[1] Thus, our continuous and differentiable approximation of Eq. 13 is $\mathbf{Er}_{\text{cc}}(D, i, j, \mathbf{d})$

$$\approx \sum_{k=i+1}^{j-1} \frac{1}{1 + e^{-2\kappa(d_{ik}-D)}} 2^k - \frac{1}{1 + e^{-2\kappa(d_{ij}-D)}} 2^j$$

where d_{ij} are linear functions of $\epsilon_k(\nu_k)$. We use the monomial approximation technique (Section 8.2 of [16]) for this expression. This results in $\mathbf{Er}_{\text{cc}}(D, i, j, \mathbf{d})$

$$\begin{aligned} &\approx \sum_{k=i+1}^{j-1} \frac{1}{1 + e^{-2\kappa(d_{ik}-D)}} 2^k - \frac{1}{1 + e^{-2\kappa(d_{ij}-D)}} 2^j \\ &\approx c\epsilon_1^{a^1}(\nu_1)\epsilon_2^{a^2}(\nu_2)\dots\epsilon_N^{a^N}(\nu_N) \qquad (14) \end{aligned}$$

where $c \in \mathbf{R}^+$ and $a^m \in \mathbf{R}$ for all $1 \le m \le N$.

We then construct the objective function $Er(D, \mathbf{d})$ as a posynomial [16]. For detailed examples please see [15].

As $\boldsymbol{\epsilon}(\boldsymbol{\nu})$ are the decision variables, we now express $\mathbf{d}$ in terms of $\boldsymbol{\epsilon}(\boldsymbol{\nu})$ and write the average error as $Er_{\text{avg}}(D, \boldsymbol{\epsilon})$. Then the problem is reduced to minimizing a posynomial subject to posynomial inequality constraints, giving us a geometric program in a standard form:

$$\text{Minimize } Er_{\text{avg}}(D, \boldsymbol{\epsilon}) = \sum_{j=1}^{C} c_j \epsilon_1^{a_j^1}(\nu_1)\epsilon_2^{a_j^2}(\nu_2)\dots\epsilon_N^{a_j^N}(\nu_N) \quad (15)$$

$$\text{subject to min-delay}_\ell \le \epsilon_\ell(\nu_\ell) \le \text{ max-delay}_\ell,\ k = 1, \dots, N$$

$$\text{and } \sum_{\ell=1}^{N} \left(\gamma_\ell \frac{1}{\epsilon_\ell^2(\nu_\ell)} w_\ell^{\text{a}} + P_\ell^{\text{stat}}(\nu_\ell) D \right) \le \text{ Energy Budget}$$

where C is the number of possible carry chains in an n-bit adder and N is the number gates in the adder. In the case of an $n = 16$-bit adder, $C = \frac{n(n+1)}{2} = 156$ [15].

We use a standard geometric programming toolbox [16], [17] to solve this program. The solution of the first iteration is used to compute the posynomial approximation again, until the objective value starts to converge. This gives us the final allocation of delays to the components such that the average error is minimum for the given constraints. Using the delays allocated to the components, we can obtain the voltages to be supplied to them.

The solution from the geometric program in Eq. 15, in principle, can assign any voltage to any gate under the given constraints. For a practical application of the solution we need to limit the number of supply voltages and also the number of voltage islands. We will present our approach to limit the number of supply voltages. Let the possible set of supply voltages be $\mathcal{V}$ and the maximum number of voltages be M_ν. We then pick a M_ν-combination of elements from $\mathcal{V}$. The voltages from this subset are then assigned to the gates in the RCA with gates having a higher voltage in the geometric program solution getting a higher voltage from this subset. This process is referred to as *binning*. We exhaustively search through all possible *binning* schemes. Using Eq. 11 and the relationship between propagation delay and supply voltage we can compute a closed form solution of the average error. Our algorithm for supply voltage binning is explained in detail in [15]. We refer to the binned solution of the geometric program as the Binned Geometric Program Solution (BGPS) in this paper.

As an example, we present the solution obtained from the geometric program and binning for the 3-bit RCA in Fig. 1 in Table II.

[1]This particular value of κ was chosen empirically by observing the plots of the two functions, $\text{sgn}(x)$ and $\tanh(\kappa x)$, and that the transition from -1 to 1 is fast enough.

TABLE II
PROPAGATION DELAY AND SUPPLY VOLTAGE VALUES FROM THE GEOMETRIC PROGRAM AND CORRESPONDING BINNED SUPPLY VOLTAGE VALUES FOR THE GATES IN FIG. 1

Gate Index	ϵ_ℓ (ps)	ν_ℓ (volts)	Binned ν_ℓ (volts)
1	44.6	0.84	0.8
2	46.9	0.8	0.8
3	34.0	1.16	1.2
4	44.6	0.84	0.8
5	40.8	0.92	0.9
6	33.3	1.2	1.2
7	39.3	0.96	1.0
8	38.5	0.98	1.0
9	33.3	1.2	1.2

VI. Simulation Framework for RCA Experimentation

For comparison, we will consider the biased voltage scaling (BIVOS) approach of George et al. [10] modified as follows. First, we split the number of bits equally into four sets: for 16 bits, there are four sets of four bits each, while for 32 bits, there are four sets of 8 bits each. Then, we tried the following possible combinations of four distinct voltages, from 0.8V to 1.2V, assuming a step size of 0.1V: (i) {0.8V, 0.9V, 1.0V, 1.1V}, (ii) {0.8V, 0.9V, 1.0V, 1.2V}, (iii) {0.8V, 0.9V, 1.1V, 1.2V}, (iv) {0.8V, 1.0V, 1.1V, 1.2V} and (v) {0.9V, 1.0V, 1.1V, 1.2V} where the voltages are assigned from lowest to highest from the LSB to the MSB. For example, 0.8V is the supply voltage for the least significant four bits (in the case of a 16-bit RCA) or eight bits (for the 32-bit RCA) in four out of five of the cases above. We call this approach "naive-BIVOS" or n-BIVOS for short. In addition to n-BIVOS, we will also compare our approach with uniform voltage scaling or UVOS. All the simulations were performed in HSPICE using a 90nm process technology. Each result presented has been computed as an average over 10,000 additions with inputs drawn from a uniform distribution. The average error magnitude is computed by taking an average, over all of the additions simulated, of the absolute magnitude of the difference between the correct output and the overclocked approximate output. On the other hand, the average energy consumption is measured from our HSPICE simulations by taking an average, over all of the additions, of the total energy consumption.

VII. Experimental Results

Table III summarizes the key results of this paper. A solution found via BGPS (takes about 5 min for a 16-bit RCA) outperforms the two best prior approaches, UVOS and n-BIVOS, by as much as a factor of 2.58X and by a median of 1.58X. Though the target problem in this paper is to minimize error for a given Energy Budget, the dual problem would be to minimize energy for a given Error Budget. The simulations we have performed can give us initial results for this problem as well. For example, for the same average error of 36.8 the BGPS solution has an energy consumption of 77.14fJ whereas the UVOS solution consumes 132.9 fJ (savings of 1.72X) and an n-BIVOS solution consumes 139.92 fJ (savings of 1.81X). Energy Savings in Table III is the ratio of energy consumption of a correct RCA (no errors, 1.2V supply voltage) and the energy consumption of the overclocked adder giving us the amount of energy

TABLE III
SUMMARY OF RESULTS OF 16-BIT AND 32-BIT APPROXIMATE RIPPLE CARRY ADDERS

n-bit	D (ns)	Average Error Magnitude					Energy Consumption(fJ)	Energy Savings
		UVOS	n-BIVOS	BGPS	UVOS /BGPS	n-BIVOS /BGPS		
16-bit	0.4	36.83	50.06	21.66	1.70	2.31	110.78	1.41
16-bit	0.4	36.83	45.97	17.96	2.05	2.56	128.14	1.22
16-bit	0.4	36.83	38.55	26.31	1.40	1.47	132.9	1.18
16-bit	0.4	40.92	36.83	23.25	1.76	1.58	139.92	1.12
32-bit	0.6	18171	33704	13978	1.30	2.41	132.74	2.07
32-bit	0.6	15634	24637	11412	1.37	2.16	139.41	1.97
32-bit	0.6	14892	15215	11142	1.34	1.37	152.54	1.8
32-bit	0.6	14199	10931	8931	1.59	1.22	159.3	1.73

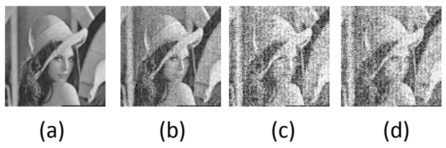

Fig. 3. Images generated by (a) Correct adders (b) BGPS adders (c) n-BIVOS and (d) UVOS

saved due to overclocking. The non-monotonic relationship between energy consumption and average error (as observed from Table III) is discussed in detail in [15].

A. FFT Results

Approximate circuits can be used in applications which do not demand 100% accuracy such as digital image processing. To demonstrate the efficiency of BGPS adders over UVOS and n-BIVOS adders, we construct three different 8-point approximate FFTs with these three types of approximate adders but with 40% less energy consumption than the correct adders. We used an image as input to the approximate FFT so that the quality tradeoff is perceptible.

We use a 16-bit approximate RCA but using the image data as input to the adder. Then we collect average error for the three types of approximate adders through simulations in HSPICE using the same framework as described in Section VI except for the input data. We use a Gaussian noise source at the output of every adder in MATLAB to simulate the effect of overclocking with the mean and variance collected from HSPICE simulations of the RCA. This will result in an approximately computed FFT of the input image. We then perform a correct inverse-FFT in MATLAB of this approximate FFT of the input image. Our goal is to see the extent to which the data has been preserved in this experiment. We would expect, if both the FFT and the inverse-FFT were correct, that the final image would be an exact copy of the original image.

The resultant images from the four cases (three approximate RCA and one correct RCA) that are discussed above are shown in Fig. 3. The image generated by the FFT using UVOS adders has a PSNR that is 15db lower than the image generated by the FFT using BGPS adders with the same energy consumption. Also, the PSNR of the image generated by the n-BIVOS adders is 8.5 dB lower than the image generated by the BGPS adders with the same energy consumption.

REFERENCES

[1] L. Chakrapani, K. Muntimadugu, A. Lingamneni, J. George, and K. Palem, "Highly energy and performance efficient embedded computing through approximately correct arithmetic: A mathematical foundation and preliminary experimental validation," in *Proc. of the IEEE/ACM Intl. Conf. on Compilers, Architecture, and Synthesis for Embedded Systems*, 2008, pp. 187–196.

[2] S. M. Martin, K. Flautner, T. Mudge, and D. Blaauw, "Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors under dynamic workloads," in *Proc. of the Intl. Conf. on Computer Aided Design*, 2002.

[3] T. Pering, T. Burd, and R. Brodersen, "The simulation and evaluation of dynamic voltage scaling algorithms," in *Proc. of the Intl. Symp. on Low Power Electronics and Design*, 1998, pp. 76–81.

[4] J. Chang and M. Pedram, "Energy minimization using multiple supply voltages," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 5, no. 4, pp. 436–443, Dec. 1997.

[5] A. Manzak and C. Chaktrabarti, "Variable voltage task scheduling algorithms for minimizing energy/power," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 11, no. 2, pp. 270–276, 2003.

[6] Y. Yeh, S. Kuo, and J. Jou, "Converter-free multiple-voltage scaling techniques for low-power CMOS digital design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 1, pp. 172–176, 2001.

[7] Y. Yeh and S. Kuo, "An optimization-based low-power voltage scaling technique using multiple supply voltages," in *Proc. of the IEEE Intl. Symp. on Circuits and Systems*, vol. 5, May 2001, pp. 535–538.

[8] S. Das, D. Roberts, S. Lee, S. Pant, D. Blaauw, T. Austin, and T. M. Krisztian Flautner, "Self-tuning dvs processor using delay-error detection and correction," in *IEEE Journal of Solid-State Circuits(JSSC)*, 2006.

[9] D. Ernst, N. S. Kim, S. Das, S. Lee, D. Blaauw, T. Austin, T. Mudge, and K. Flautner, "Razor: Circuit-level correction of timing errors for low-power operation," in *IEEE MICRO special issue on Top Picks From Microarchitecture Conferences of 2004*, 2005.

[10] J. George, B. Marr, B. Akgul, and K. Palem, "Probabilistic arithmetic and energy efficient embedded signal processing," in *Proc. of the IEEE/ACM Intl. Conf. on Compilers, Architecture, and Synthesis for Embedded Systems*, 2006, pp. 158–168.

[11] 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 the IFIP Intl. Conf. on Very Large Scale Integration (VLSI-SoC)*, 2005, pp. 452–457.

[12] T. Kamei, T. Yamada, T. Koike, M. Ito, T. Irita, K. Nitta, T. Hattori, and S. Yoshioka, "A 65nm dual-mode baseband and multimedia application processor soc with advanced power and memory management," in *Proc. of the Asia and South Pacific Design Automation Conf.*, 2009, pp. 535–539.

[13] AMD, "AMD Turion X2 Ultra and Turion X2 Key Architecture Features," http://www.amd.com/uk/products/notebook/processors/turion-_x2/Pages/turion-_x2-_mobile-_features.aspx, 2010.

[14] J.-M. Chang and M. Pedram, "Energy minimization using multiple supply voltages," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 5, no. 4, pp. 436 –443, Dec. 1997.

[15] Z. Kedem, V. Mooney, K. K. Muntimadugu, and K. Palem, "Optimizing energy to minimize errors in approximate ripple carry adders," *Rice University Technical Report*, vol. TR11-02, 2011.

[16] S. Boyd, S. Kim, L. Vandenberghe, and A. Hassibi, "A tutorial on geometric programming," *Optimization and Engineering*, vol. 8, no. 1, pp. 67–127, 2007.

[17] A. Mutapcic, K. Koh, S. Kim, and S. Boyd, "GGPLAB: A MATLAB toolbox for geometric programming," 2006.