Optimizing Energy to Minimize Errors in Approximate Ripple Carry Adders

Zvi M. Kedem[1], Vincent J. Mooney[2,3,5], Kirthi Krishna Muntimadugu[4,5], and Krishna V. Palem[4,5]

[1]Courant Institute of Mathematical Sciences, New York University, New York, USA, Email:kedem@nyu.edu
[2]School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, USA, Email:mooney@gatech.edu
[3]School of Electrical & Electronic Engineering and School of Computer Engineering, Nanyang Technological University, Singapore, Email:vjmooney@ntu.edu.sg
[4]Department of Electrical and Computer Engineering, Rice University, Houston, Texas, USA, Email:(kirthi.krishna, palem)@rice.edu
[5]NTU-Rice Institute of Sustainable and Applied InfoDynamics (ISAID), Nanyang Technological University, Singapore

Abstract

We present a theoretical foundation and a methodology for automatically assigning supply voltages to approximate ripple carry adders in which accuracy is traded for energy consumption. The error minimization problem for a fixed energy budget is formulated as a binned geometric program. We first use geometric programming to minimize the average error of the adder and compute the supply voltages at the gate level, after which we bin the voltages to a finite set (of four or five voltages) using a heuristic. Using HSPICE in 90nm technology, we show simulation results by applying our methodology to a ripple carry adder and obtain savings of up to 2.58X (and by a median of 1.58X) in average error, when compared to uniform voltage scaling, for the same energy consumption. Compared to a naive biased voltage scaling (n-BIVOS), which is the best prior art in literature, a Binned Geometric Program Solution (BGPS) as proposed in this paper saves 32.3% energy with the same PSNR in an 8-point FFT example or, alternatively, increases the PSNR by 8.5db for the same energy consumption for the FFT.

1 Introduction

Low power circuit design is vital to sustaining higher functionality and prolonging battery life of today's ubiquitous portable electronic devices. The simplest way to save energy is to decrease the supply voltage of a circuit as the feature size becomes smaller. But the rapid shrinking of transistors has made it very difficult to maintain 100% reliability, which typically requires the supply voltage not to be decreased, thus imposing contradictory requirements. Also, with scaling of transistors, process variations do not allow a fine-grained control over the parameters of the circuit components, such as delay, where it has been shown that the variation is up to 30% in 70nm technology [1]. Thus, a VLSI designer may have to deal with circuits where the delays of the components are not predictable and the accuracy of the output is less than 100%. These circuits can potentially be used in many applications such as audio and video signal processing where 100% accuracy is not essential and thus the reliability of these devices can be relaxed.

To model circuits with reduced voltages resulting in occasional errors, a novel methodology of *probabilistic* and *approximate* arithmetic circuits [2, 3, 4] has been proposed. One case of probabilistic circuits involves modeling the effect of the inherent thermal noise in the devices especially as the supply voltages are lowered. In this paper we deal with approximate circuits, in particular arithmetic adders, where the accuracy of the output is traded for energy consumption through deliberate overclocking and multiple voltage levels; in the circuits we consider, thermal noise does not play a role, and so approximate arithmetic explored in this paper is applicable to current process technologies. It has been shown [3, 4] that in approximate

circuits, where outputs are sometimes not accurate due to delay errors, accuracy can be increased by *biased voltage scaling* (BIVOS) in which the computation of more significant bits is given higher voltage than that of lower significant bits, compared to *uniform voltage scaling* (UVOS) in which a uniform lower voltage is supplied across the adder [3]. But the voltage assignment to the components in the circuit has been ad-hoc. There has not been a well-grounded approach to guide the designer in determining the voltage allocation in the adder to maximize accuracy while staying within the available energy budget.

In this paper, we present a methodology which efficiently finds a supply voltage investment in the case of approximate adders for minimizing the average error at the output of the adder while keeping the energy consumption under a given budget. The primary contributions are as follows:

- We develop a formal model to characterize the average error in an approximate adder as a function of its design parameters.
- We also develop an energy model to estimate the energy consumption of an approximate adder. We present an algorithm to compute the reduced switching activities for a ripple carry adder due to overclocking.
- We then pose our target problem as an optimization problem where the objective function is the average error of the approximate adder. The constraint is that the energy consumption of the adder should be under the given Energy Budget.
- We compute an approximation of our non-linear optimization problem as a geometric program. We then solve it using a standard toolbox which results in a continuous allocation of supply voltages to all the gates in the adder.
- For pragmatic concerns, we perform "supply voltage binning" which limits the number of supply voltages in the adder by use of a heuristic algorithm.
- We demonstrate the improvement that results from this methodology by applying it to a 16-bit ripple carry adder (RCA). We were able to reduce the error in an BGPS-based RCA by up to 2.3X when compared to an RCA with a naive biased voltage scaling (n-BIVOS) scheme with the same energy consumption.

We present here a roadmap to this paper. In Section 2, we present a brief background on current day CMOS VLSI technology and also discuss some physical fabrication constraints on the implementation of multiple voltages. We also describe the technique of approximate arithmetic and the motivation behind it in Section 2. To avoid ambiguity about the exact problem that we target in this paper, we discuss our goal briefly differentiating it against conventional circuit design optimization problems in Section 3. We present a comprehensive list of the variables and their definitions that will be used throughout the paper in Section 4. In Section 5 we discuss a few instances of prior work which are related to the problem addressed in this paper. We describe assumptions that we use in the paper on variables related to delay in Section 6 and with respect to energy modeling in Section 7.

The crux of the paper is discussed in Section 8 where we present our model for the output error in an approximate RCA. To elaborate, Section 8.1 presents the gate level description of the ripple carry adder. In Section 8.2, we present the model of an approximate RCA and the mathematical characterization of the output error in an approximate RCA. We describe our approach to reducing the computational complexity, which is one of the primary reasons for the practical validity of our approach, of the mathematical characterization of the average output error of an approximate ripple carry adder over all possible inputs, in Section 9. We present our approach to model the energy consumption in an approximate RCA in Section 10. We present our target optimization problem and the technique of geometric programming that we use for solving our target optimization problem in Section 11. We propose a supply voltage binning scheme in Section 11.3 where we map the solution obtained from geometric programming to a given set of supply voltages.

We then describe our simulation framework with which we perform all of our experiments in Section 12. We present the experimental results in Section 13 that compare our binned geometric programming solution (BGPS) to a uniform voltage scaling scheme (UVOS) and a naive-biased voltage scaling scheme (n-BIVOS) on our target adder designs. In Section 13, we also discuss the non-monotonic behavior of output error rates of adders and present the impact of a globally optimized RCA design in the context of an FFT.

The impact of our optimization methodology on circuit design is discussed in Section 14, and in Section 15 we present our conclusions and future directions.

2 Technology Background

In this section we discuss some background about the technology that we use in this paper. Specifically, we explain

the concepts of approximate arithmetic, overclocking and the use of multiple supply voltages in a circuit.

The problem of reducing energy consumption has in the current day circuit design industry taken the role of a "first citizen." There are multiple approaches that have been proposed to combat this problem. In this paper, we focus on one such approach by Chakrapani et al. [3] where a technique called "approximate arithmetic" was proposed. In conventional design methodology, a circuit is operated at a speed that is directly related to the critical path delay of the circuit. The critical path delay, in turn, is directly related to the topology, process technology and supply voltage(s) of the circuit. The supply voltage dependence is due to the switching delay of a transistor being inversely proportional to its supply voltage. Also, it is known that the dynamic energy consumption of a transistor is in general proportional to the square of its supply voltage. Hence, a very direct way of reducing the switching power of a circuit is to reduce its supply voltage. But this would lead to a reduction in the operating frequency of the circuit and thus impact performance which is also a serious constraint. Keeping these constraints in mind, Chakrapani et al. [3] proposed a methodology in which the operating frequency of a circuit is maintained at the same rate but the supply voltage is decreased past the limit at which the critical path of the circuit 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 [3]. Such circuits which are "overclocked," being operated at a frequency higher than required to guarantee 100% accuracy, we call "approximate circuits."

In this paper, as described in Section 1, 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 multiple voltages that could be useful in practice in the fabricated chip. Circuit designers are using an increasing number of different voltages in their architectures. Typical high end chips seem to have four or five different voltages [5, 6]. To benefit from having multiple voltages on the die, the circuit designer has to overcome the challenge of creating power distribution networks that feed from the voltage regulator modules that supply all the devices on 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. [7] 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 limit ourselves to four or five voltages and do not consider the overhead of voltage level shifters. Furthermore, we assume that four or five voltages are already part of the design. So we do not consider in this paper the energy cost of generating and distributing multiple voltages on-chip.

We have given a brief background about approximate circuits and overclocking which are used throughout in this work. We also use multiple voltages in our circuits and hence we mention some constraints and issues in using multiple supply voltages.

3 Discussion About Circuit Optimization And Our Target Problem

In this section we first describe our target problem. We then present a brief background about circuit design optimization in the context of our target problem.

3.1 Our target problem: approximate adder supply allocation problem

In this subsection we present the problem that we target in this paper.

An approximate circuit as described in Section 2 is a circuit whose actual output might not be equal to the correct logical output because the circuit is overclocked. In this paper we specifically target approximate adders. The two most basic parameters that characterize an approximate adder are the average output error over all possible input cases and the average energy consumption. The two main directions in which an approximate adder can be optimized are as follows:

1. Minimize the energy consumption of an approximate adder for a given average output error

2. Minimize the average output error of an approximate adder for a given energy consumption

While the former direction may likely be most appropriate for some applications, in this paper we address the latter problem as a first step and are attempting to target the former problem for future work.

In short, our target problem in this paper is to minimize the average output error of an approximate adder for a given energy consumption.

3.2 Background about circuit optimization problems

In this subsection we briefly discuss circuit optimization problems in general and the distinction of our target problem with respect to general circuit optimization problems.

Most circuit design optimization problems are computationally expensive to solve. For example, a typical place and route problem may be modeled as a linear programming or an integer linear programming problem which in the worst case are very expensive to solve. Hence such optimization problems of practical sizes are usually solved using heuristics and not by exact methods.

On the other hand, the problem that we target in this paper is the "approximate adder supply allocation problem" described in Subsection 3.1. Adders are typically used in a few options as far as number of bits are concerned: 16, 32 and 64 are the most common. In contrast to regular place and route problems which typically target 100,000 to a million transistors, we currently do not need to look at adders larger than 64 bits wide. Furthermore, due to limitations in the design of power converter circuitry [8] and layout overheads for power planes, typically only a small number of voltages, e.g., four or five, are used in practice. As a result, in the design of an energy-optimized adder, at most 64 bits and at most five voltages is in practice a reasonable limitation leading to a limited range in the variables of the problem. As such, the problem can be formulated and solved using exact methods which can be exponential in terms of run time in the worst case. In this paper we use geometric programming which typically has polynomial run time but can be exponential in the worst case.

In summary, general circuit optimization problems are computationally very expensive to solve exactly and hence typically heuristics are used. The distinction of our target problem with respect to general circuit optimization problems is that we target specifically approximate arithmetic adders of a set of exact sizes (64 bits or less) and thus are able to utilize an exact solution method.

4 Terminology

In this section, we define the terms that we use in this paper to solve the approximate ripple carry adder (RCA) supply allocation problem. The reader may choose to skip this section and refer back only as needed.

1. $\mathbf{a}$: A multi-bit binary number with n bits
2. a_k : Represents the k^{th} bit in the multi-bit binary number $\mathbf{a}$ where $0 \leq k < n$.
3. $\mathbf{s}$: Represents the correct $n+1$-bit sum output of an n-bit RCA.
4. s_k : Denotes the k^{th} bit of $\mathbf{s}$, where $0 \leq k < n+1$.
5. Approximate adder: An approximate adder is an adder circuit whose sum output might not be the correct output because the adder is overclocked. In this paper we consider approximate RCAs.
6. $\mathbf{s}^{\text{a}}$: Represents the approximate $n+1$-bit sum output of an n-bit RCA. The sum output of an approximate n-bit RCA, $\mathbf{s}^{\text{a}}$, may have errors, i.e., it may be the case that $\mathbf{s}^{\text{a}} \neq \mathbf{s}$.
7. s_k^{a} : Denotes the k^{th} bit of $\mathbf{s}^{\text{a}}$, where $0 \leq k < n+1$.
8. c_k: Represents the k^{th}, where $0 \leq k < n+1$, carry bit in an RCA.
9. $\mathbf{c}$: The sequence of carry bits in the RCA. Therefore $\mathbf{c} = c_n c_{n-1} \ldots c_0$.
10. $0 \leq i < j \leq n-1$: Represents all cases for i and j which satisfy that equation. For example, $\sum_{0 \leq i < j \leq n-1} f(i,j)$ denotes the sum of $f(i,j)$ for all values of i and j which satisfy $0 \leq i < j \leq n-1$.
11. Carry chain : Given an n-bit RCA and two specific n-bit binary numbers $\mathbf{a} = a_{n-1}a_{n-2}\ldots a_0$ and $\mathbf{b} = b_{n-1}b_{n-2}\ldots b_0$ as inputs to the RCA, define $\mathbf{a}^{\text{x}} = a_n^{\text{x}} a_{n-1}^{\text{x}} a_{n-2}^{\text{x}} \ldots a_0^{\text{x}}$ and $\mathbf{b}^{\text{x}} = b_n^{\text{x}} b_{n-1}^{\text{x}} b_{n-2}^{\text{x}} \ldots b_0^{\text{x}}$ where $a_i^{\text{x}} = a_i$, $b_i^{\text{x}} = b_i$ for $0 \leq i \leq n-1$ and $a_n^{\text{x}} = b_n^{\text{x}} = 0$. 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
    - $a_i^{\text{x}} = b_i^{\text{x}} = 1$. This case is referred to as the generation of a carry.
    - $a_w^{\text{x}} \neq b_w^{\text{x}}$. This case is referred to as the propagation of a carry.

- $a_j^x = b_j^x$. If $a_j^x = 0$, the carry is said to be killed and if $a_j^x = 1$ 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 $0 \leq i < w < j \leq n$ (or, if $j = i + 1$, $0 \leq i < j \leq n$ and $i \neq n - 1$). Please note that this definition of a carry chain has been adapted from [9].

12. $C_{ij}(\mathbf{a}, \mathbf{b})$: Consider an n-bit RCA and two specific n-bit binary numbers **a** and **b** as inputs to the RCA. A boolean variable $C_{ij}(\mathbf{a}, \mathbf{b})$ is defined on whether there is carry chain starting from position i and ending at position j such that

$$C_{ij}(\mathbf{a}, \mathbf{b}) = \begin{cases} 1 & \text{if there is a carry chain from i to j} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where $0 \leq i < j \leq n$.

13. N : Represents the number of gates in the RCA.

14. v_ℓ : Represents the supply voltage of the ℓ^{th} gate, where $1 \leq \ell \leq N$.

15. $\boldsymbol{v}$: Represents a vector of size N of all the supply voltages, i.e., the ℓ^{th} element in the vector $\boldsymbol{v}$ is v_ℓ, the supply voltage of the ℓ^{th} gate.

16. $\epsilon_\ell(v_\ell)$: Represents the worst-case propagation delay of the ℓ^{th} gate with a supply voltage of v_ℓ, where $1 \leq \ell \leq N$.

17. $\boldsymbol{\epsilon}(\boldsymbol{v})$: Represents a vector of size N where the elements are the worst-case propagation delays of the gates in the adder which is a function of the voltage vector.

18. t_{in} : Represents the time instant when the inputs are provided to the RCA.

19. t_k : Represents the absolute time instant when the correct s_k is computed where $0 \leq k < n + 1$. Therefore the time taken for the correct s_k to be computed is equal to $t_k - t_{\text{in}}$.

20. d_{pr} : Given an n-bit RCA and two specific n-bit binary numbers **a** and **b** and assuming that $C_{pq} = 1$ for $p < r < q$, then we define $d_{pr} = t_r - t_{\text{in}}$. d_{pr} is the time elapsed from the instant when the inputs are provided to the RCA till the correct sum bit, s_r, is generated.

21. $\mathbf{d}$: Represents an $(n + 1) \times (n + 1)$ matrix where the r^{th} element in the p^{th} row is d_{pr}. Elements d_{pr} for which $p \geq r$ are invalid and so are not considered.

22. D : This is the clock cycle time of the RCA, which is the difference in time between when the inputs are given to the RCA and the outputs are taken from the RCA.

23. $I_k(\mathbf{a}, \mathbf{b}, \mathbf{d}, D)$: Variable that denotes if there is an error and also the sign of the error (positive or negative) at the k^{th} output bit, where $0 \leq k < n + 1$, of an n-bit RCA with inputs **a**, **b** and clock cycle time D.

24. $I_k^{\text{cc}}(D, i, j, \mathbf{d})$: Variable that denotes if there is an error and also the sign of the error (positive or negative) at the k^{th} output bit only within a carry chain starting from position i to position j, where $0 \leq k < n + 1$ and $0 \leq i < j \leq n$, of an n-bit RCA (with a clock cycle time of D).

25. $I_k^{\text{E}}(D, i, j, \mathbf{d})$: Variable that denotes if there is an error at the k^{th} output bit only within a carry chain starting from position i to position j, where $0 \leq k < n + 1$ and $0 \leq i < j \leq n$, of an n-bit RCA (with a clock cycle time of D).

26. $\mathbf{I}_k(D, i, j, \mathbf{d})$: Variable that denotes if there is an error at the k^{th} output bit only within a carry chain starting from position i to position j, where $0 \leq k < n + 1$ and $0 \leq i < j \leq n$, of an n-bit RCA (with a clock cycle time of D).

27. Critical Path Delay : The worst case delay of a circuit for all possible inputs is called the critical path delay of the circuit.

28. Overclocked RCA: In general, an RCA's clock cycle time is greater than or equal to the RCA's critical path delay to avoid erroneous outputs. But in our methodology we challenge this constraint and operate the RCA at a clock cycle time lower than the critical path delay of the RCA. We call this type of RCA as an overclocked RCA and the general concept is called overclocking.

29. v_{min} : The minimum supply voltage permitted to be used for a gate in the process technology used.

30. v_{max} : The maximum supply voltage permitted to be used for a gate in the process technology used.

31. N_v : Represents the number of voltage levels that we consider in our experiments between v_{min} and v_{max}. For the target process technology of 90nm used in this paper, $v_{\text{min}} = 0.8V$, $v_{\text{max}} = 1.2V$, and, with a granularity of $0.01V$, N_v is equal to 40.

32. max-delay$_\ell$: Denotes $\epsilon_\ell(v_{\min})$ which is the upper bound on the maximum worst-case propagation delay of the ℓ^{th} gate.

33. min-delay$_\ell$: Denotes $\epsilon_\ell(v_{\max})$ which is the upper bound on the minimum worst-case propagation delay of the ℓ^{th} gate.

34. $\mathcal{A}$: Number of additions in a chosen benchmark used to calculate the switching activities in an RCA and also for an approximate RCA.

35. w_ℓ : Represents the switching activity of the ℓ^{th} gate in a ripple carry adder. Switching activity is defined as the average number of toggles per addition. The average is computed as the ratio of number of toggles at the output of the gate to the number of additions ($\mathcal{A}$) in a chosen benchmark. Hence this quantity is a positive real number.

36. $\mathbf{W}$: Represents a vector of size N where the ℓ^{th} element is w_ℓ.

37. w_ℓ^{a} : Represents the switching activity of the ℓ^{th} gate in an approximate ripple carry adder.

38. $\mathbf{W}^{\text{a}}$: Represents a vector of size N where the ℓ^{th} element is w_ℓ^{a}.

39. $E_\ell^{\text{dyn}}(v_\ell)$: The average dynamic energy consumption of the ℓ^{th} gate for a single transition when its supply voltage is equal to v_ℓ where $v_{\min} \leq v_\ell \leq v_{\max}$. The dynamic energy consumed by a gate for a single transition depends on many other factors such as (i) whether the transition is a LOW to HIGH or a HIGH to LOW, (ii) the input(s) and (iii) parasitic capacitances. For the gates and the target technology used in this paper, to compute $E_\ell^{\text{dyn}}(v_\ell)$ we consider an average energy consumption per transition over various input conditions.

40. $P_\ell^{\text{stat}}(v_\ell)$: Represents the static power consumed by the ℓ^{th} gate at a supply voltage of v_ℓ.

41. γ_ℓ : Represents the proportionality constant between the propagation delay and the average dynamic energy consumption of the ℓ^{th} gate for a single transition averaged over several supply voltages.

42. N_v: Denotes the number of various voltage levels that we use to compute the average proportionality constant γ_ℓ.

43. E : Represents the average (over different inputs to the adder) total energy consumption of an adder for a single addition.

44. Energy Budget : The total energy budget, in the context of a geometric program (see Section 10), allocated as per our approximate adder supply allocation problem which is described in Section 3.

45. $\mathcal{V}$: Denotes the set of possible supply voltages for supply voltage binning.

46. M_v : The maximum number of distinct voltages allowed by the circuit designer for supply voltage binning.

47. r-combination of a set S : Is a subset of set S with r elements.

48. $P(\mathcal{V})$: The power set is the set of all subsets of the set $\mathcal{V}$, including the empty set and the set $\mathcal{V}$ itself [10].

49. m_i : Let $p \in P(\mathcal{V})$. Then m_i denotes the number of gates in the adder with a supply voltage equal to the i^{th} element of p.

50. Globally optimized RCA : RCA with a voltage allocation scheme that is generated as a result of solving our target problem is referred to as a globally optimized RCA.

51. Conventional correct RCA : RCA without overclocking whose outputs are always correct.

The terms that we have defined in this section serve as a reference for the models and procedures described in this paper.

5 Prior Work

In this section we discuss a few instances of prior work which are related to our target problem in this paper. We discuss the prior work based on the different techniques that they use in the following order: (i) multiple voltages and (ii) overclocking. We also discuss the distinction between our approach and the prior work discussed.

Techniques such as the ones by Blaauw et al. [11] and Broderson et al. [12] are *adaptive* which means that the throughput of the circuit is based on the workload. Essentially, the throughput is adjusted by modifying the supply voltages of the circuit dynamically based on the input load at that particular time as opposed to operating at the worst case frequency at all times. Non-adaptive techniques typically operate a circuit at multiple voltages which might rely on circuit implementation techniques like transistor sizing for energy efficiency [13]. Manzak and Chaktrabarti [14] as well as Yeh et al. [15, 16] 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 [17, 18] 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 [17, 18] can be used.

Tschanz et al. [19] describe the development of timing error resilient circuitry including a prototype chip that lets timing errors happen and then corrects them using less power overall. Shim et al. [20] show a design in which circuit level timing errors are corrected using signal processing techniques. Soft digital signal processing by Hegde and Shanbhag et al. [21] allows errors in the computation but then uses digital noise tolerance schemes to attempt to correct these errors though partially in some cases. Banerjee et al. [22] present a specialized two-dimensional discrete cosine transform circuit where computations that are more important have a lower critical path delay than the rest. Thus the chance of timing errors happening in the important computations is lower than the rest.

George et al. [2] present 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 [2] built on the result shown by Cheemalavagu et al. [23] 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. The prior approaches, on the other hand, have either been ad-hoc circuit level approaches or algorithmic level optimizations (altering the algorithm being executed).

6 RCA Delay Assumptions

In this section, we explain some basic assumptions on the variables that are related to propagation delay of gates and timing that we use in this paper.

We need to clarify some concepts about variables related to propagation delay and timing that we use in this paper because the technique of "approximate circuits" is, relatively, a novel circuit design technique. Before the conception of Probabilistic CMOS (PCMOS) by Palem et al. [24, 25, 26], four of the major VLSI circuit design techniques were digital VLSI design, asynchronous VLSI design, analog VLSI design and radio frequency VLSI design. In digital design, if we consider a certain circuit, the circuit designer can choose to ignore the intermediate results that the circuit produces as long as the final output of the circuit when the output is read at the end of the clock cycle is correct. But approximate circuits are overclocked, which means that the intermediate values can no longer be ignored. Also, similar to the case of asynchronous logic design, in approximate circuits the consideration of worst-case vs. average-case vs. best-case propagation delays of an approximate circuit is important to properly characterize the output error of the circuit. When we refer to worst-case or average-case or best-case propagation delay of a circuit, we consider the variation in the propagation delay of the circuit for different input transitions in our simulations at a given fixed supply voltage. There are many other factors that affect the propagation delay such as temperature and process variations. For the simulations done in this paper we consider that these factors do not change spatially or temporally. Also, for the purpose of this paper, we consider only worst-case errors at all places. To clarify, worst-case delay of a gate is characterized for various values of supply voltages. For example, in this paper we compute worst-case delays for $N_\nu = 40$ equally spaced supply voltage levels between $\nu_{\text{min}} = 0.8V$ and $\nu_{\text{max}} = 1.2V$ for 90nm technology. At this point, we have not modeled the effect on the output error of an approximate adder if average-case or best-case propagation delays are used. Instead, we start with an analysis of approximate RCAs assuming worst case delays. We do not have a formal proof to show that average and best case delays have

lower error rates; however, empirically we have found that increasing the delay increases the errors in the adder at least 98% of the time (see Section 13.4).

In the rest of this paper, we consider circuits which are RCAs. Each gate in a particular RCA could potentially (but not practically) be supplied with a different supply voltage. As per the definition in Section 4, the worst-case propagation delay (worst-case with respect to different input combinations) of the k^{th} gate at supply voltage v_k is denoted as $\epsilon_k(v_k)$. It has been shown by many independent sources [27, 28] that the propagation delay of a gate is inversely proportional to its supply voltage. This relationship was also used by Chakrapani et al. [3] in the modeling of the effect of biased voltage scaling in an approximate ripple carry adder. For our modeling, the values of propagation delays of gates for a given supply voltage are measured directly from HSPICE for our target technology. To relate the two quantities we use the same relationship, that is, $\epsilon_\ell(v_\ell)$ is inversely proportional to the supply voltage of the ℓ^{th} gate, v_ℓ. From Section 4, we know that min-delay$_\ell$ denotes the worst-case delay of the ℓ^{th} gate when the supply voltage is equal to $v_{\max}$ and max-delay$_\ell$ denotes the worst-case delay of the ℓ^{th} gate when the supply voltage is equal to $v_{\min}$. Thus min-delay$_\ell \leq \epsilon_\ell(v_\ell) \leq$ max-delay$_\ell$.

For our error and energy modeling we assume for the sake of simplicity that all the carry bits are zero when the inputs are provided to the adder for each addition. In reality the carry bits may retain values from the prior computations and therefore would not be reset to zero every time. Further discussion about the effect of non-zero carry bits on error and energy modeling is presented in Appendix A.

In addition to the above, we assume that the clock cycle time chosen for overclocking (D) is never less than the propagation delay of a single full adder.

In this section we have discussed the type of propagation delays that we use in this paper. We have also described our assumptions regarding the worst-case propagation delay of each gate in the RCA circuits that we consider in this paper.

7 Energy Modeling Assumptions

In this section we discuss the main assumptions we make in this paper about our energy models.

The total energy consumption of a circuit consists of two separate components, dynamic energy consumption and static energy consumption. The dynamic energy consumption constitutes the energy spent during the charging and discharging of capacitive loads during logic changes. The average dynamic energy consumed by a CMOS circuit thus depends on the number of logic changes which is denoted by the switching activity of the adder circuit. We define (see Section 4) the switching activity of gate ℓ, denoted as w_ℓ, as the average number of logic changes that the ℓ^{th} gate undergoes in a single addition. w_ℓ for each gate 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 (say $\mathcal{A}$) of additions in a chosen benchmark. Therefore,

$$w_\ell = \frac{\text{Total number of toggles of gate } \ell}{\mathcal{A}}$$

To use our energy model to solve our target problem, we need to represent the average dynamic energy consumption ($E_\ell^{\text{dyn}}(v_\ell)$) of a gate in terms of the worst-case propagation delay ($\epsilon_\ell(v_\ell)$) of that gate.

From Section 4, $\epsilon_\ell(v_\ell)$ denotes the worst-case propagation delay of the ℓ^{th} gate when its supply voltage is v_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 and, as described in Section 6, the propagation delay of a gate is inversely proportional to its supply voltage. To represent $E_\ell^{\text{dyn}}(v_\ell)$ in terms of $\epsilon_\ell(v_\ell)$ we 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.

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

where γ_ℓ is the proportionality constant for the ℓ^{th} gate.

The proportionality constant, γ_ℓ, is dependent on the process technology of the ℓ^{th} gate. γ_ℓ is computed by taking the average, over several supply voltage levels, of the product of the square of the worst-case (over all possible input transitions) propagation delay and the switching energy consumption of the ℓ^{th} gate. Thus γ_ℓ is computed for the ℓ^{th} gate as follows

$$\gamma_\ell = \frac{1}{N_v} \sum_{i=1}^{N_v} \epsilon_\ell^2(v_\ell^i) \times E_\ell^{\text{dyn}}(v_\ell^i) \tag{3}$$

where N_v is the number of various voltage levels that we use to compute the average proportionality constant and v_ℓ^i is the i^{th} (out of N_v) voltage supplied to the ℓ^{th} gate.

Thus, γ_ℓ is computed separately for each type of gate. For example, in the design of the RCA that we consider there are two types of gates, XOR and MUX. An example of computing the proportionality constant for the XOR gate in 90nm technology is given in Example 1.

sec:energymodelassumptions

Table 1: Propagation delay and average dynamic energy per transition of an XOR gate in 90nm process technology for various supply voltage values

v_ℓ	$E_\ell^{\text{dyn}}(v_\ell)$ (femto-J)	$\epsilon_\ell(v_\ell)$ (pico-sec)	$E_\ell^{\text{dyn}}(v_\ell) \times \epsilon_\ell^2(v_\ell)$ (10^{-35}Jsec^2)
0.8	8.63	46.89	1.90
0.9	11.18	41.61	1.94
1	14.30	37.93	2.06
1.1	17.71	35.26	2.20
1.2	21.65	33.33	2.41

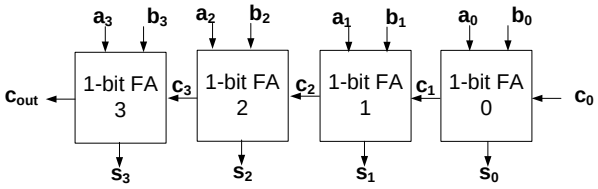


Figure 1: Diagram of a 4-bit ripple carry adder

Example 1. Consider an XOR gate in our target 90nm process technology. We will consider 5 specific voltages for this example and compute the proportionality constant. Refer to Table 1 for the values of worst-case propagation delay and average dynamic energy consumption per transition for the different supply voltage values. The first column shows the various supply voltage values. The last column shows the product of the average dynamic energy per transition and the square of the propagation delay which denotes the proportionality constant, γ_ℓ, as per Eq. 2. For our modeling and simulations, we take an average of the proportionality constant over various supply voltages as shown in Eq. 3 and in this example the average proportionality constant turns out to be 2.1×10^{-35}Jsec^2. □

8 Error Model for an Approximate RCA

In this section we first present a brief discussion of a ripple carry adder. This is followed a description of the mathematical framework and the models that we use to characterize the average error at the output of an approximate RCA.

8.1 Description of a ripple carry adder

A ripple carry adder (RCA) is the most basic adder design that is typically considered. Consider an n-bit ripple carry adder with inputs to the adder being denoted as **a**, **b** and c_0, where **a** and **b** are two n-bit binary numbers and c_0 is the carry input bit to the full adder at the least significant bit (LSB) position. The RCA is made up of a series of blocks called full adders. Each full adder is a collection of logic gates which adds two input bits and a carry input to produce a sum output and a carry output. This means that the i^{th} full adder adds the two input bits a_i and b_i (a_i and b_i are the i^{th} bits in **a** and **b**, see Section 4) and the carry input c_i to produce s_i, the i^{th} sum bit, and c_{i+1}, the $(i+1)^{\text{th}}$ carry bit. The sum of the adder is $\mathbf{s} = s_n s_{n-1} s_{n-2} \ldots s_0$. This adder is called a ripple carry adder because the carry ripples through the adder one full adder at a time from the least significant bit to the most significant bit.

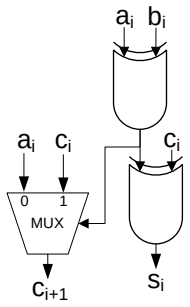


Figure 2: Diagram of a 1-bit full adder (1-bit FA)

Table 2: Maximum and minimum propagation delays of the XOR gate and the MUX in 90nm technology

$Gate$	min-delay$_\ell$ (pico-sec)	max-delay$_\ell$ (pico-sec)
XOR	33.3	55.2
MUX	30.5	51.2

A diagram of a 4-bit ripple carry adder is shown in Fig. 1. In the diagram, each full adder is shown as a single block which is in fact a collection of logic gates. The logic diagram for each full adder is shown in Fig. 2. We admit that this might not be the best full adder design to optimize for area or speed, but we use Fig. 2 nonetheless for its simplicity. There are two types of gates present in the full adder design shown in Fig. 2, an XOR-gate and a multiplexer (MUX). We present the transistor level diagrams of these gates that we use for our simulations in Appendix B. In this paper, we will be using the delay values of these gates that were computed from HSPICE simulations. The minimum (for supply voltage of 0.8V) and maximum (for supply voltage of 1.2V) delay values of the two types of gates in Fig. 2 using 90nm technology are shown in Table 2.

Note that a single n-bit RCA based on Fig. 2 and Fig. 1 would have $N = 3 \times n$ gates. In general, we will use the following indexing scheme in this paper to refer to a gate in an RCA.

- The top XOR gate in the full adder in Fig. 2 that

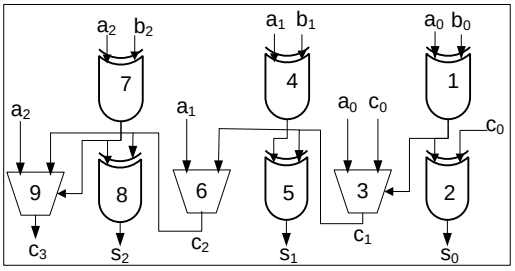


Figure 3: Diagram of a 3-bit RCA with all the gates indexed

computes $a_k \oplus b_k$ at position k would be referred to using the index $3k+1$ where the n full adders in the RCA are indexed by $0 \leq k \leq n-1$ according to bit position (as shown in Fig. 1).

- The bottom XOR gate in the full adder as per Fig. 2 that computes the sum output, s_k, at position k would be referred to using the index $3k+2$.
- The one and only multiplexer in the full adder at position k would be referred to using the index $3k+3$.

We demonstrate this indexing scheme in Example 2.

Example 2. An example of a 3-bit RCA at the gate level is shown in Fig. 3. Consider the 1-bit FA at bit position '0'; then, per the indexing scheme described in Section 8.1, the top XOR gate that computes $a_0 \oplus b_0$ should be referred to by the index $3 \times 0 + 1 = 1$. Similarly, the bottom XOR gate that computes the sum output bit (in this case at position '0') s_0 should be referred to by the index $3 \times 0 + 2 = 2$. The only multiplexer in the full adder at bit position '0' is referred to by the index $3 \times 0 + 3 = 3$. The indices for the other gates in Fig. 3 can also be deduced by the same indexing scheme.

□

8.2 Modeling the error at the output of an approximate RCA

In this section, we first present boolean logic functions we use to represent the outputs of an RCA. Then we define and describe carry chains in ripple carry adders. We then describe the effect of carry chains on error at the output of an approximate ripple carry adder. We also present the behavior of errors at the output of an overclocked RCA in the presence of a carry chain. We then describe the procedure that we follow to characterize the time, denoted as d_{pr} (see Section 4), it takes for a specific sum bit in an RCA to be correctly computed. We describe a mathematical formulation for the error function modeling the error at the output of an approximate RCA.

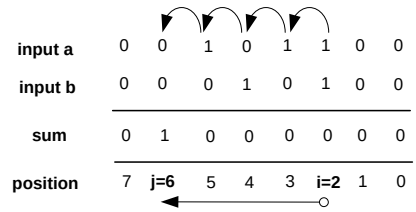


Figure 4: An example of a carry chain in a binary addition using an RCA

8.2.1 RCA logic

In this subsection we present one set of (from among many) boolean logic functions of binary addition with respect to an RCA. We will also discuss the effect of a carry chain on the outputs of an RCA.

We consider n-bit RCAs for some n. Numbers to be added will be in the range $0, \ldots, 2^n - 1$ and will have the standard binary representation. In this paper, we will show our procedure for modeling and analysis of only unsigned ripple carry adders. We agree that many applications require handling of signed numbers as well, typically in 2's complement format. But because typical 2's complement adders utilize an unsigned adder to propagate the carry bits with additional circuitry around them to handle the sign, it appears reasonable to analyze errors due to carry propagation in an unsigned adder as an initial step.

Consider the addition of two n-bit numbers $\mathbf{a}$ and $\mathbf{b}$ in an n-bit RCA, resulting in an $(n+1)$-bit number, consisting of $\mathbf{s} = s_n s_{n-1} \ldots s_0$. Thus sum $\mathbf{s}$ is computed in an RCA as $s_k = a_k \oplus b_k \oplus c_k$ for $0 \leq k \leq n-1$. The sequence of carries $\mathbf{c}$ is computed as follows. c_0 is input (and is zero for addition) while $c_{k+1} = (a_k \oplus b_k) \cdot c_k + \overline{(a_k \oplus b_k)} \cdot a_k$ for $0 \leq k \leq n-1$. Note that we have yet to define s_n; in fact, the last sum bit s_n is defined as being equal to the carry bit c_n.

8.2.2 Carry chains in ripple carry adders

In this section, we discuss carry chains in the context of an RCA.

We repeat the definition of a carry chain as described in Section 4. Given an n-bit RCA and two specific n-bit binary numbers $\mathbf{a} = a_{n-1}a_{n-2}\ldots a_0$ and $\mathbf{b} = b_{n-1}b_{n-2}\ldots b_0$ as inputs to the RCA, define $\mathbf{a}^{\mathrm{x}} = a_n^{\mathrm{x}} a_{n-1}^{\mathrm{x}} a_{n-2}^{\mathrm{x}} \ldots a_0^{\mathrm{x}}$ and $\mathbf{b}^{\mathrm{x}} = b_n^{\mathrm{x}} b_{n-1}^{\mathrm{x}} b_{n-2}^{\mathrm{x}} \ldots b_0^{\mathrm{x}}$ where $a_i^{\mathrm{x}} = a_i$, $b_i^{\mathrm{x}} = b_i$ for $0 \leq i \leq n-1$ and $a_n^{\mathrm{x}} = b_n^{\mathrm{x}} = 0$.

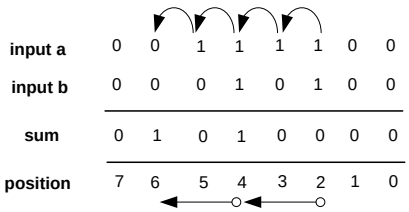


Figure 5: An example of two contiguous carry chains in a binary addition using an RCA

A carry chain is said to be present in the n-bit RCA with inputs **a** and **b** from position i to position j if and only if

- $a_i^{\mathrm{x}} = b_i^{\mathrm{x}} = 1$. This case is referred to as the generation of a carry.
- $a_w^{\mathrm{x}} \neq b_w^{\mathrm{x}}$. This case is referred to as the propagation of a carry.
- $a_j^{\mathrm{x}} = b_j^{\mathrm{x}}$. If $a_j^{\mathrm{x}} = 0$, the carry is said to be killed and if $a_j^{\mathrm{x}} = 1$ 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 $0 \leq i < w < j \leq n$ (or, if $j = i + 1$, $0 \leq i < j \leq n$ and $i \neq n - 1$).

The reason for the definition of $\mathbf{a}^{\mathrm{x}}$ and $\mathbf{b}^{\mathrm{x}}$ is that we have to take into account the carry output of the full adder at the Most Significant Bit (MSB) position since this ca rry output is the MSB of the sum. Therefore there might be a situation where a carry that is generated at position i (for $0 \leq i \leq n - 2$) is killed at position $n - 1$ because there is no full adder to propagate after that. The above definition captures this effect.

In general, if there is a carry chain from i to j, we will set a boolean variable $C_{ij}(\mathbf{a}, \mathbf{b})$ to 1—see definition of $C_{ij}(\mathbf{a}, \mathbf{b})$ in Section 4. And 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 $k \in \{i + 1, \ldots, j - 1\}$; 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 can easily determine the correct sum bits from position $(i + 1)$ to position j, i.e., $s_{i+1} = s_{i+2} = \cdots = s_{j-1} = 0$ and $s_j = 1$.

In a ripple carry adder, e.g., with a design as described in Section 8.1, in the worst-case the carry propagates from the lowest significant bit position to the most significant bit position.

Example 3. An example of a carry chain in a binary addition using an RCA is shown in Fig. 4. As is denoted in the figure, the carry chain starts from position $i = 2$ and ends at position $j = 6$, and therefore we will say that there is a carry chain from 2 to 6 and will also denote $C_{26} = 1$. The outputs of the RCA, the sum bits, are shown in Fig. 4. As described earlier in this section, the correct sum bits between position 3 and 6 are fixed based on the fact that there is a carry chain from position 2 and 6. □

Definition 1. Consider an n-bit addition with inputs **a** and **b**. If there exists two carry chains such that $C_{ij}(\mathbf{a}, \mathbf{b}) = 1$ and $C_{xy}(\mathbf{a}, \mathbf{b}) = 1$, then these carry chains are said to *overlap* if and only if $i \leq x < j \leq y$.

The case shown in Example 3 has a single carry chain across the entire 8-bit addition. While technically there can be more than one carry chain in a single addition, the carry chains cannot overlap (as per Definition 1) over each other in any case. We show this observation to be true in Observation 1.

Observation 1. Consider an n-bit addition with inputs **a** and **b**. As per the definition of a carry chain in Section 4, two carry chains cannot overlap. Specifically there cannot exist $C_{ij}(\mathbf{a}, \mathbf{b}) = 1$ and $C_{xy}(\mathbf{a}, \mathbf{b}) = 1$ such that $i \leq x < j \leq y$.

Proof. We will prove this using the method of contradiction.

From Section 4, $C_{ij}(\mathbf{a}, \mathbf{b}) = 1$ in an n-bit RCA if and only if

- $a_i^{\mathrm{x}} = b_i^{\mathrm{x}} = 1$.
- $a_w^{\mathrm{x}} \neq b_w^{\mathrm{x}}$.
- $a_j^{\mathrm{x}} = b_j^{\mathrm{x}}$.

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

Assume that $C_{ij}(\mathbf{a}, \mathbf{b}) = 1$ and $C_{xy}(\mathbf{a}, \mathbf{b}) = 1$ in the same n-bit addition such that $i \leq x < j \leq y$. From the definition, if $C_{ij}(\mathbf{a}, \mathbf{b}) = 1$ then for all $0 \leq i < w < j \leq n$ it is known that $a_w^{\mathrm{x}} \neq a_w^{\mathrm{x}}$. But if $C_{xy}(\mathbf{a}, \mathbf{b}) = 1$, then we know from the definition of the carry chain that $a_x^{\mathrm{x}} = b_x^{\mathrm{x}} = 1$. But from our assumption the carry chains are such that $i \leq x < j$. Consider the case where $w = x$. This results in a contradiction because if $C_{ij}(\mathbf{a}, \mathbf{b}) = 1$ then $a_w^{\mathrm{x}} \neq b_w^{\mathrm{x}}$ but if $C_{xy}(\mathbf{a}, \mathbf{b}) = 1$ then $a_w^{\mathrm{x}} = b_w^{\mathrm{x}}$.

Hence the observation holds.

□

Also in Example 3, the carry chain that started at position $i = 2$ was killed at position $j = 6$ where $a_6 = b_6 = 0$. But as described in the definition of a carry chain, it could also have been killed if $a_j = b_j = 1$. We clarify these points in Example 4.

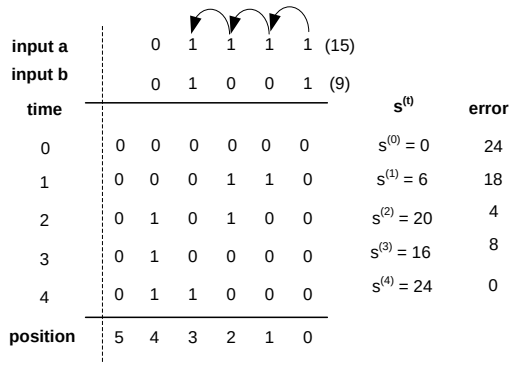


Figure 6: An example to demonstrate that in an RCA it is possible that by increasing the supply voltages the accuracy of the sum is decreased.

Example 4. An example of two contiguous carry chains in a binary addition using an RCA is shown in Fig. 5. In this case the first carry chain starts at position 2 and is killed at position 4. But because $a_4 = b_4 = 1$ another carry chain is generated at position 4 which is killed at position 6. Thus in this case both $C_{24}(\mathbf{a}, \mathbf{b}) = 1$ and $C_{46}(\mathbf{a}, \mathbf{b}) = 1$. This shows that even though $c_2 = 1$, $c_3 = 1$, $c_4 = 1$ and $c_5 = 1$, it actually consists of two separate carry chains. Also as described in the definition of a carry chain, even a position that generates a new carry chain can also kill a previous carry chain. Furthermore, as per the definition of a carry chain, multiple carry chains can exist in a single addition but cannot overlap. This is clearly shown by the directed arrows in Fig. 4 and Fig. 5. □

8.2.3 The relationship between RCA carry chains and overclocking errors

As mentioned in Section 6, 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 with all gates supplied $\nu_{\min}$. Considering an approximate RCA, this would imply that there would be a possibility of error at the output of an RCA only if there is propagation of carry. Because 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. Note that this is true only if we assume that the temporary values of all the carry bits in the adder are zeros at the beginning of the addition. As explained in Section 6, in this paper we assume that that all the carry bits are zero when the inputs are provided to the adder for each addition.

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 will now describe a point which is worth noting about RCA error with respect to a carry chain. The point is that in general [3, 4], it has been assumed that to increase the output accuracy of an overclocked RCA the supply voltage(s) of the gates in the circuit of the RCA should be increased which in turn results in increasing the energy consumption of the circuit of the RCA. However, due to overclocking in the presence of a carry chain, *increasing* the voltage provided to the components of the RCA may also result in *decreasing* the accuracy of the sum read from the RCA in some cases. An example of such a case is presented in Example 5.

Example 5. Consider a 5-bit ripple carry adder (RCA) adding 01111 and 01001. The gate-level design of the RCA is described in Section 8.1. Consider the carry chain from position 0 till position 3 as shown in Fig. 6. For the sake of this example, we will assume that all the full adders in the RCA have the same propagation delays. We are excluding the cases where the propagation delays change because of random variations and also different input transitions and hence are assuming that the propagation delays are exact. Also define $\mathbf{s}^{(t)}$, just for this example, to denote the binary sum output of the 5-bit RCA at the t^{th} instant, where t ranges from 0 (which denotes the output before the inputs were given to the adder) till the correct output is computed where each subsequent step denotes a change in the sum output. Then the computed value of $\mathbf{s}$ starts with the initial value of $\mathbf{s}^{(0)} = 000000$ (output value 0), consecutively becomes $\mathbf{s}^{(1)} = 000110$ (output value 6), $\mathbf{s}^{(2)} = 010100$ (output value 20), $\mathbf{s}^{(3)} = 010000$ (output value 16), and $\mathbf{s}^{(4)} = 011000$ (output value 24). As the correct sum is 24, the consecutive errors at $t =$ 1 to 4 are 18, 4, 8, and 0. So, assuming some specified overclocked clock cycle time, if the voltage supplied allowed the reading of $\mathbf{s}^{(3)}$, the resulting error would be 8. In contrast, if the voltage supplied were lower, allowing only the reading of $\mathbf{s}^{(2)}$ with the same overclocking, the resulting error would be only 4. □

In this subsection we discussed the behavior of errors in an RCA due to overclocking in the presence of carry chains. We also show using an example the output error of an approximate RCA might increase when the supply voltage(s) of the RCA is(are) increased.

8.2.4 Critical path of a sum bit in a carry chain of an RCA

In this subsection we define and discuss the critical path (referred to as the *sum path*) of a particular sum bit in the

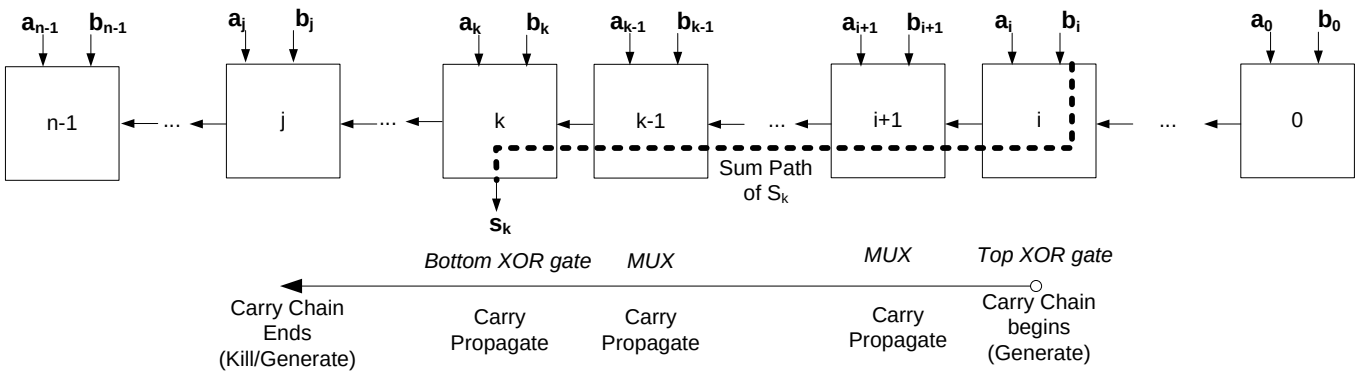


Figure 7: Diagram of an n-bit ripple carry adder to describe the modeling of the critical path of a sum bit in a carry chain

middle of a carry chain in an RCA.

We call the longest delay path with respect to a particular sum bit s_k as the "sum path for s_k." In an RCA, the sum path for bit 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 **a** and **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 **a** and **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 **a** and **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 $d_{ik} = t_k - t_{\text{in}}$. For the special case when the carry chain is from i to $j = n$, t_j is the time instant when carry c_n is generated. **d** denotes the $(n+1) \times (n+1)$ matrix of all d_{ik}, $0 \leq i < k \leq n$. Properly speaking, **d** for a particular RCA in a particular technology is a function of the critical paths of the sum bits of the RCA, v_i for each gate i in any critical path of any sum bit, and $\epsilon_i(v_i)$; however, for brevity, in this paper we will simply refer to **d** without specifying all the input values on which **d** depends.

Example 6. Consider an n-bit ripple carry adder, shown in Fig. 7, based on the gate level description described in Section 8.1. Assume that $C_{ij}(\mathbf{a}, \mathbf{b}) = 1$. In this adder, d_{ik} in the worst-case would be the sum of propagation delays of the top XOR gate in the full adder (referring to Fig. 2) at position i, MUX gates of the full adders from position $i+1$ to position $k-1$ and the bottom XOR gate in the full adder (referring to Fig. 2) at position k. This is the "sum path of bit s_k" assuming that $C_{ij}(\mathbf{a}, \mathbf{b}) = 1$. □

The "sum path of a bit" in an RCA is further clarified through Example 7.

Example 7. Consider the 3-bit ripple carry adder shown in Fig. 8 based on the gate level model of an RCA described in Section 8.1. 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 (defined in Section 4) 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 GATE-1 (where GATE-k denotes the gate with the number k inside it), GATE-3, GATE-6 and GATE-8. The critical path corresponding to d_{02} is shown in Fig. 8 as a dotted line from bit position 0 to the sum output in bit position 2. This means that 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, considering maximum supply voltages for all the gates, i.e., $v_1 = v_3 = v_6 = v_8 = 1.2V$, we get $d_{02} = 33.3$ ps $+30.5$ ps $+30.5$ ps $+33.3$ ps $= 127.6$ ps.

With calculations similar to d_{02}, we find that

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

□

We presented a description of the critical path delay of a sum bit in a carry chain in terms of the propagation delays of the gates in an RCA.

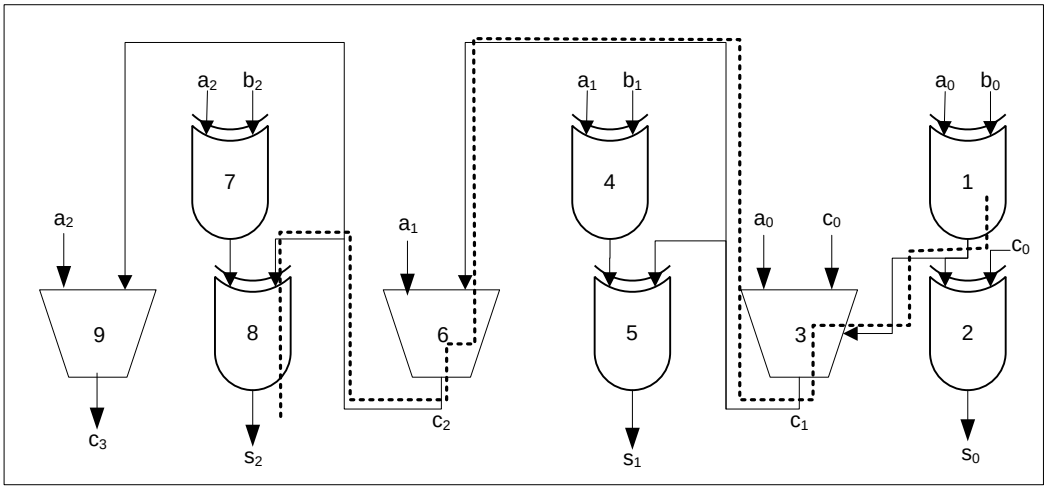


Figure 8: Gate level diagram of a 3-bit ripple carry adder showing the sum path for bit s_2

8.2.5 RCA error based on an analysis of carry chain errors

In this subsection, we will develop a function for the error at the output of an overclocked RCA. We define the error at the output of the RCA as a function of **a**, **b**, the topology of the RCA, $\epsilon_k(\upsilon_k)$ and D.

The circuit of an RCA is built of gates. The RCA is given time D for each addition.

The sum outputs are read at time $t_{\text{in}} + D$, with D independent of the inputs. Due to overclocking, which we assume, the sum actually read, $\mathbf{s}^{\text{a}}$, may be different from $\mathbf{s} = \mathbf{a} + \mathbf{b}$. We now proceed to characterize the absolute magnitude of the error $\mid \mathbf{s}^{\text{a}} - \mathbf{s} \mid$. We define an *indicator function* as follows:

$$I_k(\mathbf{a}, \mathbf{b}, \mathbf{d}, D) = \begin{cases} 1 & \text{if } \exists\, i, j \text{ such that } C_{ij}(\mathbf{a}, \mathbf{b}) = 1, \\ & i < k < j \text{ and } d_{ik} > D \\ -1 & \text{if } \exists\, i, j \text{ such that } C_{ij}(\mathbf{a}, \mathbf{b}) = 1, \\ & i < k = j \text{ and } d_{ik} > D \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

If k is in some carry chain and the correct sum is 0 but a sum bit of 1 is read, then there is positive error and thus $I_k = 1$. If the correct sum of a bit in a carry chain is 1 and a 0 is read, then $I_k = -1$, indicating negative error. If the correct value of the sum bit in a carry chain is read or if the sum bit is not in a carry chain at all, then there is no error and thus $I_k = 0$. Our approach to apply the indicator function to indicate presence of a positive or negative error at a particular bit position is demonstrated in Example 8.

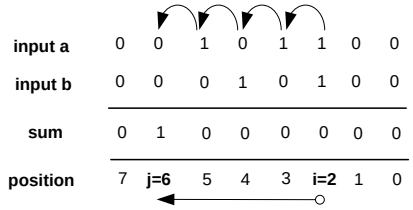


Figure 9: An example of a carry chain in a binary addition using an RCA

Example 8. Consider the 8-bit addition being performed in Fig. 4, repeated here as Fig. 9 for convenience. Let us apply the definition of the indicator function to this addition, assuming an 8-bit ripple carry adder based on the ripple carry adder described in Section 8.1 . Considering υ_{max} to be supplied to all the gates, the delays of the gates are given in Table 2 in Section 8.1. For this example, let us assume that the clock cycle time, D, is 130ps. We then calculate **d** using the method described in Section 8.2.4. For the purpose of this addition, because there is only one carry chain, $C_{26}(\mathbf{a} = 44, \mathbf{b} = 20) = 1$, we will only compute the following: $d_{23} = 97.1$ps, $d_{24} = 127.6$ps, $d_{25} = 158.1$ps and $d_{26} = 188.6$ps. Using the values of **d** and $D = 130$ps, we can compute $I_k(44, 20, \mathbf{d}, 130\text{ps})$ for $0 \leq k \leq n$ as shown in Eq. 4.

- $k = 0$: There is no i, j such that $C_{ij}(44, 20) = 1$ and $0 \leq i < k \leq j$. Therefore $I_0(44, 20, \mathbf{d}, 130\text{ ps}) = 0$.
- $k = 1$: There is no i, j such that $C_{ij}(44, 20) = 1$ and $0 \leq i < k \leq j$. Therefore

$I_1(44, 20, \mathbf{d}, 130\text{ps}) = 0$.

- $k = 2$: From the definition of a carry chain in Section 4, we know that $C_{26}(44, 20) = 1$, but $k = 2$ does not satisfy $0 \leq i < k \leq j$. Therefore $I_2(44, 20, \mathbf{d}, 130\text{ps}) = 0$.
- $k = 3$: We know that $C_{26}(44, 20) = 1$ and for $k = 3$ it satisfies $0 \leq i < k < j$. But $d_{23} = 97.1\text{ps} \leq D = 130\text{ps}$, therefore $I_3(44, 20, \mathbf{d}, 130\text{ps}) = 0$.
- $k = 4$: We know that $C_{26}(44, 20) = 1$ and for $k = 4$ it satisfies $0 \leq i < k < j$. But $d_{24} = 127.6\text{ps} \leq D = 130\text{ps}$, therefore $I_4(44, 20, \mathbf{d}, 130\text{ps}) = 0$.
- $k = 5$: We know that $C_{26}(44, 20) = 1$ and for $k = 5$ it satisfies $0 \leq i < k < j$. Also $d_{25} = 158.1\text{ps} > D = 130\text{ps}$, therefore as per the first line of the indicator function in Eq. 4, $I_5(44, 20, \mathbf{d}, 130\text{ps}) = 1$.
- $k = 6$: We know that $C_{26}(44, 20) = 1$ and for $k = 6$ it satisfies $0 \leq i < k = j$. Also $d_{26} = 188.6\text{ps} > D = 130\text{ps}$, therefore as per the second line of the indicator function in Eq. 4, $I_6(44, 20, \mathbf{d}, 130\text{ps}) = -1$.
- $k = 7$: There is no i, j such that $C_{ij}(44, 20) = 1$ and $0 \leq i < k \leq j$. Therefore $I_7(44, 20, \mathbf{d}, 130\text{ps}) = 0$.
- $k = 8$: There is no i, j such that $C_{ij}(44, 20) = 1$ and $0 \leq i < k \leq j$. Therefore $I_8(44, 20, \mathbf{d}, 130\text{ps}) = 0$.

□

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

Theorem 1.

$$Er(D, \mathbf{a}, \mathbf{b}, \mathbf{d}) = \sum_{k=0}^{n} I_k(\mathbf{a}, \mathbf{b}, \mathbf{d}, D) 2^k. \quad (5)$$

Proof. To prove the theorem, we will describe the three cases of the definition of $I_k(\mathbf{a}, \mathbf{b}, \mathbf{d}, D)$, as given in Eq. 4, which is used in the right hand side of Eq. 5.

1. Consider the first line in the right hand side of Eq. 4. This is the case where the correct c_k has not been computed by time D because $d_{ik} > D$. Here, $c_k = 1$ and $a_k \oplus b_k = 1$ because we know that $C_{ij}(\mathbf{a}, \mathbf{b}) = 1$ and $i < k < j$. So, $s_k = 0$ but the correct c_k has not been computed by time D and thus $s_k^{\text{a}} = 1$. Therefore, the error at bit position k contributed $(s_k^{\text{a}} - s_k) 2^k = (1 - 0) 2^k = I_k(\mathbf{a}, \mathbf{b}, \mathbf{d}, D) 2^k$ to the total error.
2. Consider the second line in the right hand side of Eq. 4. In this situation the correct c_j (because $k = j$) has not been computed by time D because $d_{ij} > D$. Here $c_j = 1$ and either $a_j = b_j = 0$ or $a_j = b_j = 1$ because we know that $C_{ij}(\mathbf{a}, \mathbf{b}) = 1$ and $k = j$. So, $s_j = 1$ but the correct c_j has not been computed by time D and thus $s_j^{\text{a}} = 0$. Therefore, the error at bit position j contributed $(s_j^{\text{a}} - s_j) 2^j = (0 - 1) 2^j = I_j(\mathbf{a}, \mathbf{b}, \mathbf{d}, D) 2^j$ to the total error.
3. Consider the third line in the right hand side of Eq. 4. This is the case where there is no error at the output of the adder. We have two situations in which there is no error.[1]
   (a) The first situation is when the particular bit-position is in a carry chain but the correct c_k has been computed by time D because $d_{ik} \leq D$. Therefore $C_{ij}(\mathbf{a}, \mathbf{b}) = 1$ and $i < k \leq j$. This means that $s_k^{\text{a}} = s_k$. Therefore there was no error at k and $0 = I_k(\mathbf{a}, \mathbf{b}, \mathbf{d}, D) 2^k$.
   (b) The second situation is when the particular bit position is not in a carry chain. As we mentioned in Section 6, the clock cycle time is at least greater than any possible delay of a single full adder. So if the particular full adder is not in a carry chain then there would not be any error at that position. Therefore, there are no i and j for which $C_{ij}(\mathbf{a}, \mathbf{b}) = 1$ and $i < k \leq j$. So, $c_k = 0$. Therefore there was no error at k and $0 = I_k(\mathbf{a}, \mathbf{b}, \mathbf{d}, D) 2^k$.

□

8.2.6 Summary of Subsection 8.2

The goal of Subsection 8.2 is to develop a function for the error at the output of an approximate RCA. To do this, we first describe boolean logic functions for the bits computed in an RCA. We then describe the reason that, given our assumptions, errors at the RCA output occur only in the presence of carry chains. We also describe how the outputs of an RCA are affected in the presence of a carry chain.

[1] The case where $k = 0$ comes under this case because there is no error at this position since it is assumed that D is always greater than or equal to the worst-case delay of a single full adder. Hence there is no situation in which there could be an error at this position.

Also, we show using a couple of examples the behavior of RCA error due to overclocking in a carry chain. We then describe a sample computation of the critical path delay of a sum output of an RCA. The function for the error at the output of an approximate RCA is computed using the critical path delay of a sum output in the form of an indicator function.

In this section, we have described an RCA design and have characterized the associated delays and errors in the presence of overclocking. Most importantly, we have defined an indicator function able to calculate error given a particular input and our assumptions regarding circuit operation.

9 Efficient Evaluation of Average Error of an Approximate RCA

In this section, we describe our approach to compute the average of the error at the output of an approximate RCA over a candidate set of inputs. We then present the technique that we follow to efficiently compute this average error. We also describe the constraints that we pose on our target problem.

Theorem 1 (Eq. 5) gives $Er(D, \mathbf{a}, \mathbf{b}, \mathbf{d})$ 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

$$Er_{\text{avg}}(D, \mathbf{d}) = \underset{0 \le \mathbf{a},\mathbf{b} \le 2^n - 1}{\text{avg}} Er(D, \mathbf{a}, \mathbf{b}, \mathbf{d}). \tag{6}$$

where $\mathbf{a}$, $\mathbf{b}$, $\mathbf{d}$ and D are defined in Section 4.

Eq. 6 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. 6 into a form that can be computed in $O(n^2)$ operations.

Recall that 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 (therefore $C_{ij} = 1$) is

$$\begin{aligned} \mathbf{Er}_{\text{cc}}(D, i, j, \mathbf{d}) &= \sum_{k=i+1}^{j} (s_k^{\text{a}} - s_k) 2^k \\ &= \sum_{k=i+1}^{j} I_k^{\text{cc}}(D, i, j, \mathbf{d}) 2^k \end{aligned} \tag{7}$$

where

$$I_k^{\text{cc}}(D, i, j, \mathbf{d}) = \begin{cases} 1 & \text{if } i < k < j \text{ and } d_{ik} > D \\ -1 & \text{if } i < k = j \text{ and } d_{ik} > D \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

From Section 8.2.5, we know that the indicator function I_k depends on D, $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{d}$. Looking at Eq. 4, we can see that in the definition of $I_k(\mathbf{a}, \mathbf{b}, \mathbf{d}, D)$ we use the variables $\mathbf{a}$ and $\mathbf{b}$ to determine whether there is a carry chain around position k. In contrast to Eq. 4, in Eq. 8, we assume that $C_{ij} = 1$ for $i < k \le j$. (note that this can be satisfied for many different input pairs). Therefore we do not need $\mathbf{a}, \mathbf{b}$ as inputs to $I_k^{\text{cc}}(D, i, j, \mathbf{d})$; we only need i, j with a carry chain from i to j.

Theorem 2.

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

Proof. Consider an n-bit addition. Let there be α carry chains in the addition, where $0 \le \alpha \le n - 1$.[2]

We know from Theorem 1 that there could be an error at the output of an addition only if there is a carry chain. Therefore, for the case when $\alpha = 0$, there is no error. If $\alpha = 0$ that means there does not exist i, j such that $C_{ij}(\mathbf{a}, \mathbf{b}) = 1$. Hence the theorem holds for $\alpha = 0$.

Consider $\alpha \neq 0$. Let the x^{th} carry chain start from position i^x and be killed at position j^x. From Theorem 1, we know that

$$Er(D, \mathbf{a}, \mathbf{b}, \mathbf{d}) = \sum_{k=0}^{n} I_k(\mathbf{a}, \mathbf{b}, \mathbf{d}, D) 2^k \tag{9}$$

If we expand the right hand side of Eq. 9, we get

$$Er(D, \mathbf{a}, \mathbf{b}, \mathbf{d}) = I_0 2^0 + I_1 2^1 + \cdots + I_{n-1} 2^{n-1} + I_n 2^n \tag{10}$$

where each I_k, for $0 \le k \le n$, is dependent on $\mathbf{a}, \mathbf{b}, \mathbf{d}$ and D.

As $I_k(\mathbf{a}, \mathbf{b}, \mathbf{d}, D) = 0$ if there is no carry chain, we can group the terms on the right hand side of Eq. 10 into groups of carry chains. This is possible since we have shown in Observation 1 that carry chains cannot overlap in the same addition.

$$\begin{aligned} Er(D, \mathbf{a}, \mathbf{b}, \mathbf{d}) = \left(I_{(i^1+1)} 2^{i^1} + \cdots + I_{j^1} 2^{j^1}\right) + \cdots + \\ \left(I_{(i^\alpha+1)} 2^{i^\alpha} + \cdots + I_{j^\alpha} 2^{j^\alpha}\right) \end{aligned} \tag{11}$$

[2] For example, if $\alpha = 0$ then that means there is no carry being propagated in the entire addition. The other extreme case would be if $\alpha = n - 1$ which would happen if $\mathbf{a} = \mathbf{b} = 2^n - 1$.

where carry chain 1 is generated at i^1 and killed at j^1 and carry chain α is generated at i^α and killed at j^α.

Observing $I_k(\mathbf{a}, \mathbf{b}, \mathbf{d}, D)$ from Eq. 4 and $I_k^{\text{cc}}(D, i, j, \mathbf{d})$ from Eq. 8, we can conclude that $I_k(\mathbf{a}, \mathbf{b}, \mathbf{d}, D) = I_k^{\text{cc}}(D, i, j, \mathbf{d})$ if $C_{ij}(\mathbf{a}, \mathbf{b}) = 1$.

Therefore from Eq. 11 and Eq. 7

$$Er(D, \mathbf{a}, \mathbf{b}, \mathbf{d}) = \left(I^{\text{cc}}_{(i^1+1)} 2^{i^1} + \cdots + I^{\text{cc}}_{j^1} 2^{j^1}\right) + \cdots + \left(I^{\text{cc}}_{(i^\alpha+1)} 2^{i^\alpha} + \cdots + I^{\text{cc}}_{j^\alpha} 2^{j^\alpha}\right) \quad (12)$$

Substituting Eq. 7 in Eq. 12 we get the following:

$$Er(D, \mathbf{a}, \mathbf{b}, \mathbf{d}) = \mathbf{Er}_{\text{cc}}(D, i^1, j^1, \mathbf{d}) + \cdots + \mathbf{Er}_{\text{cc}}(D, i^\alpha, j^\alpha, \mathbf{d}) \quad (13)$$

Hence the theorem holds.

□

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.

$$Er_{\text{avg}}(D, \mathbf{d}) = \sum_{0 \le i < j \le n} p_{ij} \mathbf{Er}_{\text{cc}}(D, i, j, \mathbf{d}), \quad (14)$$

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 $n(n+1)/2$ (see Observation 2) terms only.

Observation 2. Consider an n-bit addition with inputs **a** and **b**. The total number of different types of carry chains that can exist in an n-bit addition is $n(n+1)/2$.

Proof. Consider an n-bit addition. From the definition of a carry chain from Section 4, the number of carry chains that could start from each position are as follows.

- A carry chain that starts at position 0, could end at position 1, position 2, position 3, ..., position n. Therefore there are n carry chains that start at position 0.
- A carry chain that starts at position 1, could end at position 2, position 3, ..., position n. Therefore there are $n-1$ carry chains that start at position 1.
- ...
- A carry chain that starts at position $n-2$, could only end at position $n-1$. Therefore there is only one carry chain that starts at position $n-2$.
- A carry chain cannot start at position $n-1$.

By summing the number of carry chains starting from various positions we get,

$$\text{Total number of carry chains} = n+(n-1)+(n-2)+\cdots+1 \quad (15)$$

The right hand side of Eq. 15 is an arithmetic progression whose total is $n(n+1)/2$.

Hence the observation holds.

□

The probabilities p_{ij} can be computed given the distributions of the inputs a and b. Here we will assume a uniform distribution, that is, a_i and b_i, for all $0 \le i < n$, are each 0 or 1 with probability $\frac{1}{2}$. Based on the definition of a carry chain from Section 4, for a carry chain to be present from position i to position j the following conditions have to be satisfied when $j \neq i+1$.

- $a_i = b_i = 1$. Probability of $a_i = b_i = 1$ is $\frac{1}{2} \times \frac{1}{2} = \left(\frac{1}{2}\right)^2 = \frac{1}{4}$
- $a_w \neq b_w$. Probability of $a_w \neq b_w$ is equal to $P(a_w = 0 \text{ and } b_w = 1) + P(a_w = 1 \text{ and } b_w = 0)$. This is equal to $\left(\frac{1}{2} \times \frac{1}{2}\right) + \left(\frac{1}{2} \times \frac{1}{2}\right) = \frac{1}{2}$.
- $a_j = b_j$. Probability of $a_j = b_j$ is equal to $P(a_j = 0 \text{ and } b_j = 0) + P(a_j = 1 \text{ and } b_j = 1)$. This is equal to $\left(\frac{1}{2} \times \frac{1}{2}\right) + \left(\frac{1}{2} \times \frac{1}{2}\right) = \frac{1}{2}$.

Therefore p_{ij} is the product of probabilities of all the three conditions i.e.,

$$\begin{aligned} p_{ij} &= P(a_i = b_i = 1) \times P(a_w \neq b_w) \times P(a_j = b_j) \\ &= \left(\frac{1}{2}\right)^2 \times \left(\prod_{w=i+1}^{j-1} \frac{1}{2}\right) \times \frac{1}{2} \\ &= \left(\frac{1}{2}\right)^{j-i+2} \end{aligned}$$

where $j \neq i+1$.

The probability of a carry chain to be present from position i to position j when $j = i+1$ is

$$\begin{aligned} p_{ij} &= P(a_i = b_i = 1) \times P(a_j = b_j) \\ &= \left(\frac{1}{2}\right)^2 \times \frac{1}{2} \\ &= \left(\frac{1}{2}\right)^3 \end{aligned}$$

The above method calculates the probability that there exists a carry chain if the input distribution is uniform. 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. This would require only one pass through the entire candidate set of inputs which is an $O(n)$ operation. In this paper, we will leave the case of non-uniform input bits for future work.

In this section we discussed our approach to efficiently computing the average error of an approximate RCA. We also presented the constraints that we pose on our target problem.

10 Energy Consumption Models

In this section, we first describe an energy model to estimate energy consumption for a CMOS circuit of an RCA. We then describe our approach to extend the energy model of an RCA to estimate the energy consumption of an approximate RCA for solving our target problem.

10.1 Energy model for an RCA

In this subsection we will discuss an energy model for an RCA.

The total energy consumption in an RCA consists of two separate components, the dynamic energy consumption and the static energy consumption. The dynamic energy consumption constitutes the energy spent during the charging and discharging of capacitive loads during logic changes. The average dynamic energy consumed by a CMOS circuit thus depends on the number of logic changes which is denoted by the switching activity of the adder circuit. The switching activity of gate ℓ, denoted as w_ℓ, is the average number of logic changes that gate undergoes in a single addition. w_ℓ is approximately estimated as the ratio of the number of logic changes of gate ℓ to the total number (say $\mathcal{A}$) of additions.

To estimate the dynamic energy consumption at the gate level of a CMOS circuit of an RCA, we use the following:

$$E^{\text{dyn}} = \sum_{\ell=1}^{N} E_\ell^{\text{dyn}}(v_\ell) w_\ell \tag{16}$$

where $E_\ell^{\text{dyn}}(v_\ell)$ is the dynamic energy consumption of the ℓ^{th} gate being operated at supply voltage v_ℓ, w_ℓ is the average switching activity of the ℓ^{th} gate in a single clock cycle (assuming a non-pipelined adder) and N is the total number of gates in the RCA.

The total energy consumption also includes the static energy consumption. The static energy consumption in a CMOS circuit is due to the leakage current between different nodes in a transistor when the transistor is not switching. In general we assume that the leakage current does not change over time and hence static energy consumption of a CMOS circuit per clock cycle is estimated to be a linear function of the clock cycle time. To estimate static energy consumption we use the following model:

$$E^{\text{stat}} = \sum_{\ell=1}^{N} P_\ell^{\text{stat}}(v_\ell) D \tag{17}$$

where $P_\ell^{\text{stat}}(v_\ell)$ is the static power consumption of the ℓ^{th} gate being operated at supply voltage v_ℓ and D is the clock cycle time of the circuit.

Therefore, the total energy consumption is the sum of both the dynamic energy consumption and the static energy consumption given by the following:

$$E = \sum_{\ell=1}^{N} \left(E_\ell^{\text{dyn}}(v_\ell) w_\ell + P_\ell^{\text{stat}}(v_\ell) D \right) \tag{18}$$

10.2 Energy model for an approximate RCA

In this subsection we will present our approach to model energy consumption of an approximate RCA by extending the energy model for an RCA discussed in Section 10.1.

The total energy consumption of an RCA as described in Eq. 18 is the following:

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

For an approximate RCA, Eq. 18 may be used if we find the switching activities for the gates under overclocking. As described in Section 8.2, 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 is computed in time or not is modeled using the indicator function in Eq. 4 in Section 8.2.5. We will 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 to reflect this.

Consider an n-bit approximate RCA. Assume a carry chain starting from position i and ending at position j. Define an indicator function for computing the energy consumption as follows:

$$I_k^{\mathrm{E}}(D,i,j,\mathbf{d}) = \begin{cases} 1 & \text{if } i < k \leq j \text{ and } d_{ik} > D \\ 0 & \text{otherwise.} \end{cases} \quad (19)$$

Based on the gate level design of the ripple carry adder described in Section 8.1, we assume that in correct computation, soon after the inputs are provided to the adder all the sum bits are initially computed with all the carry bits as 0.[3] We discuss the effect of non-zero carry bits on the energy modeling in Appendix A.

As the adder is computing, all the gates might switch (probably more than once) before the correct carry inputs are computed at each position. Looking at the design of an RCA described in Section 8.1 we can conclude that once the correct carry bit arrives at the input of a full adder then the bottom XOR gate (that computes the sum output) and the the MUX (that computes the carry output) switch once again. This is because of our assumption that all the carry bits are zero to begin with. Therefore, if there is supposed to be a carry input of one at a particular full adder and because of overclocking the correct carry input is not computed within the clock cycle time, then the XOR gate and the MUX gate do not switch for the last time. That means, in a single addition assuming $C_{ij} = 1$, if $I_k^{\mathrm{E}}(D,i,j,\mathbf{d}) = 1$, then the number of toggles at the XOR gate and the MUX gate is one less than what they are supposed to be if the adder is not overclocked.

Here we only need to consider the effect of a single carry chain at a particular bit position because multiple carry chains cannot overlap as per Observation 1 in Section 8.2.2. A carry chain starting from position i to position j occurs with a probability of p_{ij}.

We also know from the definition of switching activities in Section 4 that

$$w_\ell = \frac{\text{Total number of toggles of gate } \ell}{\mathcal{A}}$$

where $1 \leq \ell \leq N$. Here we are using the indexing scheme that we defined in Section 8.1.

But if $I_k^{\mathrm{E}}(D,i,j,\mathbf{d}) = 1$ then the number of toggles at gate ℓ, where $\ell = 3k+2$ or $\ell = 3k+3$ (using the indexing scheme defined in Section 8.1), decreases by 1 for one addition. Since a carry chain generated from position i and killed at position j occurs with a probability of p_{ij}, there would be $p_{ij} \times \mathcal{A}$ additions (among $\mathcal{A}$ additions) with $C_{ij} = 1$. Since the clock cycle time and supply voltages are not being changed, we can conclude that for all of the $p_{ij} \times \mathcal{A}$ additions $I_k^{\mathrm{E}}(D,i,j,\mathbf{d})$ is a constant. Therefore the number of toggles at gate ℓ (for $\ell = 3k+2$ or $\ell = 3k+3$) is decreased by $p_{ij} \times \mathcal{A}$ when $I_k^{\mathrm{E}}(D,i,j,\mathbf{d}) = 1$. Hence, only considering the carry chain from i to j,

$$\begin{aligned} w_\ell^{\mathrm{a}} &= \frac{(\text{Total number of toggles of gate } \ell) - p_{ij}\mathcal{A}I_k^{\mathrm{E}}}{\mathcal{A}} \\ &= \frac{\text{Total number of toggles of gate } \ell}{\mathcal{A}} - \frac{p_{ij}\mathcal{A}I_k^{\mathrm{E}}}{\mathcal{A}} \\ &= w_\ell - p_{ij} I_k^{\mathrm{E}}(D,i,j,\mathbf{d}) \end{aligned}$$

where $\ell = 3k+2$ or $\ell = 3k+3$.

Taking into account all possible carry chains in an n-bit addition, the corresponding switching activities of these gates in an n-bit approximate RCA should be decreased accordingly based on the following equations.

$$w_{3k+2}^{\mathrm{a}} = w_{3k+2} - \left(\sum_{0 \leq i < j \leq n-1} p_{ij} I_k^{\mathrm{E}}(D,i,j,\mathbf{d}) \right) \quad (20)$$

$$w_{3k+3}^{\mathrm{a}} = w_{3k+3} - \left(\sum_{0 \leq i < j \leq n-1} p_{ij} I_k^{\mathrm{E}}(D,i,j,\mathbf{d}) \right) \quad (21)$$

where $0 \leq k < n$. As per the design of an RCA described in Section 8.1, each full adder has 3 gates. Therefore the number of gates is $N = 3 \times n$. As we compute the switching activity for each gate, there would be $N = 3n$ unique switching activities. The switching activities are indexed based on the indexing scheme for gates defined in Section 8.1.

Table 3: Maximum and minimum propagation delays of the XOR gate and the MUX in 90nm technology

Gate	max-delay$_\ell$ (pico-sec)	min-delay$_\ell$ (pico-sec)
XOR	33.3	55.2
MUX	30.5	51.2

Example 9. Consider the 3-bit RCA shown in Fig. 8. We will compute the approximate switching activities (w_ℓ^{a} for $1 \leq \ell \leq N$) for the gates in this adder. The first step in computing $\mathbf{W}^{\mathrm{a}}$ is to compute the switching activities ($\mathbf{W}$) for an RCA that is not being overclocked. The total number of input combinations to a 3-bit RCA is $\mathcal{A} = 2^3 \times 2^3 = 64$. Using simulations of the 3-bit RCA, we compute the average switching activity at the output of each gate when there is no overclocking, which is the ratio of the number of toggles at the output the gate to the number of additions. This gives the average switching of each gate per addition.

Now we reduce these switching activities to take into account the effect of overclocking. For the sake of this

[3] To estimate the energy consumption model for an approximate RCA we assume that all the carry bits are 0 when the inputs are provided for each addition. But in reality the carry bits typically retain values from the prior computations and are not reset to 0 every time.

example, let us consider that $C_{13}(\mathbf{a}, \mathbf{b}) = 1$. From the description of the sum path in Section 8.2.4 and the propagation delay values shown in Table 3 (repeated from Table 2 in Section 8.1), we know that $d_{01} = 97.1$ps and $d_{02} = 127.6$ps. Let the clock cycle time, D, be 120ps. This means that from Eq. 19, $I_0^{\mathsf{E}}(120\text{ps}, 1, 3, \mathbf{d}) = 0$, $I_1^{\mathsf{E}}(120\text{ps}, 1, 3, \mathbf{d}) = 0$ and $I_2^{\mathsf{E}}(120\text{ps}, 1, 3, \mathbf{d}) = 1$. Therefore from Eq. 20 and Eq. 21, the switching activities of gate 8 ($3 \times 2 + 2 = 8$), the XOR gate that computes the sum in the last full adder in Fig. 8, and gate 9 ($3 \times 2 + 3 = 9$), the MUX that computes the carry out in the last full adder, have to be reduced. Also, in this case we assume that $\mathcal{A} = 2^3 \times 2^3 = 64$.[4] Therefore the approximate switching activities of the two gates are $w_8^{\mathsf{a}} = w_8 - p_{13} I_2^{\mathsf{E}}(120\text{ps}, 1, 3, \mathbf{d})$ and $w_9^{\mathsf{a}} = w_9 - p_{13} I_2^{\mathsf{E}}(120\text{ps}, 1, 3, \mathbf{d})$, where $I_2^{\mathsf{E}}(120\text{ps}, 1, 3, \mathbf{d}) = 1$ and $p_{13} = \left(\frac{1}{2}\right)^4$ (assuming uniform input probabilities). □

Our algorithm to re-evaluate the switching activities for an approximate RCA is shown in Algorithm 1. This algorithm takes as input the size of the adder (n), the probability that a particular carry chain occurs ($\mathbf{p}$) and the switching activities ($\mathbf{W}$) of the gates in a RCA without overclocking and computes the switching activities ($\mathbf{W}^{\mathrm{a}}$) for an approximate RCA.

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

$$E^{\mathrm{a}} = \sum_{\ell=1}^{N} \left(E_\ell^{\mathrm{dyn}}(v_\ell) w_\ell^{\mathrm{a}} + P_\ell^{\mathrm{stat}}(v_\ell) D \right) \qquad (22)$$

Substituting the relationship between average dynamic energy consumption and worst-case propagation delays from Eq. 2, as described in Section 7, in Eq. 22 we get the following:

$$E^{\mathrm{a}} = \sum_{\ell=1}^{N} \left(\gamma_\ell \frac{1}{\epsilon_\ell^2(v_\ell)} w_\ell^{\mathrm{a}} + P_\ell^{\mathrm{stat}}(v_\ell) D \right) \qquad (23)$$

10.3 Summary to Section 10

In this section we have described the energy model we use for a CMOS ripple carry adder. We also presented our approach to extend the energy model to an approximate RCA using carry chains.

[4] Since it is a simple 3-bit adder our benchmark consists of all possible input cases. But in general the number of additions considered to compute the switching activities could be lower than all possible input cases if the number of input cases is prohibitively large.

Algorithm 1 Approximate Switching Activities

```
1:  procedure SWITCHING ACTIVITIY(n, p, ε(v), W) ▷ Calculates W^a, switching activities of the gates in an approximate RCA
2:      Compute d                                ▷ Discussed in Section 8.2.4
3:      for 0 < j ≤ n − 1 do
4:          for 0 ≤ i < j do
5:              for i < k ≤ j do
6:                  Compute I_k^E(D, i, j, d)    ▷ As defined in Eq 19
7:                  if I_k^E(D, i, j, d) ≠ 0 then
8:                      w^a_{3k+2} ← w_{3k+2} − p_ij I_k^E
9:                      w^a_{3k+3} ← w_{3k+3} − p_ij I_k^E
10:                     w_{3k+2} ← w^a_{3k+2}
11:                     w_{3k+3} ← w^a_{3k+3}
12:                 else
13:                     w^a_{3k+2} ← w_{3k+2}
14:                     w^a_{3k+3} ← w_{3k+3}
15:                 end if
16:             end for
17:         end for
18:     end for
19: end procedure
```

11 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. Then we present our approach to perform supply voltage binning on the solution obtained from the geometric program.

11.1 Formulation of an optimization problem

In this subsection we formulate our problem, as described in Section 3.1, of minimizing average error of an approximate RCA with a given energy budget.

We form an optimization problem consisting of an objective function and one or more constraint functions. The independent variables are called the decision variables whose values are the solution to the optimization problem.

In our case, the objective function is the average error of an approximate RCA as given in Eq. 14 in Section 9. The average error as shown in Eq. 14 is a function of D, $\mathbf{d}$ and the circuit topology. The clock cycle time D is an independent variable, but $\mathbf{d}$ (described in Section 8.2.4)

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 gate supply voltages which directly alters **d**. We found 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 $\boldsymbol{\nu}$.

Therefore we consider the propagation delays of the gates 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 this we need to compute delays $\epsilon_1(\nu_1), \epsilon_2(\nu_2), \ldots, \epsilon_N(\nu_N)$ 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. 14 which is

$$Er_{\text{avg}}(D, \mathbf{d}) = \sum_{0 \le i < j \le n-1} p_{ij} \mathbf{Er}_{\text{cc}}(D, i, j, \mathbf{d}), \quad (24)$$

subject to the following two constraints and assumptions.

1. For each gate ℓ (as per the indexing scheme defined in Section 8.1), $\ell = 1, 2, \ldots, N$

   $$\text{min-delay}_\ell \ \le \epsilon_\ell(\nu_\ell) \le \ \text{max-delay}_\ell, \quad (25)$$

   where the lower and the upper bounds depend on the transistor technology, the type of component and fanout. Please refer to Section 6 for discussion of the assumptions behind Eq. 25.

2. The total energy consumption of all the gates is bounded from above by the given energy budget. Thus,

   $$E^{\text{a}} = \sum_{\ell=1}^{N} \left(\gamma_\ell \frac{1}{\epsilon_\ell^2(\nu_\ell)} w_\ell^{\text{a}} + P_\ell^{\text{stat}}(\nu_\ell) D \right) \le \text{ Energy Budget.} \quad (26)$$

   The left hand side part of the above inequality has been obtained from Eq. 23 in Section 10. In Eq. 26, the proportionality constants (γ_ℓ) and the static power consumption values ($P_\ell^{\text{stat}}(\nu_\ell)$) are constants which depend on the process technology and transistor-level designs of the gates used in the RCA. Since we alter neither the process technology nor the transistor-level designs of the gates $P_\ell^{\text{stat}}(\nu_\ell)$ is constant as far as the optimization problem is concerned. And though γ_ℓ does vary with supply voltage, we consider it as a constant as described in Section 7. The clock cycle time D and the Energy Budget are variables which we can determine. The propagation delays of the gates, $\epsilon_\ell(\nu_\ell)$, are the variables that can change during the optimization process.

The assumptions under which the above optimization problem is framed are as follows.

(i) The gate topology of the circuit of the adder is considered as a given and is not changed during the optimization.

(ii) Potentially the optimization problem can result in a solution that allocates each gate a unique supply voltage. But as described in Section 2, with multiple supply voltages the need for voltage level converters arises. So to make the design more pragmatic we will bin the solution from the optimization problem to a fixed set of voltages. Currently we do not account for the additional energy consumption due to the voltage level converters in the optimization problem.

(iii) As discussed in Section 6, the propagation delays of the gates are considered to be worst-case non-varying (except with only supply voltage) deterministic delays.

In this subsection we formulated our target problem as an optimization problem by describing the objective function, the constraints and the assumptions.

11.2 How to minimize average RCA error using geometric programming

In this section we present our approach using a technique called geometric programming to increasing the accuracy at the output of the RCA by minimizing the function given in Eq. 24 in Section 11.1 (which is repeated from Eq. 14).

In general the full class of optimization problems could be classified into two categories, linear optimization problems (LP) and non-linear optimization problems (NLP). The objective function of our optimization problem in this paper, which is shown in Eq. 24, is not a linear function. To clarify this point, let us observe the objective function.

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

Substituting Eq. 7 we obtain the following:

$$Er_{\text{avg}}(D, \mathbf{d}) = \sum_{0 \le i < j \le n-1} p_{ij} \left(\sum_{k=i+1}^{j} I_k^{\text{cc}}(D, i, j, \mathbf{d}) 2^k \right) \quad (27)$$

In Eq. 27 it can be observed that the indicator function $I_k^{\text{cc}}(D, i, j, \mathbf{d})$ is not a linear function and in fact it is not even a continuous function.

The authors have not been able to come up with a linear function to estimate/represent Eq. 27; therefore, the authors have so far not been able to apply traditional linear programming optimization techniques to this problem. The alternative is to model our problem as a nonlinear optimization problem (NLP). Although modeling our problem as an NLP is trivial, solving a general NLP is computationally difficult. In contrast, a sub-class of NLP known as geometric programs (GP) are easy to solve, and also a global solution can be achieved efficiently. In addition, there are effective and reliable methods to solve a GP. Hence we chose to compute an approximation of our target problem as a GP.

Therefore, our solution is to formulate the problem of minimizing Eq. 24 subject to the constraints outlined in Section 11.1 as a geometric program and then solve it. To further explain our procedure we present in Definition 2 a particular type of function called a monomial and, in Definition 3 an extension of monomials known as a posynomial (short for positive polynomial) [29].

Definition 2. Let $x_1, \ldots, x_n$ denote n real positive variables, and $\mathbf{x} = (x_1, \ldots, x_n)$ a vector with components x_i. A real valued function f of $\mathbf{x}$, with the form

$$f(\mathbf{x}) = c x_1^{a_1} x_2^{a_2} \cdots x_n^{a_n},$$

where $c > 0$ and $a_i \in \mathbf{R}$, is called a monomial [29].

Definition 3. A posynomial is a sum of one or more monomial functions [29].

To model our target problem as a geometric program, the objective function and all the constraints should be in the form of a posynomial. But as can be observed from Eq. 27, 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 [29]. 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 [29].

As can be seen from Eq. 24, our objective function is not a continuous function because of the indicator functions. As per the methodology given in Section 8.2 of [29], we have to use a feasible initial guess to find a posynomial approximation of the continuous approximation of our objective function. In our case, we start with a feasible uniform voltage allocation scheme as the initial guess and compute a posynomial approximation of the objective function in Eq. 29. The following is a mathematical description of the approximations and redefinitions that we use.

To simplify the continuous approximation of the discontinuous function $Er_{\text{avg}}(D, \mathbf{d})$ (shown in Eq. 27), we will redefine an indicator function adapted from $I_k^{\text{cc}}(D, i, j, \mathbf{d})$ (given in Eq. 8), which is a part of the right hand side of Eq. 27, as follows:

$$\mathbf{I}_k(D, i, j, \mathbf{d}) = \begin{cases} 1 & \text{if } d_{ik} > D,\ i < k \leq j \\ 0 & \text{otherwise.} \end{cases} \tag{28}$$

Using this definition, $\mathbf{Er}_{\text{cc}}(D, i, j, \mathbf{d})$ which is used in Eq. 24 (previously defined in Eq. 7) is transformed as follows

$$\begin{aligned} & \mathbf{Er}_{\text{cc}}(D, i, j, \mathbf{d}) \\ = & \sum_{k=i+1}^{j} I_k^{\text{cc}}(D, i, j, \mathbf{d}) 2^k \\ = & \sum_{k=i+1}^{j-1} \mathbf{I}_k(D, i, j, \mathbf{d}) 2^k - \mathbf{I}_j(D, i, j, \mathbf{d}) 2^j \end{aligned} \tag{29}$$

where $I_k^{\text{cc}}(D, i, j, \mathbf{d})$ is shown in Eq. 8. Because the new indicator function $\mathbf{I}_k(D, i, j, \mathbf{d})$ 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. 4 and the error function in Eq. 7 in Section 9 results in the same value as the redefined indicator function in Eq. 28 and transformed error function in Eq. 29.

This redefinition allows us to make the indicator function a non-negative function so that it can be represented in terms of the signum function. We do this because a signum function can be approximated to a smooth continuous analytical function which we will then approximate to a posynomial as required for a geometric program.

We now represent $\mathbf{I}_k(D, i, j, \mathbf{d})$, by

$$\mathbf{I}_k(D, i, j, \mathbf{d}) = \frac{1 + \text{sgn}(d_{ik} - D)}{2}$$

where $\text{sgn}(x)$ is the signum function. For $\kappa \gg 0$, $\text{sgn}(x) \approx \tanh(\kappa x)$, and we use $\kappa = 200$.[5] Therefore,

$$\mathbf{I}_k(D, i, j, \mathbf{d}) \approx \frac{1}{2} + \frac{\tanh(\kappa(d_{ik} - D))}{2} = \frac{1}{1 + e^{-2\kappa(d_{ik} - D)}}.$$

Thus, our continuous and differentiable approximation

[5] 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.

of Eq. 29 is

$$\begin{aligned}
&\mathbf{Er}_{\text{cc}}(D,i,j,\mathbf{d}) \\
&\quad= \sum_{k=i+1}^{j-1} \mathbf{I}_k(D,i,j,\mathbf{d})2^k - \mathbf{I}_j(D,i,j,\mathbf{d})2^j \\
&\approx \sum_{k=i+1}^{j-1} \frac{1}{1+e^{-2\kappa(d_{ik}-D)}}2^k - \frac{1}{1+e^{-2\kappa(d_{ij}-D)}}2^j
\end{aligned} \tag{30}$$

where d_{ij} are linear functions of $\epsilon_\ell(v_\ell)$. We use the monomial approximation technique (Section 8.2 of [29]) for this expression. This results in

$$\begin{aligned}
&\mathbf{Er}_{\text{cc}}(D,i,j,\mathbf{d}) \\
&\approx \sum_{k=i+1}^{j-1} \frac{1}{1+e^{-2\kappa(d_{ik}-D)}}2^k - \frac{1}{1+e^{-2\kappa(d_{ij}-D)}}2^j \\
&\approx c\epsilon_1^{a^1}\epsilon_2^{a^2}\ldots\epsilon_N^{a^N}
\end{aligned} \tag{31}$$

where $c \in \mathbf{R}^+$ and $a^\ell \in \mathbf{R}$ for all $1 \le \ell \le N$. The expression in Eq. 31 is a monomial as per Definition 2.

We then construct the objective function $Er(D,\mathbf{d})$ as a posynomial (defined in Definition 3) from Eq. 27.

As $\boldsymbol{\epsilon}(\boldsymbol{v})$ are the decision variables, we now express $\mathbf{d}$ in terms of $\boldsymbol{\epsilon}(\boldsymbol{v})$, 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:

$$\text{Minimize } Er_{\text{avg}}(D,\boldsymbol{\epsilon}) = \sum_{j=1}^{C} c_j \epsilon_1^{a_j^1}\epsilon_2^{a_j^2}\ldots\epsilon_N^{a_j^N}$$

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

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

where C is the number of possible carry chains (see Observation 2) in a n-bit adder and N is the number of lower level components (such as gates) in the adder. In the case of a $n = 16$-bit adder, $C = \frac{n(n+1)}{2} = 156$.

We use a standard geometric programming toolbox [29, 30] 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.

Table 4: Propagation delay and supply voltage values from the geometric program and corresponding binned supply voltage values for the gates in Fig. 8

Gate Index	ϵ_ℓ (ps)	v_ℓ (volts)	Binned v_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

To clarify our approach to minimize error using geometric programming, we present an example of a 3-bit adder in Example 10.

Example 10. Consider a 3-bit RCA as shown in Fig. 8. We will compute the objective function for the 3-bit RCA. The possible carry chains are $C_{02}(\mathbf{a},\mathbf{b})$, $C_{01}(\mathbf{a},\mathbf{b})$ and $C_{12}(\mathbf{a},\mathbf{b})$. From Eq. 27,

$$\begin{aligned}
Er(D,\mathbf{d}) = p_{01}\mathbf{Er}_{\text{cc}}(D,0,1,\mathbf{d}) + p_{12}\mathbf{Er}_{\text{cc}}(D,1,2,\mathbf{d}) \\
+ p_{02}\mathbf{Er}_{\text{cc}}(D,0,2,\mathbf{d})
\end{aligned}$$

Following Eq. 30, the objective function becomes

$$\begin{aligned}
Er(D,\mathbf{d}) = p_{01}\left(-\frac{1}{1+e^{-2\kappa(d_{01}-D)}}2^1\right) \\
+ p_{12}\left(-\frac{1}{1+e^{-2\kappa(d_{12}-D)}}2^2\right) \\
+ p_{02}\left(\frac{1}{1+e^{-2\kappa(d_{01}-D)}}2^1 - \frac{1}{1+e^{-2\kappa(d_{02}-D)}}2^2\right)
\end{aligned}$$

From Section 8.2.4 and Fig. 8, we know that $d_{01} = \epsilon_1(v_1)+\epsilon_3(v_3)+\epsilon_5(v_5)$, $d_{12} = \epsilon_4(v_4)+\epsilon_6(v_6)+\epsilon_8(v_8)$ and $d_{02} = \epsilon_1(v_1)+\epsilon_3(v_3)+\epsilon_6(v_6)+\epsilon_8(v_8)$. Therefore, the objective function becomes,

$$\begin{aligned}
Er(D,\mathbf{d}) = p_{01}\left(-\frac{1}{1+e^{-2\kappa(\epsilon_1(v_1)+\epsilon_3(v_3)+\epsilon_5(v_5)-D)}}2^1\right) \\
+ p_{12}\left(-\frac{1}{1+e^{-2\kappa(\epsilon_4(v_4)+\epsilon_6(v_6)+\epsilon_8(v_8)-D)}}2^2\right) \\
+ p_{02}\left(\frac{1}{1+e^{-2\kappa(\epsilon_1(v_1)+\epsilon_3(v_3)+\epsilon_5(v_5)-D)}}2^1 - \right. \\
\left. \frac{1}{1+e^{-2\kappa(\epsilon_1(v_1)+\epsilon_3(v_3)+\epsilon_6(v_6)+\epsilon_8(v_8)-D)}}2^2\right)
\end{aligned}$$

As described in Section 11.2, we first start with an initial guess of uniform voltage allocation. For this example, say

that all the gates are provided with the highest supply voltage of 1.2V and the corresponding minimum propagation delay values are shown in Table 3 from Section 10. Let us say that the clock cycle time is $D = 120$ps. Hence if the values for the propagation delay and D are substituted then it results in $\epsilon_1(v_1) + \epsilon_3(v_3) + \epsilon_5(v_5) < D$ and $\epsilon_4(v_4) + \epsilon_6(v_6) + \epsilon_8(v_8) < D$. Therefore,

$$\frac{1}{1 + e^{-2\kappa(\epsilon_1(v_1)+\epsilon_3(v_3)+\epsilon_5(v_5)-D)}} \approx 0$$
$$\frac{1}{1 + e^{-2\kappa(\epsilon_4(v_4)+\epsilon_6(v_6)+\epsilon_8(v_8)-D)}} \approx 0$$

Thus the objective function becomes

$$Er(D, \mathbf{d}) = \left| p_{02} \left(\frac{1}{1 + e^{-2\kappa(\epsilon_1(v_1)+\epsilon_3(v_3)+\epsilon_6(v_6)+\epsilon_8(v_8)-D)}} 2^2 \right) \right|$$

because we only consider the absolute magnitude of the error. We also know from Section 9 that $p_{02} = \left(\frac{1}{2}\right)^4$.

Now we use the monomial approximation technique (Section 8.2 of [29]) for this expression. For keeping the constants at a reasonable exponent (for example, not have 2.5×10^{-49} as a constant), we use all the delay values in nanosecond units. This results in the following posynomial approximation (in this case in fact it is a monomial) objective function:

$$Er(D, \mathbf{d}) = 0.2790\epsilon_1^{0.00833}\epsilon_3^{0.00762}\epsilon_6^{0.00762}\epsilon_8^{0.00833}$$

The final allocation of delays and their corresponding supply voltage values obtained through signomial programming [29] are shown in Table 4. The gate indices used in Table 4 are based on the indexing scheme defined in Section 8.1. The Energy Budget used to obtain this solution is 1.11×10^{-13} J.

□

In this section we present our solution approach which is a geometric problem based modeling of our target minimization problem. We also discuss about the specific toolbox that we use in this paper.

11.3 Supply voltage binning

In this section we present our approach to binning the solution obtained from Section 11.2 to a specific set of voltages. We also compare the solutions of the geometric program and the binned solutions in the context of a ripple carry adder.

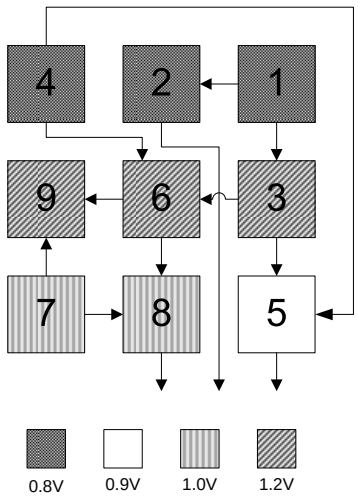


Figure 10: A sample floorplan with 4 voltage islands based on the binned solution of a 3-bit RCA shown in Table 4

The solution from Section 11.2, 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 first present our approach to limit the number of supply voltages. Let the possible set of supply voltages be $\mathcal{V}$ and the number of distinct voltages required be M_v. We then pick an M_v-combination of elements from set $\mathcal{V}$ (see definition in Section 4). 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. 27 and the relationship between propagation delay and supply voltage we can compute a closed form solution of the average error. The algorithm that we use for supply voltage binning is shown in Algorithm 2. Let $P(\mathcal{V})$ be the power-set of the set $\mathcal{V}$ and p an element of the power-set. Let m_i be the number of gates assigned with the i^{th} element of p. Without loss of generality assume that all the voltages in the sets are sorted in ascending order.

Example 11. Consider the 3-bit RCA shown in Fig. 8. Let $\mathcal{V} = \{0.8, 0.9, 1.0, 1.1, 1.2\}$ and $M_v = 4$. We will now go through Algorithm 2 to bin the supply voltages shown in Column 3 (titled v_ℓ(volts)) Table 4. The first step in the algorithm is to sort the gates in an ascending order based on the supply voltage. The order in which the gates would be arranged is $9, 6, 3, 8, 7, 5, 4, 1, 2$ as per the gates' respective indices. Now we need to pick an element from

Algorithm 2 Supply Voltage Binning

procedure BINNING($P(\mathcal{V}), M_v$)
  Sort all the gates in ascending order with respect to the supply voltage assigned by the geometric program
  for $p \in P(\mathcal{V})$ **do**
    if $|p| = M_v$ **then**
      for $\forall m_i : \sum_{i=1}^{M_v} m_i = N, m_i \geq 1$ **do**
        Assign the first m_i unassigned gates with the i^{th} element (i^{th} voltage) of p
      end for
      if $\sum_{\ell=1}^{N} \left(\gamma_\ell \frac{1}{\epsilon_\ell^2} w_\ell^{\text{a}} + P_\ell^{\text{S}}(v_\ell) D \right) \leq$ Energy Budget **then**
        Evaluate $Er_{\text{avg}}(D, \boldsymbol{\epsilon}(\boldsymbol{v}))$ from Eq. 27
      end if
    end if
  end for
  Pick the p and m_i's with the minimum $Er_{\text{avg}}(D, \boldsymbol{\epsilon}(\boldsymbol{v}))$
end procedure

$P(\mathcal{V})$ such that the element's cardinality is $M_v = 4$. The number of elements in $P(\mathcal{V})$ that satisfy this condition is $\binom{6}{4} = 15$. A few of such elements are $\{0.8, 0.9, 1.0, 1.1\}$, $\{0.8, 0.9, 1.1, 1.2\}$ and $\{0.9, 1.0, 1.1, 1.2\}$. Here each element is a set of voltages. For each such set we have to determine all possible ways of mapping the gates (from the sorted order) to the voltages. An obvious heuristic that we follow is that gates which have been assigned higher voltages by the geometric program are assigned to a bin with a higher voltage. For example, consider the set $\{0.8, 0.9, 1.0, 1.1\}$ and a few ways of binning are

- gates $9, 6, 3 \rightarrow 0.8$V, gates $8, 7 \rightarrow 0.9$V, gates $5, 4 \rightarrow 1.0$V and gates $1, 2 \rightarrow 1.1$V
- gates $9, 6, 3 \rightarrow 0.8$V, gates $8, 7, 5 \rightarrow 0.9$V, gate $4 \rightarrow 1.0$V and gates $1, 2 \rightarrow 1.1$V
- gates $9, 6, 3, 8 \rightarrow 0.8$V, gates $7, 5, 4 \rightarrow 0.9$V, gate $1 \rightarrow 1.0$V and gate $2 \rightarrow 1.1$V

From all such schemes for binning we will pick the scheme that has the least error and satisfies the energy constraint as shown in Algorithm 2. For the 3-bit RCA it turns out that one of the binning schemes that satisfies the energy constraint and has the lowest average error magnitude is as shown in Column 4 of Table 4. □

To determine the number of voltage islands we have to design the floorplan of the actual circuit. We agree that designing the floorplan and supply voltage binning concurrently could lead to a more efficient solution but we leave this problem for future work. For this paper, we first bin the solution obtained from the geometric program and then propose that the floorplan be designed accordingly to minimize the number of different voltage islands. This might result in some extended interconnects. But for this paper, we assume that the overheads due to interconnects is not very significant. Based on the supply voltage binning and acceptable overheads of multiple voltage lines, we can fix the number of voltage islands. In Fig. 10 we present an example of a floorplan with four different voltage islands for the 3-bit RCA solution shown in Table 4. Each box in the figure represents a single gate. The digit in each box corresponds to the index of the gate as indicated in Fig. 8.

12 Simulation Framework for RCA Experimentation

In this section we describe the simulation framework that we used for our experiments through which we aim to show that the voltage assignments generated by our geometric program described in Section 11.2 after voltage binning have lower error when compared to corresponding uniform voltage scaling (UVOS) or a naive biased voltage scaling assignment (BIVOS).

We use Synopsys HSPICE Version B-2008.09 with Synopsys 90nm technology to design and simulate our RCAs. We discuss our models for energy consumption and average error in Section. 8. The range of voltages with which the gates in the circuit are operated is 0.8V to 1.2V. As discussed in Section 2 we realize that using multiple voltages may necessitate voltage level converters. Currently, we do not include voltage level converters in our simulations because our voltage shifts are usually very small (in the 0.1V – 0.2V range) [7].

We simulate the circuits with different supply voltage configurations and obtain the average error magnitude and average energy consumption values. The average error magnitude for each experiment is computed by taking an average, over the number of additions performed in the experiment, of the absolute magnitude between the correct output of the approximate adder and the actual output of the approximate adder which might be different due to overclocking. The average energy consumption is measured from our HSPICE simulations by taking an average, over the total number of additions, of the total energy consumption. The total energy consumption is computed as the sum of the integrals of the product of the current drawn and the magnitude of the supply voltage over the entire period of simulation.

The input data set for the experiments is drawn from a

uniform distribution. To simulate the behavior of the adder we used 10,000 input combinations. We admit that the number of test cases is not very high, but we chose this number so that we could explore across multiple cases of voltage allocation schemes.

We designed transistor level models in HSPICE for the RCA based on the gate level description from Section 8.1. We apply the procedure described in Sec. 11.2 to obtain a globally optimized supply voltage allocation scheme for the RCA. This scheme is then binned to specific supply voltage values using the approach in Section 11.3. For our experiments, to perform supply voltage binning we used specific voltage levels which are 0.8V, 0.9V, 1.0V, 1.1V and 1.2V.

The critical path delay of an adder is computed in HSPICE by providing the adder with worst-case inputs. For an adder a worst-case input is one in which there is a carry chain from position 0 to position n−1.

We use three cases for comparing the advantage of our approach. The three cases are as follows:

- Case 1: Uniform voltage scaling (UVOS) : All the gates in the RCA are assigned the same voltage. The voltage levels can vary from 0.8V to 1.2V at the granularity of 0.01V. Therefore there are 41 different voltage allocations.

- Case 2: Naive biased voltage scaling (n-BIVOS): For comparison, we will consider the biased voltage scaling (BIVOS) approach of George et al. [2] 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 assuming a step size of 0.1V from 0.8V to 1.2V: (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.

- Case 3: Binned geometric programming solution (BGPS): The solution generated from the geometric program which is binned to limit the number of supply voltages.

The metrics that we compare are the average energy consumption and the average error magnitude. We also use the metric of relative error which is the ratio of error to the correct output, indicating how much the error affects the output. From these results we picked a few data points that illustrate the savings yielded by our methodology.

13 RCA Experimental Results

We first present the results for individual 16-bit and 32-bit ripple carry adders. Then we show the energy impact of a 16-bit globally optimized RCA in the context of an FFT.

13.1 Simulation results of 16-bit and 32-bit approximate ripple carry adders

In this section, we present the results for 16-bit and 32-bit addition using approximate RCAs.

Based on the circuit description of the RCA in subsection 8.1, the number of gates in a 16-bit RCA is 48 and in a 32-bit RCA is 96. We model these adders using the error model described in Section 8.2. The objective function for optimization, presented in Eq. 14, is computed efficiently in Section 9. The constraints for the geometric program, which are dependent on the technology, are also described in Section 9. The first constraint in Eq. 25 limits $\epsilon_k(v_k)$, the propagation delay of a single gate, between the maximum worst case delay (based on the lowest supply voltage allowed which is 0.8V for our target technology) and the minimum worst case delay (based on the highest supply voltage allowed which is 1.2V). The RCA circuit that we are using is built using two types of gates, an Exclusive OR (XOR) gate and a multiplexer (MUX). The minimum and maximum delays for these gates in our target technology, computed in HSPICE, are provided in Table 5, which is repeated from Section 8.1 for convenience. The second constraint in Eq. 26 limits the sum of energy consumption of all the gates. For our target technology, the constant γ_ℓ in Eq. 26 has been computed for the two types of gates and is shown in Table 5.

Table 5: Maximum, minimum propagation delays and proportionality constants of the XOR gate and the MUX in 90nm technology

$Gate$	min-delay (pico-sec)	max-delay (pico-sec)	γ_ℓ (Jsec^2)
XOR	33.3	55.2	2.0E-35
MUX	30.5	51.2	1.6E-35

For the experiments that we present here for the 16-bit RCA we chose 4E-10 sec (=400ps) as the clock cycle time (D). To describe the results, we chose an instance where

Table 6: 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

110 fJ is the Energy Budget in the geometric program for all the 48 gates. We present the results in Table 6 comparing the three cases described in Section 12. The solution for Case 3 (BGPS) is where 18 gates had 0.8V, 11 gates had 0.9V, 8 gates had 1.0V and 11 gates had 1.2V for supply voltages. The actual distribution of voltages across the gates in the 16-bit RCA is described in Appendix C.

Furthermore, we observed that a 16-bit RCA would have an energy consumption of 157pJ (by supplying 1.12V to all the gates) to have no overclocking errors (100% accuracy) at the same frequency of 2.5Ghz and uniform supply voltage (which is the current design methodology). When we compare this adder with no errors with the approximate adder using BGPS consuming 110fJ (as shown in Table 6) we can see that when we tolerate an error of 21.66 we reduce the energy consumption by 1.4X. These comparisons are also shown in Table 6, where "Energy Savings" refers to the reduction in energy consumption of the BGPS approximate adder with respect to the conventional correct RCA (see Section 4) operating with the same clock cycle time (D). Simulation results of a 16-bit RCA being overclocked at 2.5Ghz but for different values of Energy Budget in HSPICE are shown in Fig. 11.

The clock cycle time (D) for the 32-bit RCA has been chosen to be 6E-10 sec (=600ps). To demonstrate the results, let us choose one instance of the 32-bit approximate RCA with an Energy Budget (in the geometric program) of 132 fJ for all the 96 gates. The solution that resulted after supply voltage binning of the geometric program solution is a 32-bit RCA where 39 gates have 0.8V, 5 gates have 0.9V, 24 gates have 1.1V and 28 gates have 1.2V for supply voltages. The results for a 32-bit RCA are shown in Table 6. Also, average error magnitude versus average energy consumption results for a 32-bit approximate RCA are shown in Fig. 12.

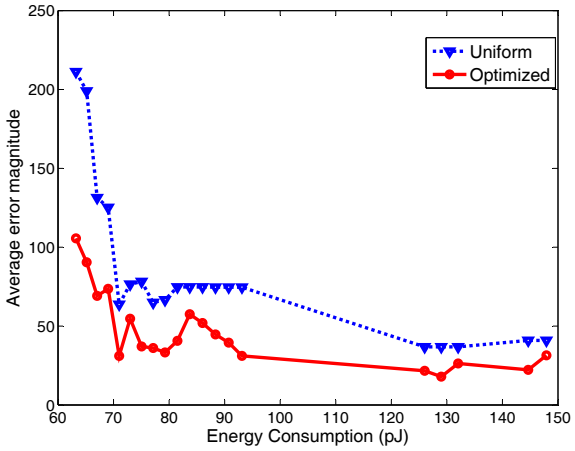


Figure 11: Average error magnitude versus average energy consumption of uniform voltage allocation and optimized voltage allocation in a 16-bit ripple carry adder

13.2 A sample scenario

Consider a popular application such as video recording. Let us pick a device such as the *Flip UltraHD* video camera (third generation) by CISCO. It records video in MPEG4 format. It has been shown that the motion estimation block is the most computationally expensive part of a MPEG4 video encoder accounting about 66% to 94% of the encoder computations [31]. Also motion estimation algorithms primarily consist of 16-bit additions.

The *Flip UltraHD* video camera comes with a battery with a supply voltage of $2.4V$ and a capacity of 2100mAh. It has been observed that this device can function only for about 2.5 hours with this battery. This means that the average power consumption (assuming a linear drain in energy from the battery) of the device will be $\frac{2100}{2.5} \times 2.4\text{mW} =$

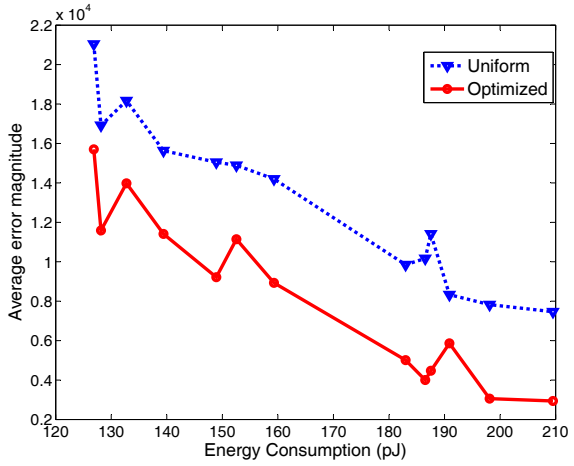


Figure 12: Average error magnitude versus average energy consumption of uniform voltage allocation and optimized voltage allocation in a 32-bit ripple carry adder

2.01W. Let us assume conservatively, that only 50% of this power is actually consumed by the electronics and the rest is consumed by other parts of the recorder such as the LCD and the camera.

The *Flip UltraHD* video camera records HD video at 720p which means that the video is being recorded at 60fps with a resolution of 1280×720.

From Table 6 we can see that by using an approximate 16-bit RCA we can save at least 1.4X energy per each addition when compared to a conventional correct RCA. This means that the power consumption of the motion estimation block can be reduced by 1.4X by using approximate adders. From the average power consumption of the entire device, if we account for the motion estimation to be approximately 90% of the total computational load, then the power consumption of the motion estimation block alone is around 0.9W. If we reduce this by 1.4X we get 0.65W. This results in a total power consumption of 1.75W.

With an average power consumption of 1.75W when compared to 2.01W the *Flip UltraHD* video camera using the same battery can function for up to 2.9 hours, which is around half an hour higher than the original battery life, for a slight decrease in accuracy of the video being recorded.

Of course a BGPS solution can be computed for a "required" battery life by setting the Energy Budget appropriately.

13.3 Assumptions regarding proportionality constant

Consider Table 7 repeated from Section 7 for convenience. γ_ℓ, as defined in Section 7, slightly increases with an increase in v_ℓ. The incremental increase can be computed to be approximately 6% per 0.1V. Therefore, for simplicity, we will assume that this does not affect the final result significantly and is captured by the mean (average) of all the voltage values.

Table 7: Propagation delay and average dynamic energy per transition of an XOR gate in 90nm process technology for various supply voltage values

v_ℓ	$E_\ell^{\text{dyn}}(v_\ell)$ (femto-J)	$\epsilon_\ell(v_\ell)$ (pico-sec)	$E_\ell^{\text{dyn}}(v_\ell) \times \epsilon_\ell^2(v_\ell)$ (10^{-35}Jsec^2)
0.8	8.63	46.89	1.90
0.9	11.18	41.61	1.94
1	14.30	37.93	2.06
1.1	17.71	35.26	2.20
1.2	21.65	33.33	2.41

By considering only the mean of γ_ℓ, the average deviation of the proportionality constant (with respect to the mean) over various supply voltages is approximately equal to only 7%. Whereas if we consider either the the maximum or minimum γ_ℓ over the various supply voltage values, the average deviation from the assumed value is as high as 15.6%.

To be sure that the assumption of a constant (over various supply voltage values for a single type of gate) proportionality constant does not affect the final solution of the binned geometric program we simulated the following cases using the binned geometric programming.

Consider a 16-bit RCA. The three cases are as follows:

- The geometric program formulation shown in Section 11.2 where γ_ℓ is the *maximum* of the proportionality constants in Table 7 which is 2.41×10^{-35}Jsec^2
- The geometric program formulation shown in Section 11.2 where γ_ℓ is the *minimum* of the proportionality constants in Table 7 which is 1.9×10^{-35}Jsec^2
- The geometric program formulation shown in Section 11.2 where γ_ℓ is the *average* of the proportionality constants in Table 7 which is 2.1×10^{-35}Jsec^2

The binned solutions of these three cases were compared. It turned out that the binned solutions were not affected at all by the changes in the proportionality constants. Of course, there were minor deviations in the results of the geometric program before binning. Therefore, in this paper, we directly use the *average* of the proportionality constants over several supply voltages and leave a more accurate analysis of proportionality constants for future work.

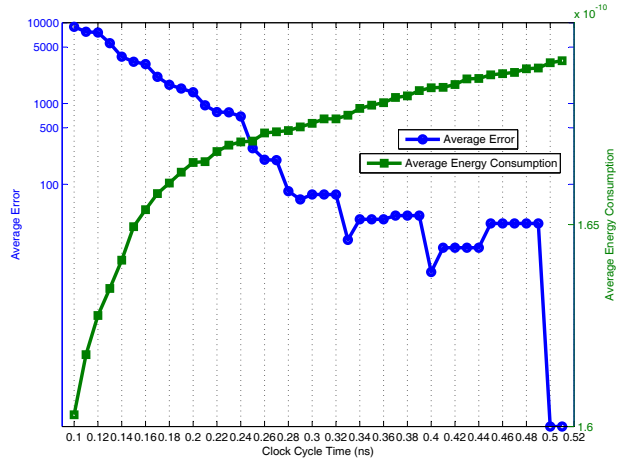


Figure 13: Average error and average energy consumption versus clock cycle time for a 16-bit approximate RCA with constant and uniform supply voltage of 1.2V

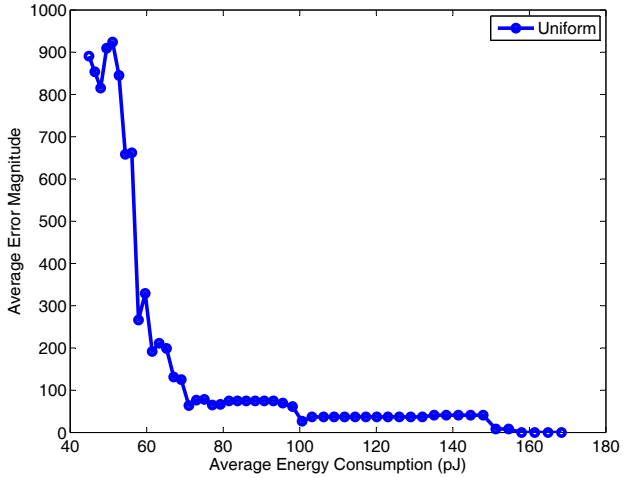


Figure 14: Average error versus energy consumption for a 16-bit approximate RCA with constant clock cycle time of 4E-10 sec

13.4 Non-monotonic error rates

In this section, we will present some results to demonstrate the non-monotonicity of the average error magnitude with respect to the average energy consumption and clock cycle time of an approximate adder.

Examining Fig. 11 and 12 we see that the typical behavior is that the accuracy of the adder typically increases with increase in energy consumption. But there are several deviations in this behavior. Therefore accuracy does not monotonically increase with increasing energy consumption. This happens because the average error magnitude of an approximate adder does not always decrease with increasing the supply voltages of the gates in the circuit of the adder. This phenomena has been explained in Section 8.2.3.

To demonstrate the non-monotonicity of the average error at the output of an approximate 16-bit RCA the behavior of average error versus clock cycle time is shown in Fig. 13. The results in in Fig. 13 were generated by keeping a constant and uniform supply voltage of 1.2V for the entire approximate RCA.

Also, as another example, we present the behavior of average error versus energy consumption for an approximate 16-bit RCA in Fig. 14 while keeping the clock cycle time constant at 4E-10 sec. This was computed by simulating a 16-bit approximate RCA with different supply voltages starting from 0.8V till 1.2V with a granularity if 0.01V. As we change the supply voltage to the adder we are changing the delays of the gates. By analyzing the data in Fig. 14 we can conclude that an increase in the delays of the gates did not result in a decrease in average error magnitude at least 98% of the time.

13.5 Approximate RCA FFT example

In this section we first describe the architecture of the 8-point FFT that we use for these experiments. Then we present simulation results of an 8-point FFT designed using conventional adders, uniform voltage scaled (UVOS) adders, naive-biased voltage scaled (BIVOS) adders and BGPS adders.

13.5.1 Architecture of an 8-point FFT

In this subsection we will present the definition of a discrete Fourier transform and the architecture of the 8-point FFT that we use.

The fast Fourier transform (FFT) is a discrete Fourier transform (DFT) algorithm which reduces the number of computations needed for P points from $2P^2$ to $2P \lg P$. The DFT can be represented as follows

$$X[k] = \begin{cases} \sum_{n=0}^{N-1} x[n] W_N^{kn} & \text{if } 0 \le k \le N-1 \\ 0 & \text{otherwise.} \end{cases}$$

where x is the input which is a finite duration sequence (of length N) of complex numbers, X is the DFT of x, and $W_N = e^{\frac{-2\pi i}{N}}$.

We use a complete decimation-in-time decomposition [32] of an 8-point FFT. A flow graph of this 8-point FFT is shown in Fig. 15. We use techniques [33] to transform multiplications in the 8-point FFT to additions of shifted inputs. We do this transform because in this paper we target only approximate adders and do not optimize

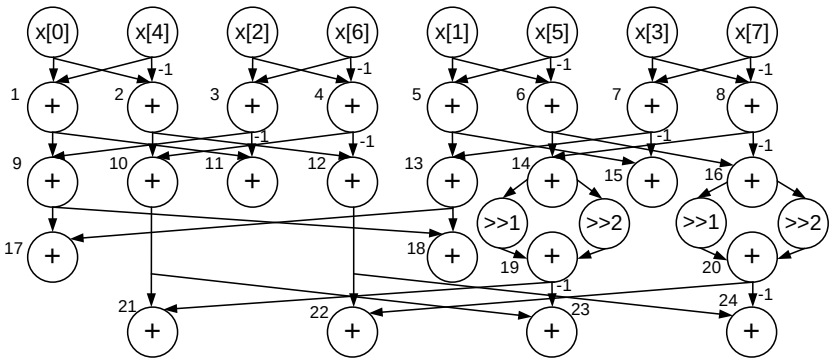


Figure 16: Graph-theoretic representation of an 8-point FFT

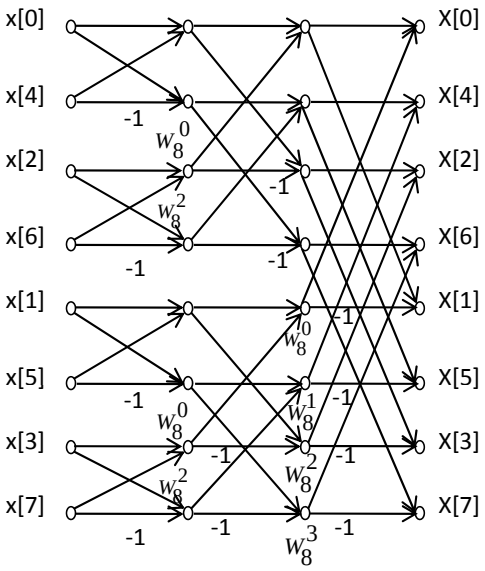


Figure 15: Flow graph of a complete decimation-in-time decomposition of a 8-point FFT

approximate multipliers. Also, it has been shown by Zhou et al. [34] that multiplier-less FFTs have lower number of computations than the FFTs with both multipliers and adders. The transformed 8-point FFT with 24 fixed point real 16-bit ripple carry adders is shown in Fig. 16.

We described the mathematical formulation of the DFT. We also discussed about the design of the 8-point FFT and the transformation we used to convert the multipliers into adders.

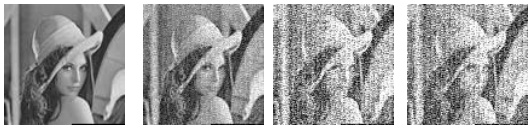
13.5.2 Experimental results of an approximate 8-point FFT

The approximate FFT was simulated in MATLAB by artificially introducing errors at the outputs of adders in the FFT based on the average error values from the simulations described in Section 13.1. As input to the FFT, we used the image shown in Fig. 17(a) with a resolution of 100X100 pixels.

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 12 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 different cases that we compare are

- Case 1: Conventional design where all the adders are being operated at 100% accuracy.
- Case 2: BGPS: The voltage allocation scheme generated by mathematical formulation and the optimization scheme presented in this paper.
- Case 3: n-BIVOS: The voltage allocation scheme where the adder is divided into 4 equal bins with voltages assigned from the set 0.8, 0.9, 1.0, 1.1, 1.2V.
- Case 4: UVOS: Approximate adders in the circuit have



(a) (b) (c) (d)

Figure 17: Images generated by (a) Case 1 (b) Case 2 (c) Case 3 and (d) Case 4

uniform voltage allocation but still being operated at the same frequency.

The resultant images from the four cases that are discussed above are shown in Fig. 17. The image generated by the FFT using BGPS adders has a PSNR that is 15db lower than the image generated by the FFT using UVOS adders with a similar energy consumption. Also the image generated by the n-BIVOS adders is 8.5 dB lower than the image generated by the BGPS adders. All these adders have a similar energy consumption.

14 Impact on Circuit Design

The novel methodology provides the circuit designer an efficient way to decide on the optimal design parameters in approximate adders. As mentioned in Section 1, there has been no method for allocating supply voltages to approximate arithmetic circuits. This means that the designer has had to exhaustively explore across all possible voltage allocation schemes, which is not practical for other than modestly sized circuits.

Our design automation solution, through geometric programming, very quickly gives a *quantitative* comparison of the relative importance of the components in an adder and assign supply voltages that can be selected by the designer.

Thus we enable the improvement of the output quality of low power approximate arithmetic adders by intelligently allocating voltages and thus make an attempt to tackle the issues of reliability and process variations.

15 Conclusions and Future Directions

Our primary contribution is a tractable solution methodology to automatically assign supply voltages, while obeying design constraints, to components in approximate ripple carry adders, so that the average error of the target adder is minimized. The work is applicable to any current RCA design of any size.

To do the optimization we use the *expected* behavior of the average error weighted with bit significance and then formulate our target problem as a geometric program. We also show a method to intelligently bin the solution from the geometric program given a list of available supply voltages from the circuit designer.

As almost all common circuits consist of mostly adders and multipliers, extending this work to other arithmetic primitives such as multipliers would be very useful. Also, clearly understanding the non-monotonic behavior of the average error versus energy consumption in an adder would be important for automating such designs.

In this paper, we do not change the actual circuit topology of the adder, but our methods could assist a designer attempting to optimize the circuit topology of an adder circuit specifically targeting approximate computation.

References

[1] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De, "Parameter variations and impact on circuits and microarchitecture," in *Proc. of the 40th Annual Design Automation Conf.*, 2003, pp. 338–342.

[2] 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.

[3] 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.

[4] K. Palem, L. Chakrapani, Z. Kedem, A. Lingamneni, and K. Muntimadugu, "Sustaining Moore's law in embedded computing through probabilistic and approximate design: Retrospects and prospects," in *Proc. of the IEEE/ACM Intl. Conf. on Compilers, Architecture, and Synthesis for Embedded Systems*, 2009, pp. 1–10.

[5] 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 *Proceedings of the 2009 Asia and South Pacific Design Automation Conference*, ser. ASP-DAC '09. Piscataway, NJ, USA: IEEE Press, 2009, pp. 535–539. [Online]. Available: http://portal.acm.org/citation.cfm?id=1509633.1509759

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

[7] 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.

[8] B. Sahu and G. Rincon-Mora, "A low voltage, dynamic, noninverting, synchronous buck-boost converter for portable applications," *Power Electronics, IEEE Transactions on*, vol. 19, no. 2, pp. 443 – 452, 2004.

[9] N. Pippenger, "Analysis of carry propagation in addition: An elementary approach," University of British Columbia, Vancouver, BC, Canada, Tech. Rep., 2001.

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Second Edition*. The MIT Press, September 2001.

[11] 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.

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

[13] 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.

[14] 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.

[15] 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.

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

[17] 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.

[18] 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.

[19] J. Tschanz, K. Bowman, S.-L. Lu, P. Aseron, M. Khellah, A. Raychowdhury, B. Geuskens, C. Tokunaga, C. Wilkerson, T. Karnik, and V. De, "A 45nm resilient and adaptive microprocessor core for dynamic variation tolerance," in *Proc. of IEEE Intl. Solid-State Circuits Conf.*, 2010, pp. 282–283.

[20] B. Shim, S. R. Sridhara, and N. R. Shanbhag, "Reliable low-power digital signal processing via reduced precision redundancy," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 5, pp. 497–510, 2004.

[21] R. Hegde and N. R. Shanbhag, "Soft digital signal processing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 9, no. 6, pp. 813–823, Dec. 2001.

[22] N. Banerjee, G. Karakonstantis, and K. Roy, "Process variation tolerant low power dct architecture," in *Proceedings of the conference on Design, automation and test in Europe*, ser. DATE '07. San Jose, CA, USA: EDA Consortium, 2007, pp. 630–635. [Online]. Available: http://portal.acm.org/citation.cfm?id=1266366.1266499

[23] 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 *Proceedings of the IFIP International Conference on Very Large Scale Integration (VLSI-SoC)*, 2005, pp. 452–457.

[24] K. V. Palem, "Energy aware algorithm design via probabilistic computing: from algorithms and models to Moore's law and novel (semiconductor) devices," in *Proceedings of the IEEE/ACM International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, 2003, pp. 113–117.

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

[26] ——, "Proof as experiment: Probabilistic algorithms from a thermodynamic perspective," in *Proceedings of the International Symposium on Verification (Theory and Practice),*, June 2003.

[27] M. Ditzel, W. Serdijn, and R. Otten, *Power-aware architecting for data-dominated applications*. Springer Verlag, 2007.

[28] A. Matsuzawa, "Low-voltage and low-power circuit design for mixed analog/digital systems in portable equipment," *Solid-State Circuits, IEEE Journal of*, vol. 29, no. 4, pp. 470–480, 2002.

[29] 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.

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

[31] P. Kuhn, *Algorithms, complexity analysis, and VLSI architectures for MPEG-4 motion estimation*. Kluwer Academic Publishers, 1999.

[32] J. Cooley and J. Tukey, "An algorithm for the machine calculation of complex fourier series," *Mathematics of Computation*, vol. 19, no. 90, pp. 297–301, 1965.

Bit position	5	4	3	2	1	0			
Intermediate carry bits at Time =0		0	0	1	0	0			
Input A		0	1	1	1	1	(15)		
Input B		0	1	0	0	1	(9)		
Time								$s^{(t)}$	Error
0	0	0	0	0	0	0		$s^{(0)} = 0$	24
1	0	0	0	0	1	0		$s^{(1)} = 6$	18
2	0	1	1	1	0	0		$s^{(2)} = 28$	-4
3	0	1	1	0	0	0		$s^{(3)} = 16$	8
4	0	1	1	0	0	0		$s^{(4)} = 24$	0

Figure 18: An example of a 5-bit binary addition using an RCA when the intermediate carry bits are assumed to non-zero at the beginning of the addition.

[33] N. Boullis and A. Tisserand, "Some optimizations of hardware multiplication by constant matrices," *IEEE Transactions on Computers*, vol. 54, pp. 1271–1282, 2005.

[34] Y. Zhou, J. M. Noras, and S. J. Shepherd, "Novel design of multiplier-less fft processors," *Signal Process.*, vol. 87, no. 6, pp. 1402–1407, 2007.

A Effect of Non-zero Carry Bits on Error and Energy Models

In this section we discuss the effect of non-zero carry bits on the error and energy models presented in this paper.

Consider the indicator function given in Eq. 4 in Section 8.2.5.

$$I_k(\mathbf{a},\mathbf{b},\mathbf{d},D) = \begin{cases} 1 & \text{if } \exists\ i,j \text{ such that } C_{ij}(\mathbf{a},\mathbf{b}) = 1, \\ & i < k < j \text{ and } d_{ik} > D \\ -1 & \text{if } \exists\ i,j \text{ such that } C_{ij}(\mathbf{a},\mathbf{b}) = 1, \\ & i < k = j \text{ and } d_{ik} > D \\ 0 & \text{otherwise.} \end{cases} \tag{32}$$

Eq. 32 is useful in modeling the error at the output of an adder because it denotes whether there is a possibility of error at the sum output of a given bit position provided the adder topology, worst-case propagation delays of the gates and the clock cycle time.

As per Theorem 1, this is based on the fact that when $C_{ij} = 1$, then unless $d_{ik} \leq D$ the sum output at bit position k is not computed correctly. If observed carefully, this is assuming that a carry cannot begin in the middle of a carry chain which could happen if the intermediate carry bits were not all zero. For example, if an intermediate carry bit at a particular bit position is 1 before the addition begins then a "rogue" carry chain could be generated. This is illustrated in Example 12.

Example 12. Consider the 5-bit addition of the two numbers 01001 and 01111. This is a duplication of Example 5 but instead of assuming that all the intermediate carry bits are 0, we will assume the intermediate carry bits as shown in Fig. 18. As we can see from the figure, due to the fact that the intermediate carry at bit position 2 is 1 before the addition started there is a "rogue" carry chain that begins from bit position 2 which, in this case, produces the correct sum output at bit position 3 earlier (shown in Example 5) than it would have been computed without the rogue carry chain. □

To model the effect of "rogue" carry chains using the procedure of indicator functions is very difficult. Since now we have to take into account the probability that an intermediate carry bit is 1 and a rogue carry chain is generated. For example, consider that $C_{ij} = 1$, then to evaluate whether there is a possibility of error at bit position k, where $i < k \leq j$, then we have to take into account all the cases where a rogue carry chain could have begun from the between bit position i and bit position k.

We also need to model the effect of multiple rogue carry chains. Our optimization of evaluating the average error at the output of the adder by computing an average over all possible carry chains is valid because of the fact that carry chains do not overlap (as shown in Observation 1. But as explained above, "rogue" carry chains can overlap and hence our error model would not hold.

Similarly our energy model is based on a similar indicator function which again would not be valid in the presence of these "rogue" carry chains.

Thus a major modification of error and energy modeling would be needed to take the effect of non-zero intermediate carry bits into account. Modeling this effect would probably result in a closer estimate to the reality but will significantly increase the complexity of the algorithm.

B Transistor Level Diagrams of the Gates used in an RCA

The design of an RCA that we use in this paper is described in Section 8.1. In this section we will present the transistor level designs of the two gates, Exclusive-OR and Multiplexer, that we use in an RCA.

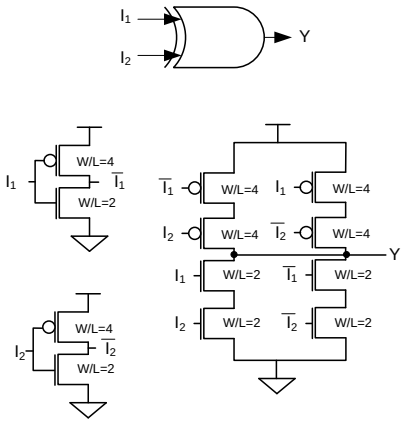


Figure 19: The transistor level diagram of the XOR gate in the RCA

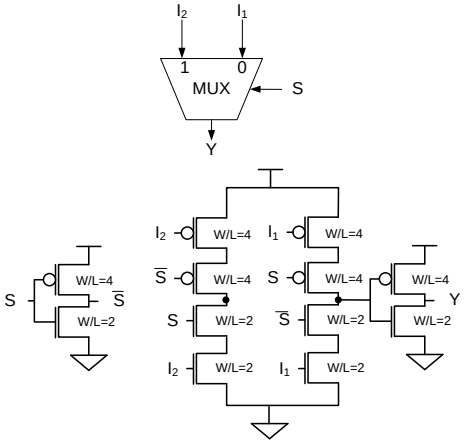


Figure 20: The transistor level diagram of the MUX gate in the RCA

The transistor level design of an XOR gate is shown in Fig. 19 and the corresponding design for a MUX gate is presented in Fig. 20.

C Distribution of Voltages Across Gates in a BGPS Approximate RCA

In this section we present an example of a 16-bit approximate RCA with a binned geometric programming solution to illustrate the distribution of binned supply voltages across the gates in the RCA. Consider Fig. 21 where the series of blocks indicate the 16-bit RCA where each block represents a 1-bit FA and the digit inside each block denotes the corresponding bit position.

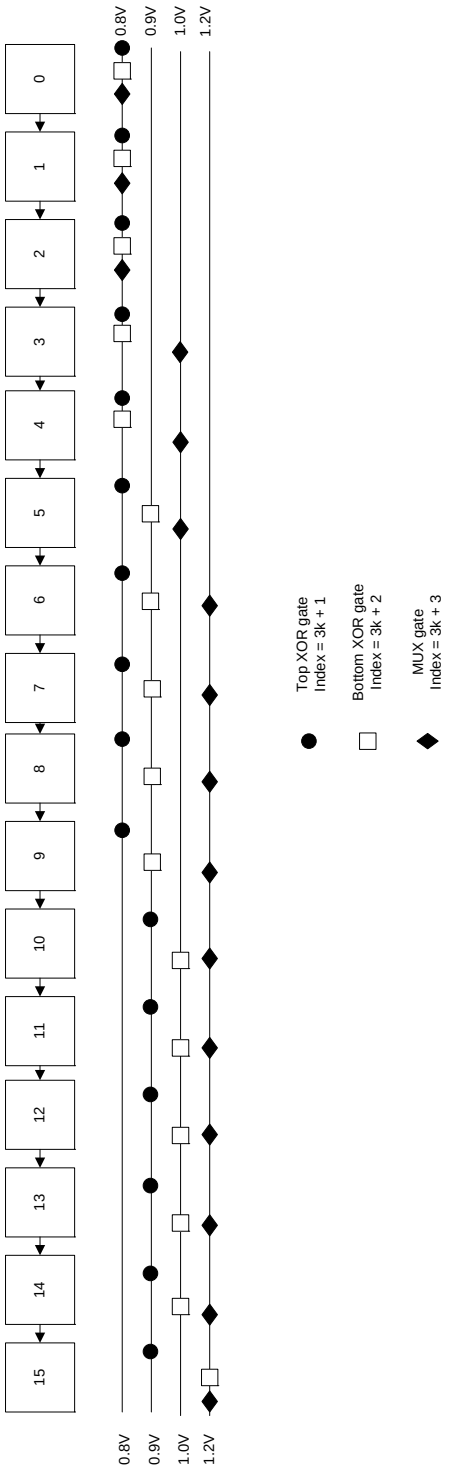


Figure 21: The transistor level diagram of the MUX gate in the RCA

The four horizontal lines below the series of blocks represent the four different supply voltage bins. In this exam-

ple they are 0.8V, 0.9V, 1.0V and 1.2V. Under each block (or 1-bit FA) there are three symbols which are placed in one of the horizontal lines. The three symbols and their meanings are as follows.

- The solid black circle represents the top XOR gate that computes the XOR of the two inputs to a 1-bit FA and in terms of our gate indexing scheme, from Section 8.1, its index is $3k + 1$ where k represents the bit position (in this case the digit inside the corresponding block and $0 \leq k \leq 15$).
- The hollow square represents the bottom XOR gate that computes the sum output in each 1-bit FA whose index is $3k + 2$.
- The solid black rhombus represents the MUX gate in each 1-bit FA that computes the next carry output and whose index is $3k + 3$.

The placement of a symbol with respect to one of the horizontal lines represents the corresponding supply voltage of that gate. For example, consider the block with the digit '10' inside. The solid circle corresponding to this block is placed on the second line which corresponds to 0.9V. This means that the top XOR gate in the 1-bit FA at bit position 10 has been assigned a supply voltage of 0.9V. The hollow square corresponding to the same block is placed on the third horizontal line which corresponds to 1.0V which means that the bottom XOR gate that computes the sum bit s^{a}_{10} has a supply voltage of 1.0V. And finally the solid black rhombus for this block is placed on the last line corresponding to the highest supply voltage which is 1.2V and hence the MUX gate at bit position 10 is supplied with 1.2V.

The reason for such a pictorial representation is to give the reader an understanding of the relative distribution of voltages across the RCA. Also there could be interesting observations that could be made from Fig. 21 such as the critical path of the RCA (in this case the MUX gates) are usually assigned as high as a voltage possible. Also intuitively since the higher order bits cause higher error in the final output gates computing the higher order bits are supplied with a higher voltage when compared to the gates computing the lower order bits.