Evaluation of Design Trade-offs for Adders in Approximate Datapath

Kan Shi and George A. Constantinides
Department of Electrical and Electronic Engineering
Imperial College London
London, UK
Email: {k.shi11, g.contantinides}@imperial.ac.uk

***Abstract*—Releasing the stringent accuracy requirement would potentially offer greater freedom to create a design with better performance or energy efficiency. In this paper, we evaluate the design trade-offs for adders, which are key building blocks for many applications. We demonstrate the optimum design metric for adders, under the consideration of various design constraints, such as accuracy, operating frequency, silicon area, different types of adder structure and alternative form of computer arithmetic for adder implementation. Two design scenarios are compared: one is the conventional design scenario where timing closure is met by adjusting operand word-lengths. The other is an overclocking scenario where timing violations are allowed to occur. We show that applying the overclocking approach to ripple carry adders can be more beneficial than using fast adders to achieve similar latency, because the worst cases only happen with very small probabilities. We also show that using the conventional approach on adders with new form of computer arithmetic is optimal for a wide range of design constraints.**

Keywords*—*approximate computing; adder; overclocking; online arithmetic;

I. Introduction

Circuit performance has increased tremendously over the past decades. However, in recent years people have observed an end of performance scaling [1] due to the improvement of CMOS technology to the nanometer regime. In order to boost circuit performance, two standard design techniques are commonly adopted: one is to heavily pipeline the datapath, the other is to reduce the design precision. For the first technique, pipelining can be used to increase operating frequency, whereas the overall computational latency will not be affected. Therefore the utilization of this technique is limited in many embedded applications, which are often designed with strict latency specifications. The second technique is common especially with FPGA technology, which embodies the flexibility to implement customized variable representation and is able to optimize the operand word-length across the entire datapath to meet a given performance and area constraint [2]. However, the precision loss will inevitably introduce quantization errors into the design.

Unfortunately, in both situations circuits are designed for the worst cases. In order to avoid potential timing violations, timing analysis tools tend to provide a conservative critical path delay by adding guard bands and timing margins. However, continuing with this approach will become increasingly expensive and difficult as technology shrinks.

To tackle this problem, a large volume of research activities were focused on the design methodology of relaxing the 100% accuracy requirements and design constraints, while reasoning about the corresponding benefits and costs. Studies have shown that computing approximately provides extra opportunities to design circuits with better performance and less energy consumption. A brief review of the work in this area is given in Section II.

In this paper, we provide detailed evaluations of design trade-offs for adders, which are key arithmetic primitives for a broad range of applications. In order to provide the optimum design choice for adders, we examine several design constrains and options, such as accuracy, operating frequency, silicon area, different types of adder structure, as well as different forms of computer arithmetic for adder implementation. In comparison to the conventional design scenario that timing closure is achieved by truncating the operand word-length, we take a radical step forward by adopting an alternative design scenario, which is named "overclocking scenario". In this case, circuits are operated far beyond the conservation region with full precision, while timing violation can be tolerated. Previous work has shown that for many applications, it is preferable to follow the overclocking scenario because the worst case is triggered only by certain input patterns and hence it happens rarely [3]. In addition to the adder structures that are created based on standard binary arithmetic, such as ripple carry adder and carry select adder, we also evaluate adder with "online arithmetic" [4], which employs different style of data representation to eliminate carry propagation. We demonstrate experimentally on FPGAs that for limited area budgets, applying the overclocking approach to ripple carry adders can be more beneficial than using fast adders to achieve similar latency. With more relaxed area constraints, the online adder outperforms in terms of accuracy or performance. Potentially our study will provide circuit designers with guidance and extra options when balancing between various design trade-offs.

In summary, the main contributions of this paper are:

1) Detailed evaluation of three adder structures: ripple carry adder, carry select adder and online adder;
2) Comparison of design scenarios and demonstration of the optimum design choice under the consideration of trade-offs between accuracy, performance and area.

The rest of this paper is organized as follows: we first

provide a brief overview of the relevant work in this area, together with the background knowledge of online arithmetic in Section II. The comparison between different adder structures and design trade-offs are detailed in Section III. This is followed by the evaluation of optimum design choices when considering trade-offs between accuracy, performance and area in Section IV. The conclusion and suggestions of possible future work is given in Section V.

II. Background

A. Approximate Datapath Design

The International Technology Roadmap for Semiconductors (ITRS) has identified that additional benefits of manufacturing, test, power and timing can be achieved by releasing the requirement of absolute correctness [5]. This fact has inspired a series of studies to create circuits operating beyond the worst cases and the safety margins. Typically this can be achieved by either reducing the supply voltage [6] or increasing the operating frequency beyond the rated value [3], [7]. The Razor project [8] served as one of the exemplary designs, in which both supply voltage and clock frequency were scaled over the most conservative value, while the error rate was monitored by using a self-checking circuit. It was shown that over 30% power saving can be obtained at the cost of around 1% errors.

Another stream of research was focusing on proposing either approximate circuit structures or design tools to trade accuracy for performance and/or energy benefits. Work in this area was inspired by the fact that many applications, e.g. image processing and machine learning, were naturally error resilient. For example, code analysis tools [9], [10] were developed to divide a program into precise parts and approximate parts, which can then be mapped to hardware components with different speed-grades and supply voltages. As for the approximate hardware, numerous forms of architectures were designed for basic arithmetic operators such as adders [11] and multipliers [12]. For example, a under-designed multiplier unit was proposed, of which the worst case was replaced by a normal case based on straightforward Karnaugh-Map analysis [13]. However, many of these techniques are designed at gate-level or transistor-level, and they cannot be directly applied onto existing hardware platforms such as FPGAs.

In this paper, we overcome the previous limitations by comparing two standard adder structures: ripple carry adder and carry select adder, which are widely employed in mainstream hardware platforms. We also compare a type of adder with "online arithmetic", which can be also found when designing fast multipliers and embedded processors [14]. It is argued that our approach can be efficiently applied to existing designs, and we back up this hypothesis based on FPGA experiments.

B. Online Arithmetic

As an alternative form of computer arithmetic, online arithmetic has been used in numerous applications such as signal processing and control algorithms [15], [16]. Online arithmetic was originally designed for digit-serial operation, as illustrated in Fig 1. It can be seen that in order to generate the first output digit, δ digits of inputs are required, where δ is called the "online delay". δ is normally a small constant, which is independent of the precision. For ease of discussion,

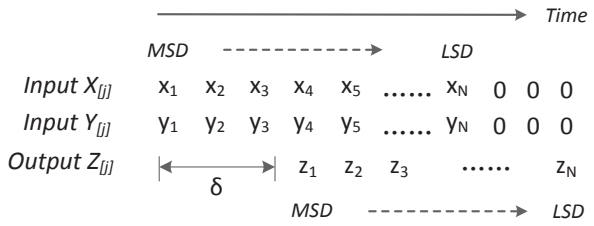


Fig. 1. Dataflow in digit-serial online arithmetic, in which both inputs and outputs are processed from the MSD to the LSD. δ denotes the online delay.

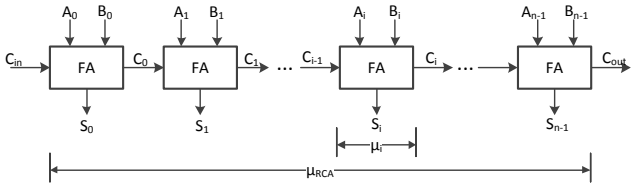


Fig. 2. An n-bit ripple carry adder. μ_i and μ_{RCA} denote the propagation delay of a full adder and the overall delay of RCA, respectively.

for the rest of this paper, the input data is assumed to be fixed point numbers in the range $(-1, 1)$. Based on this premise, the online representation of N-digit operands and result at iteration j are given by (1), where $j \in [-\delta, N-1]$ and r denotes the radix [17].

$$X_{[j]} = \sum_{i=1}^{j+\delta} x_i r^{-i},\ Y_{[j]} = \sum_{i=1}^{j+\delta} y_i r^{-i},\ Z_{[j]} = \sum_{i=1}^{j} z_i r^{-i} \quad (1)$$

MSD-first operation is possible only if a redundant number system is used. Normally there are two most commonly used redundant number representations: carry-save (CS) [18] and signed-digit (SD) [19]. With SD representation, each digit is represented using a redundant digit set $\{-a, \cdots, -1, 0, 1, \cdots, a\}$, where $a \in [r/2, r-1]$. In comparison, the standard non-redundant representation only uses a digit set $\{0, \cdots, r-1\}$. Thus a standard number corresponds to several possible redundant representations. For example, the binary number 0.011 can be represented in SD form as $0.1\bar{1}1$, $0.10\bar{1}$ or 0.011 among many other possible representations.

Due to the redundancy, the MSDs of the result can be calculated using partial information from both inputs. Then the value of the number can be revised using the subsequent digits, because each number has multiple representations.

III. Design Trade-offs of Different Adder Structures

A. Ripple Carry Adder

Adders serve as a key building block for arithmetic operations. In general, the ripple carry adder (RCA) is the most straightforward and widely used adder structure. As such, in our previous work we proposed probabilistic models of overclocking errors for RCA [3]. A brief summary of the models is described below.

For an n-bit RCA, it is composed of n serial-connected full adders (FA) as shown in Fig. 2. Typically the maximum frequency of RCA is determined by the longest carry propagation. Under the assumption that the carry propagation delay of each

FA is a constant value $\mu_i = \mu$, the critical path of the RCA is: $\mu_{RCA} = n\mu$. For the sampling period T_S, it follows that if $T_S \geqslant \mu_{RCA}$, correct result will always be sampled. Otherwise, intermediate result will be sampled and potentially generating overclocking errors. The mean value of overclocking error is modeled [3] as given by (2), where coefficient b is given in (3) and it determines the maximum length of error-free carry propagation.

$$E_{oc} = \begin{cases} 2^{-b} - 2^{-n-1}, & \text{if } b \leq n \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

$$b := \left\lceil \frac{T_S}{\mu} \right\rceil = \left\lceil \frac{1}{\mu \cdot f_S} \right\rceil \tag{3}$$

In the conventional scenario, a specific timing target can be met by truncating the word-length of RCA while timing violation is not permitted. This will result in truncation error for most data. The mean value of truncation error is also modeled [3] as given by (4), where parameters k and n denote the word-length of RCA before and after truncation, respectively.

$$E_{trad} = \begin{cases} 2^{-n} - 2^{-k}, & \text{if } n < k \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

In this paper we consider two design scenarios for RCA: one is the overclocking scenario where timing violation is allowed to happen while maintaining the original word-length; the other is the traditional scenario by truncating RCA word-length to meet timing. For a given f_S, in the first design scenario we have $n = k$. In the second design scenario we have $n = b - 1$ to prevent timing violation while minimizing the value of E_O. Therefore the comparison between the two scenarios in mean error value is given in (5) by updating (2) and (4) respectively. As seen in (5), in theory the mean value of error in the overclocking scenario (E_{oc}) is constantly two times smaller than that in the traditional scenario (E_{trad}).

$$\frac{E_{oc}}{E_{trad}} = \frac{2^{-b} - 2^{-n-1}|_{n=k}}{2^{-n} - 2^{-k}|_{n=b-1}} = \frac{1}{2} \tag{5}$$

B. Carry Select Adder

Although smaller value of mean error can be achieved in the overclocking scenario, it costs extra area because the full precision is maintained. Instead, alternative adder structures, such as carry select adder (CSA), are originally designed to trade silicon area for low latency. In a CSA, the carry chain is divided into multiple overlapped stages, and each stage contains two RCAs and two multiplexers. For a given input, two additions are performed simultaneously within a single stage where the carry input is zero and one separately. One of these two results is then selected according to the actual carry input. Although this structure brings performance benefits, it costs extra hardware resources compared to a standard RCA because the carry chain is duplicated. Furthermore, multiplexers are area-expensive to implement with FPGA technology.

C. Online Adder

In addition to standard binary arithmetic, alternative forms of computer arithmetic were studied to boost performance. For example, adder with online arithmetic is also designed for low latency at the cost of extra silicon area. The structure of a digit-parallel online adder (OA) where all signals represented with SD numbers of digit set $\{-1, 0, 1\}$ is shown in Fig. 3. The module "3:2" denotes a 3:2 compressor, which takes three inputs and generates two outputs, and is logically equivalent to a full adder (FA). A major advantage of the redundant number system over the standard ripple-carry based arithmetic is that the propagation of carry is eliminated, resulting in a precision-independent computation time for addition. As labeled in Fig. 3, ideally the computation delay of this adder is only two FA delays for any operand word-length, at the expenses of one extra FA for each digit of operands.

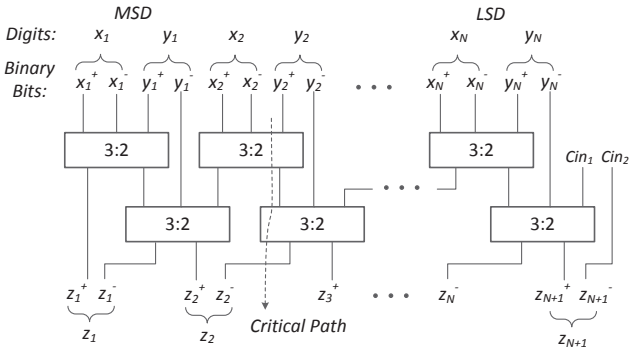


Fig. 3. An N-digit binary digit-parallel online adder. Both inputs and outputs are represented using SD representation. "3:2" denotes a 3:2 compressor.

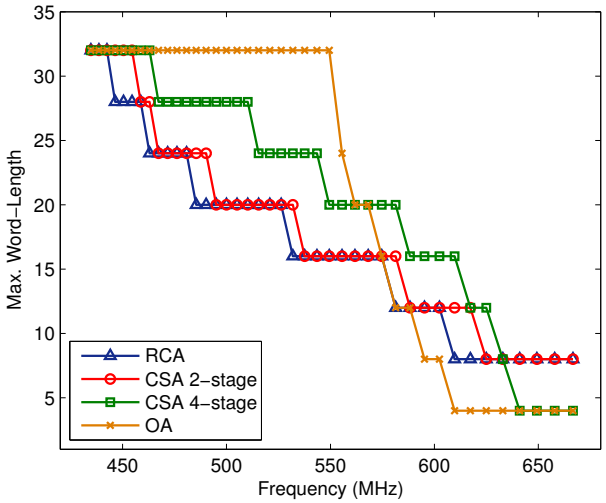


Fig. 4. The maximum word-lengths of different adder structures with respect to a variety of frequencies. The results are obtained from Xilinx ISE 14.7.

D. Design Trade-offs for Adders

In this section, we demonstrate the benefits of CSA and OA in accuracy and performance, as well as the area overhead in comparison to RCA. As an example, the maximum word-lengths with respect to a variety of operating frequencies for RCA, CSA and OA are illustrated in Fig. 4. In this experiment, circuits are operated with timing closure, i.e. the conventional design scenario is evaluated for all structures. We consider the operand word-length of adders ranging from 32-bit to 4-bit.

As can be seen in Fig. 4, for a relatively relaxed frequency requirement (e.g. 470 MHz), both CSA and OA can be implemented with greater word-lengths than RCA. For CSA,

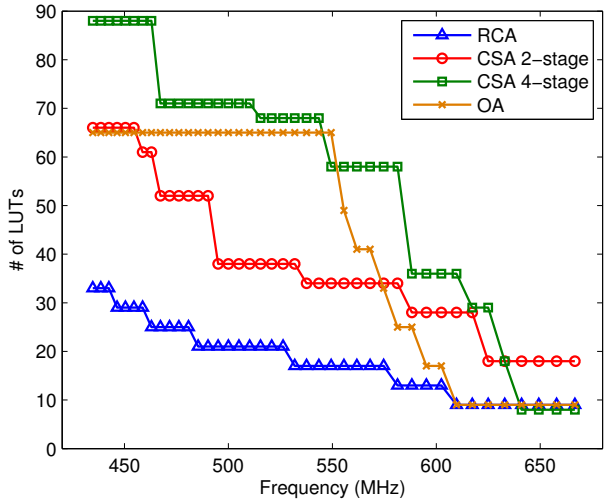


Fig. 5. The maximum area of different adder structures.

TABLE I. ADDER STRUCTURES AND DESIGN SCENARIOS CONSIDERED IN THIS PAPER.

Adders	**Design Scenarios**	
	Overclocking	Traditional
Ripple Carry Adder	√	√
Carry Select Adder		√
Online Arithmetic Adder		√

this is because the stage parallelism enables a larger word-length, even though the multiplexer delay limits the precision of each CSA stage in comparison to RCA. For OA, an even larger gap in word-length can be observed when compared to RCA. Additionally, unlike RCA and CSA of which the maximum word-length drops gradually with the increment of frequency, OA maintains full precision across a large range of frequencies. This is expected, because the critical path delay of OA is theoretically irrelevant to the operand word-length as discussed in Section III-C. However, we also observe that given a large frequency requirement, the word-length of OA drops drastically. Besides, CSA cannot be implemented with larger precision than RCA either, because in this case the multiplexer delay becomes comparable to the delay of the carry chain for CSA, and it inhibits the benefits of parallelism.

We also record the corresponding area consumption of each adder structure in terms of look-up-tables (LUTs) in FPGA as depicted in Fig. 5. It can be seen that the CSA with 4 stages and 2 stages costs up to $4.5\times$ and $3.1\times$ area than RCA, respectively. OA is also up to $3.8\times$ larger than RCA. Therefore, it is necessary to provide comprehensive evaluations of different adders by considering the trade-offs between accuracy, performance and area consumption. In summary, the evaluated adder structures as well as design scenarios are summarized in Table I.

IV. EVALUATION OF OPTIMUM ADDER STRUCTURE

Generally, our evaluation could be of interest to a circuit designer in several ways. For example, suppose the algorithm designer wish the circuit to run at a given frequency within a specific area budget, while achieving the minimum output error. Besides, for many applications, the circuit designer would wish the circuit to operate as fast as possible with the minimum resource usage, whilst a certain error budget can be tolerated. In both cases, decisions must be made on which adder structure achieves this minimum, and which design scenario should be adopted. In our experiments, the inputs are set to randomly generated data following uniform distribution.

A. Optimum Adder with Given Frequency and Area Requirements

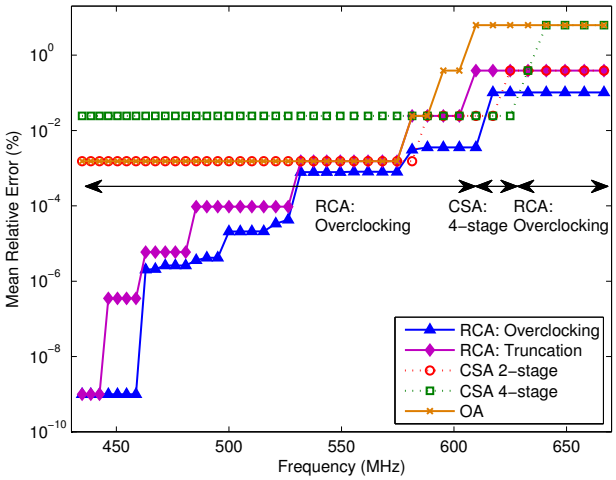
For this type of applications, the available area and expected operating frequency are given at the design time. As an example slice through the design space, in Fig. 6 we record the mean relative error (MRE) with respect to a range of operating frequencies for different design scenarios, when the area budget is set to 35 LUTs and 55 LUTs separately. MRE can be calculated as given by (6), where E_{error} and E_{out} denote the mean value of error and the mean value of correct outputs, respectively. Notice that the optimum adder design metric that achieves the minimum MRE is labeled.

$$MRE = \frac{E_{error}}{E_{out}} \times 100\% \tag{6}$$

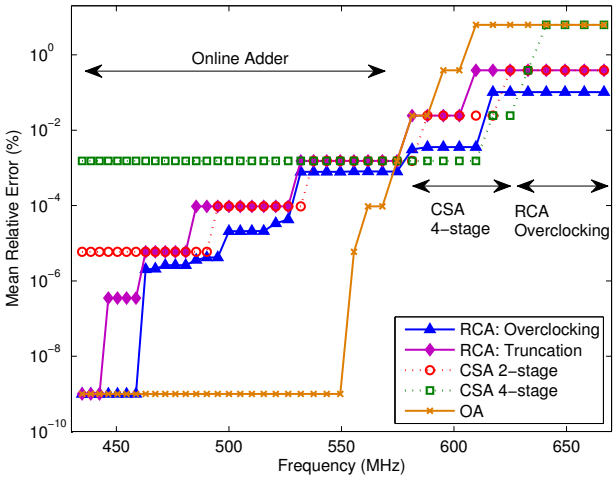
In Fig. 6(a), the area budget is set to 35 LUTs. For all frequency values, the overcloced RCA achieves no larger MRE than the RCA with truncated operand word-lengths. This is in accordance with the analysis in Section III-A. We also notice that both CSA and OA cannot be implemented with full precision due to area limitation, and this leads to large truncation errors. Despite that the CSA with 4 stages is best for some frequencies, in general the overclocked RCA is the optimum design for most frequency requirements.

However if the area budget is released to 55 LUTs, OA can be implemented with original word-length. Additionally CSA can also be implemented with larger precision, but still with truncation errors. Therefore as shown in Fig. 6(b), OA serves as the optimum design with respect to a wide range of frequency requirements. For higher frequencies, the MRE of OA increases rapidly, and CSA with 4 stages outperforms for higher frequencies. Similar to Fig. 6(a), the overclocked RCA is the optimum design for even higher frequencies, because in this case the CSA can only be implemented with small precisions, and the multiplexer delay limits the benefits of parallelism.

The results shown in Fig. 6 can be further extended by testing with a variety of area constraints. In this case, we can map the optimum design scenarios into the design space of different operating frequencies and area consumptions, as shown in Fig. 7. In general this graph can be divided into several regions. Firstly, if the frequency requirement is moderate whilst the area budget is large enough to implement an OA in full precision, it will be the optimum design. This is because long carry chains are eliminated from OA such that it can maintain full precision for a wide range of frequency values. Also this is in accordance with the results in Fig. 6(b). Secondly, CSA is a better design choice for high frequency requirements and large area budget, because it is originally designed with high speed operation. Besides, each stage of CSA is implemented in the FPGA with fast carry logic [20], hence it runs even faster than OA. Thirdly, for a tighter area budget only part of



(a) Available LUT=35.



(b) Available LUT=55.

Fig. 6. Two examples of comparisons between different design scenarios and adder implementations with limited area budget. RCA is evaluated with both overclocking and truncation scenario, whereas CSA and OA are evaluated with truncation scenario only. Design scenario with minimum error is labeled.

OA and CSA can be implemented, whereas RCA still keeps original precision. In this situation, area becomes the limitation factor. Both OA and CSA generate truncation errors, which are greater than the overclocking error of RCA. Therefore, the overclocked RCA is the optimum design option across almost the entire frequency domain. Furthermore, the precision of RCA is also limited under very stringent area constraints. This causes truncation errors for every design option. Nevertheless, RCA with either overclocking or truncation scenario equally serves to be optimal, because it maintains more precision than OA and CSA for a specific area budget.

B. Optimum Adder with Given Accuracy and Area Budgets

If the circuit is designed to operate as fast as possible with minimum area, whilst a certain amount of imprecision can be tolerated, the optimum adder design metric can also be determined as illustrated in Fig. 8. In this set of experiments, the error budget is evaluated in terms of MRE, which varies from $10^{-6}\%$ to 10%. For a given MRE, the optimum adder is selected with maximum operating frequency. In this case, if multiple designs can operate with the same frequency, the optimum design is then selected based on actual area consumption within the area constraint.

Similar to the analysis in Section IV-A, for tight accuracy requirement and large area budget, OA is the optimum design choice because it keeps full precision while running at high frequencies. If relaxing the accuracy requirement (e.g. 0.01%), CSA gradually outperforms because its precision loss is less than that of OA. Once again, if area budget is tightened, the RCA becomes the best design choice, because the precision of OA and CSA is largely limited. Furthermore, when the accuracy requirement is released (e.g. over 1%), RCA can operate with the fastest frequency across most area constraints.

V. Conclusion

In this paper, we have quantified design trade-offs for three different adder structures: ripple carry adder, carry select adder and online adder. Two design scenarios that can sacrifice accuracy for better performance have been evaluated: one is the conventional design scenario in which timing closure is achieved by truncating precision, the other is the overclocking scenario where timing violations are allowed to occur. By combining both adder structures and design scenarios, we have demonstrated experimentally the optimum design option when considering a variety of frequency-accuracy-area trade-offs. We have shown that applying the overclocking scenario to RCA can be beneficial with limited area budget, whilst applying the conventional scenario to online adder is an optimal design choice with relaxed area requirements.

In the future we wish to provide similar evaluation to other arithmetic primitives, such as multipliers. Other evaluation metrics, such as energy consumption and throughput, can also be included in the future.

References

[1] H. Esmaeilzadeh, E. Blem, R. S. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," in *Proc. Int. Symp. Computer Architecture*, 2011, pp. 365–376.

[2] G. A. Constantinides, N. Nicolici, and A. B. Kinsman, "Numerical data representations for FPGA-based scientific computing," *IEEE Design Test of Computers*, vol. 28, no. 4, pp. 8–17, 2011.

[3] K. Shi, D. Boland, and G. A. Constantinides, "Accuracy-performance tradeoffs on an fpga through overclocking," in *Proc. Int. Symp. Field-Programmable Custom Computing Machines*, vol. 0, 2013, pp. 29–36.

[4] M. D. Ercegovac, "On-line arithmetic: An overview," in *Proc. Annual Technical Symp. Real time signal processing VII*, 1984, pp. 86–93.

[5] S. I. Association, *International technology roadmap for semiconductors (ITRS)*, 2007.

[6] Z. M. Kedem, V. J. Mooney, K. K. Muntimadugu, and K. V. Palem, "An approach to energy-error tradeoffs in approximate ripple carry adders," in *Proc. Int. Symp. on Low Power Elec. and Design*, 2011, pp. 211–216.

[7] J. M. Levine, E. Stott, G. A. Constantinides, and P. Y. K. Cheung, "Online measurement of timing in circuits: For health monitoring and dynamic voltage & frequency scaling," 2012, pp. 109–116.

[8] T. Austin, D. Blaauw, T. Mudge, and K. Flautner, "Making typical silicon matter with razor," *IEEE Trans. on Computer*, vol. 37, no. 3, pp. 57–65, 2004.

[9] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger, "Architecture support for disciplined approximate programming," in *Proc. Int. Conf. Architectural Support for Programming Languages and Operating Systems*, 2012, pp. 301–312.

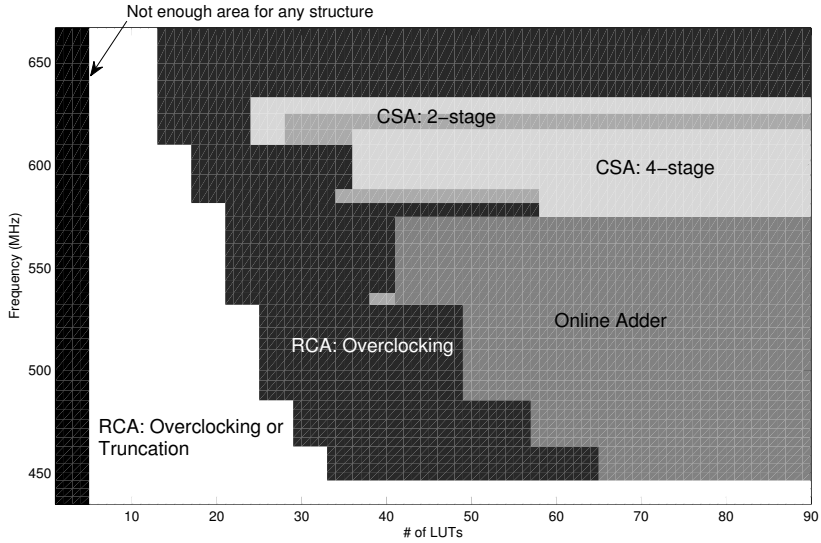


Fig. 7. Mapping of optimum design metric of adders, which achieves minimum error with respect to a variety of frequency and area constraints. The original word-length of all adders is 32-bit. The results are obtained from Xilinx ISE 14.7.

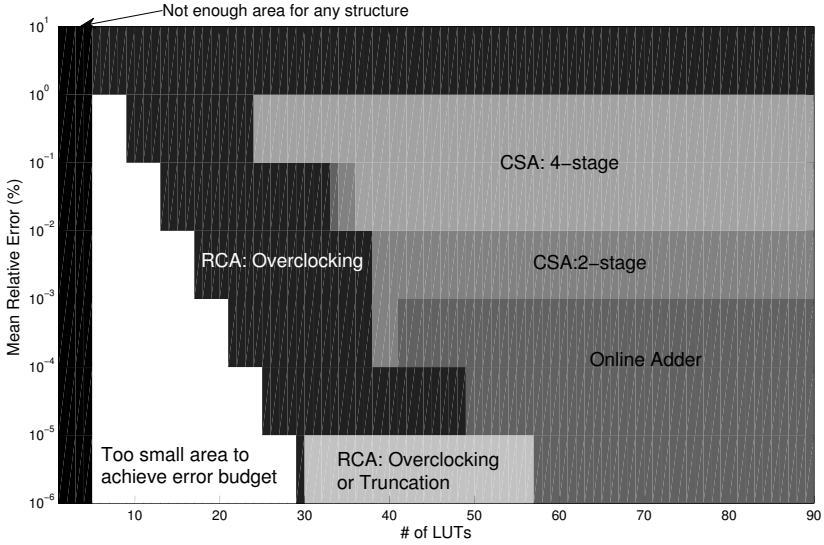


Fig. 8. Mapping of optimum design metric of adders, which achieves highest frequency with respect to a variety of accuracy and area constraints. The original word-length of all adders is 32-bit. The results are obtained from Xilinx ISE 14.7.

[10] A. Sampson, W. Dietl, E. Fortuna, D. Gnanapragasam, L. Ceze, and D. Grossman, "Enerj: Approximate data types for safe and general low-power computation," in *Proc. ACM SIGPLAN Notices*, vol. 46, no. 6, 2011, pp. 164–174.

[11] J. Liang, J. Han, and F. Lombardi, "New metrics for the reliability of approximate and probabilistic adders," *IEEE Transactions on Computers*, vol. 62, no. 9, pp. 1760–1771, Sept 2013.

[12] C. Liu, J. Han, and F. Lombardi, "A low-power, high-performance approximate multiplier with configurable partial error recovery," in *Proc. of the Conf. on Design, Automation & Test in Europe*, 2014, pp. 95:1–95:4.

[13] P. Kulkarni, P. Gupta, and M. Ercegovac, "Trading accuracy for power with an underdesigned multiplier architecture," in *Proc. Int. Conf. on VLSI Design*, 2011, pp. 346–351.

[14] Y. Harata, Y. Nakamura, H. Nagase, M. Takigawa, and N. Takagi, "A high-speed multiplier using a redundant binary adder tree," *IEEE Jourl. of Solid-State Circuits*, vol. 22, no. 1, pp. 28–34, 1987.

[15] R. Galli and A. F. Tenca, "Design and evaluation of online arithmetic for signal processing applications on FPGAs," in *Proc. SPIE Advanced Signal Processing Algorithms, Architectures and Implementations*, Aug 2001, pp. 134–144.

[16] M. Dimmler, A. Tisserand, U. Holmbeg, and R. Longchamp, "On-line arithmetic for real-time control of microsystems," *IEEE/ASME Trans. on Mechatronics*, vol. 4, no. 2, pp. 213–217, Jun 1999.

[17] M. D. Ercegovac and T. Lang, *Digital arithmetic*. Morgan Kaufmann, 2003.

[18] T. Kim, W. Jao, and S. Tjiang, "Circuit optimization using carry-save-adder cells," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, no. 10, pp. 974–984, 1998.

[19] A. Avizienis, "Signed-digit number representations for fast parallel arithmetic," *IRE Trans. on Electronic Computers*, vol. EC-10, no. 3, pp. 389–400, 1961.

[20] Xilinx Inc., "Virtex-6 FPGA configurable logic block user guide."