Imprecise Datapath Design: An Overclocking Approach

KAN SHI, DAVID BOLAND
and GEORGE A. CONSTANTINIDES, Imperial College London

Releasing the tight accuracy requirement would potentially offer greater freedom to create a design with better performance or energy efficiency. In this paper, we compare two different approaches that could trade accuracy for performance. One is the traditional approach where the precision used in the datapath is limited to meet a target latency. The other is a proposed new approach which simply allows the datapath to operate without timing closure. We demonstrate analytically and experimentally that on average our approach obtains either smaller errors for equivalent faster operating frequencies in comparison to the traditional approach. This is because the worst case caused by timing violations only happens rarely, while precision loss results in errors to most data. We also show that for basic arithmetic operations such as addition, applying our approach to the simple building block of ripple carry adders can achieve better accuracy or performance than using faster adder designs to achieve similar latency.

Categories and Subject Descriptors: C.2.2 [**Category put here**]: Detail category

General Terms: Design, FPGA, Performance

Additional Key Words and Phrases: Overclocking, Imprecise Design

ACM Reference Format:
Kan Shi, David Boland, and George A. Constantinides, 2013. Imprecise Datapath Design: An Overclocking Approach. *ACM Trans. Reconfig. Technol. Syst.* 0, 0, Article 0 (2014), 23 pages.
DOI:http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

Performance is a critical issue when designing embedded computing systems. Specifically, a large volume of current studies has demonstrated that, significant performance gains can be achieved by FPGA-based accelerators over software designs across a wide range of applications [Underwood 2004; Boland and Constantinides 2008]. However, one of the major factors that limits the performance of these accelerators is that they typically run at much lower clock frequencies than general purpose processors (GPPs) or graphics processing units (GPUs).

In order to boost the operating frequency of a datapath, the standard approaches are either to heavily pipeline the design or to reduce the precision throughout the datapath. For the former method, it should be noted that pipelining will not tend to reduce the latency of the datapath, whilst the latency in terms of clock cycles will actually increase. As a result, this method will not be applicable to many embedded applications, which often have strict latency requirements, or in any datapath containing feedback where C-slow retiming is inappropriate. For the second approach, reducing the datapath precision would reduce the latency at the cost of introducing quantization errors

This work is supported by the EPSRC, under grant EP/I020557/1 and grant EP/I012036/1.
Author's addresses: K. Shi, D. Boland and G. A. Contantinides, Department of Electrical and Electronic Engineering, Imperial College London.

DOI:http://dx.doi.org/10.1145/0000000.0000000

into the design. This has inspired research into exploiting the potential benefits of using the minimum precision necessary to satisfy a certain accuracy or performance specification [Constantinides et al. 2011]. A brief review of existing approaches in this area will be discussed in Section 2.1.

Unfortunately, neither of these conventional approaches tends to remove the conservative safety margin, which is introduced to ensure a uniform functionality across a variety of possible working environments. Nevertheless, continuing with employing the guard-bands to cover all worst cases would become increasingly difficult and expensive in the short-term future, because of the continuous scaling of the process technology. In recent years, we have seen a growth of research that explores the potential power or performance benefits that can be obtained when relaxing the guard-bands [Colwell 2004]. The detailed review of this research field will be presented in Section 2.2.

In this paper, we describe an alternative circuit design methodology when considering trade-offs between accuracy, performance and silicon area. Since it is unlikely to create a completly error-free design as any data representation would introduce quantization errors, we take a radical shift forward by suggesting that for certain applications it is beneficial to move away from the traditional model of creating a conservative design that is guaranteed to avoid timing violations. Instead, it may be preferable to create a design in which timing violations may occur, under the knowledge that they only occur rarely because specific input patterns are required to generate the worst case errors. This paper elaborates on the prior work [xxx], by incorporating silicon area as another evaluation metric. This enables us to find the optimum design choice of the basic arithmetic operators under various design constraints. A summary of the main contributions of this work are as follows:

— Detailed modeling methods of probabilistic errors generated in basic arithmetic primitives from two design scenarios to trade accuracy for performance: overclocking the datapath or truncating precisions that used throughout the datpath,
— Exploring the accuracy-performance-area trade-offs for key arithmetic operators to illustrate the optimum architecture for a certain arithmetic function under a given requirement of silicon area,
— Demonstration of the proposed findings analytically and experimentially from FPGA implementations that allowing rare timing violations to happen leads to a reduction of average errors or an improvement of performance over trancating the datapath precisions to meet timing.

The rest of this paper is organized as follows. It first reviews in detail the current literature regarding the approaches used in word-length optimization and approximate datapath design in Section 2. The analytical error models for two different adder structures: the ripple carry adder (RCA) and the carry-select adder (CSA), are discussed in Section 3 and Section 4, respectively. The decision of the optimum adder structure under the accuracy-performance-area trade-offs, is then put forward in Section 5. It is followed by a description of the probabilistic error models for another arithmetic operator, the constant coefficient multiplier (CCM), in Section 6. Section 7 discusses a practical experimental setup to verify our models and test the proposed design methodology. We then demonstrate the benefits of our proposed approach in Section 8. Finally, conclusion of this work and possible future work are discussed in Section 9.

2. BACKGROUND

2.1. Word-length Optimization

There exists a significant amount of work demonstrating that optimizing the datapath precision would bring a substantial benefits on clock speed, silicon area and power

consumption. There are two main research directions in this area. The first discusses methods to analyze the errors caused by the usage of finite precision and compute bounds on the worst-case errors seen at the output of the design. Related techniques include the simulation-based approach [Sung and Kum 1995] and the analytical-based approach such as interval arithmetic [Moore 1966], affine arithmetic [De et al. 2004], Satisfiability-Modulo Theories [Kinsman and Nicolici 2010] and polynomial representations [Boland and Constantinides 2011]. A detailed summary of these methods can be found in [Constantinides et al. 2011]. In the second research field, these tools are utilized to choose the minimum precision within a datapath to satisfy a required accuracy specification [Roldao-Lopes et al. 2009]. In this work, two different design scenarios are analyzed. In the conventional scenario, we speculate and reduce the original word-length in the datapath under a given timing constraint while ensuring no timing errors will occur. In the proposed new scenario, we operate the datapath with the original word-length while allowing timing violations sometimes to occur. The truncation errors derived in the former scenario are then compared to the overclocking errors generated in our proposed new scenario.

2.2. Imprecise Circuit Design Methodology

It is worth noting that the choice of precision is not the only source of error when designing a datapath. Recently we have seen a growth of parallel streams of research which aim at exploring alternative methods to trade accuracy for design efficiency. This strand of research is motivated by the fact that extra benefits of manufacturing, test, power and timing can be obtained if the tight requirement of absolute correctness is released for devices and interconnect, as pointed out by the International Technology Roadmap for Semiconductors (ITRS) in 2007 [Association 2007]. This topic is expected to be of growing importance in the future, because we will face new design challenges as the technology scales further.

According to the existing literature, one way to operate circuits beyond the deterministic region is to relax the design constraints and the guard-bands that are conventionally used to avoid the worst cases. A series of work known as "Better-Than-Worst-Case (BTWC) Design" introduced a universal framework to push the circuit performance to its limits [Austin et al. 2005]. In general, a BTWC design is composed of cores and checkers. The cores are operated with high performance by eliminating the guard-bands, meanwhile the possible timing errors are diagnosed by the checkers. Optionally the system can be recovered at the observation of errors. As an exemplary design, the Razor project [Ernst et al. 2003; Austin et al. 2004] was proposed to shave the conservative timing margins by overscaling the supply voltage and clock frequency, while monitoring the output error rates by utilizing a self-checking circuit. This work demonstrated that the benefits brought by removing the safe margin outweigh the cost of monitoring and recovering from errors. For example, 22% or over 30% power consumption can be saved with $\sim 0.01\%$ or $\sim 1\%$ error rates at the output, respectively. Related work also includes a similar frequency overscaling technique by operating circuits slightly slower than the critical path delay with dedicated checker circuits to ensure that timing errors will not occur [Uht 2004], or developing timing analysis tools that decide the optimum operating frequencies in the non-deterministic region due to process variation [Keutzer and Orshansky 2002]. However, the major problem of the BTWC work is the extra cost introduced by implementing the checker circuits.

Another field of study has shown that errors can be potentially tolerated in various applications, e.g. DSP applications involving human perception. In this case, even greater performance or power benefits can be achieved, as the design overhead of the checkers can be eliminated. Current research in this area focuses on designing probabilistic or imprecise circuits both at a software level and a hardware level. For in-

stance at the programming language level, a tool was proposed [Sampson et al. 2011; Esmaeilzadeh et al. 2012] to divide the program into the precise parts and the approximate parts, which are mapped to different hardware with different speed-grades, supply voltages etc., respectively. This technique enables a relatively high service quality, in the meantime energy reduction can be obtained due to approximation. Similar ideas have also been applied on hardware directly. As an example, Palem et al. [Kedem et al. 2011] described a non-uniform voltage scaling technique for the ripple carry adder. In this study, multiple voltage regions are employed for different bits along a carry chain. That is, higher voltage would be applied for computations generating most significant bits, and vice versa. Nevertheless, utilizing several voltage regions within a ripple carry adder will rarely be practical in real situations. Furthermore, the overhead of applying this technique is not addressed.

While the aforementioned literature take advantage of the fact that only specific input patterns could cause timing errors, research on imprecise architectures take this one step further by designing simplified circuits for peformance and/or energy efficiency. For example, Lu et al. proposed a "shrinking" datapath that can be utilized to mimic and speculate the original logic functions [Lu 2004]. Kulkarni et.al. described an underdesigned 2×2 multiplier unit, of which the worst case was replaced by a normal case based on the straight-forward Karnaugh-Map analysis [Kulkarni et al. 2011]. In both cases, reduction of area and power consumption can be achieved with the cost of accuracy. In addition, Gupta et al. developed approximate adders at the transistor level and compared the energy efficiency of their proposed architectures over truncation of input word-length of conventional structures [Gupta et al. 2013]. However, we should note that using imprecise architectures means correct results would never be obtained for certain input patterns. Another limitation is that the link between the probability of output correctness and energy savings or performance improvements are rarely analyzed in current research. In addition, these techniques cannot be directly applied onto existing hardware platforms such as FPGAs.

In this work, we propose an alternative methodology to remove the limitations of those existing approaches. It is argued in this paper that our method can be easily applied to existing hardware arithmetic operators, and we support this hypothesis analytically by presenting theoretical probabilistic error models, and experimentally by using FPGAs.

3. RIPPLE CARRY ADDER

3.1. Adder Structures in FPGAs

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, the philosophy of our approach is first exemplified with the analysis of a RCA. We later describe how this methodology can be extended to other arithmetic operators such as CSAs and CCMs, which are commonly used in DSP applications and numerical algorithms, in Section 4 and Section 6 respectively.

Typically the maximum frequency of a RCA is determined by the longest carry propagation delay. Consequently, modern FPGAs offer built-in architectures for very fast ripple carry addition. For instance, the Altera Cyclone series uses fast tables [Altera 2008] while the Xilinx Virtex series employs dedicated multiplexers and encoders for the fast carry logic [Xilinx b]. Fig. 1 illustrates the structure of an n-bit RCA, which is composed of n serial-connected full adders (FAs) and utilizes the internal fast carry logic of the Virtex-6 FPGA.

While the fast carry logic reduces the time of each individual carry-propagation delay, the overall delay of carry-propagation will eventually overwhelm the delay of sum

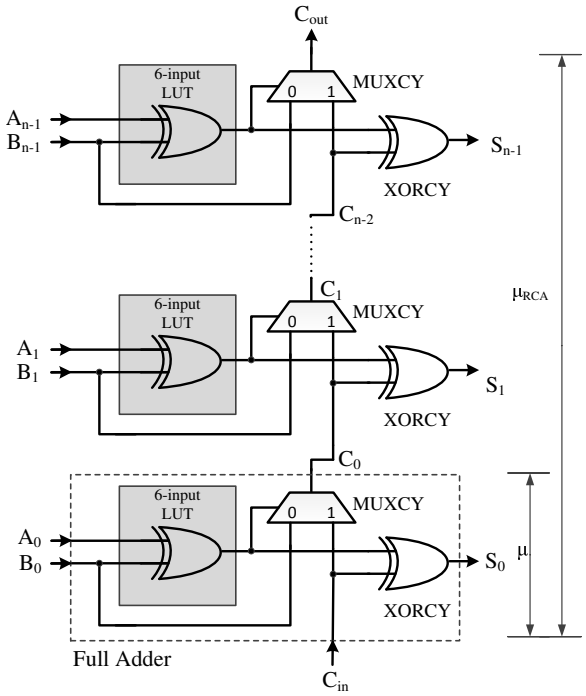


Fig. 1. An n-bit ripple carry adder in the Xilinx Virtex-6 FPGA.

generation of each LUT with increasing operand word-lengths. We perform our initial analysis based on the assumption that the carry propagation delay of each FA is a constant value μ_c, which is a combination of logic delay and routing delay. Hence the critical path delay of the RCA is given by (1), as shown in Fig. 1.

$$\mu_{RCA} = n \cdot \mu_c \tag{1}$$

For an n-bit RCA, it follows that if the sampling period T_S is greater than μ_{RCA}, correct results will be sampled. If, however, faster-than-rated sampling frequencies are applied such that $T_S < \mu_{RCA}$, intermediate results will be sampled, potentially generating errors.

In the following sections, we consider two methods that would allow the circuit to run at a frequency higher than $1/T_S$. The first is a traditional circuit design approach where operations occur without timing violations. To this end, the operand word-length is truncated in order to meet the timing requirement. This process would result in truncation or roundoff error. In our proposed new design scenario, circuits are implemented with greater word-length, but are clocked beyond the safe region so that timing violations sometimes occur. This process would generate "overclocking error".

3.2. Probabilistic Model of Truncation Error

For ease of discussion, we assume that the input to our circuit is a fixed point number scaled to lie in the range $[-1, 1)$. In our initial analysis, we assume every bit of each input is uniformly and independently generated. However, this assumption will be relaxed in Section 8 where the predictions are verified using real image data. The errors at the output are evaluated in terms of the absolute value and the probability of their occurrence. These two metrics are combined as the error expectation.

If the input signal of a circuit contains k bits, truncation error occurs when the input signal is truncated from k bits to n bits. Under this premise, the mean value of the truncated bits at signal input (E_{Tin}) is given by (2).

$$E_{Tin} = \frac{1}{2} \sum_{i=n+1}^{k} 2^{-i} = 2^{-n-1} - 2^{-k-1} \tag{2}$$

Since we assume there are two mutually independent inputs to the RCA, the overall expectation of truncation error for the RCA is given by (3).

$$E_T = \begin{cases} 2^{-n} - 2^{-k}, & \text{if } n < k \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

3.3. Probabilistic Model of Overclocking Error

3.3.1. Generation of Overclocking Error. For a given T_S, the maximum length of error-free carry propagation is described by (4), where f_S denotes the sampling frequency.

$$b := \left\lceil \frac{T_S}{\mu_c} \right\rceil = \left\lceil \frac{1}{\mu_c \cdot f_S} \right\rceil \tag{4}$$

However, since the length of an actual carry chain during execution is dependent upon input patterns, in general, the worst case may occur rarely. To determine when this timing constraint is not met and the size of the error in this case, we expand standard results [Rabaey et al. 2003] to the following statements, which examine carry generation, propagation and annihilation, as well as the corresponding summation results of a single bit i within an n-bit RCA ($0 \leq i < n$), according to the relationship between its input patterns A_i and B_i:

— Carry generation: $A_i = B_i = 1$, a new carry chain is generated at bit i, and $S_i = C_{i-1}$;
— Carry propagation: $A_i \neq B_i$, the carry propagates for this carry chain at bit i, and $S_i = C_{i-1} = 0$;
— Carry annihilation: $A_i = B_i$ (either 0 or 1), the current carry chain annihilates at bit i, and $S_i = C_{i-1} = 1$.

3.3.2. Absolute Value of Overclocking Error. For an n-bit RCA, let C_{tm} denote the carry chain generated at bit S_t with the length of m bits. For a certain f_S, the maximum length of error-free carry propagation, b, is determined through (4). The presence of overclocking error requires $m > b$. Since the length of carry chain cannot be greater than n, parameters t and m are bounded by (5) and (6):

$$0 \leq t \leq n - b \tag{5}$$

$$b < m \leq n + 1 - t \tag{6}$$

For C_{tm}, correct results will be generated from bit S_t to bit S_{t+b-1}. Hence the absolute value of error seen at the output, normalized to the MSB (2^n), is given by (7), where $\hat{S}_i$ and S_i denote the actual value and error-free value of outputs at bit i, respectively.

$$e_{tm} = \frac{\left| \sum_{i=t+b}^{n} (S_i - \hat{S}_i) \cdot 2^i \right|}{2^n} \tag{7}$$

S_i and $\hat{S}_i$ can be determined using the equations from the previous statements in Section 3.3.1. In the error-free case, the carry will propagate from bit S_t to bit S_{t+m-1}, and we will obtain $S_{t+b} = S_{t+b+1} = \cdots = S_{t+m-2} = 0$ for carry propagation, and $S_{t+m-1} = 1$ for carry annihilation. However, when a timing violation occurs, the carry will not propagate through all these bits. Hence we have $\hat{S}_{t+b} = \hat{S}_{t+b-1} = \cdots =$

$\hat{S}_{t+m-2} = 1$ for carry propagation, and $\hat{S}_{t+m-1} = 0$ for carry annihilation. Substituting these values into (7) yields (8). Interestingly as seen in (8), the magnitude of overclocking error has no dependence on the length of carry chain m.

$$e_{tm} = \frac{\left|2^{t+m-1} - 2^{t+m-2} - \cdots - 2^{t+b}\right|}{2^n} = 2^{t+b-n} \tag{8}$$

3.3.3. Probability of Overclocking Error. The carry chain C_{tm} occurs when there is a carry generated at bit t, a carry annihilated at bit $t+m-1$ and the carry propagates in between. Consequently, its probability P_{tm} is given by (9).

$$P_{tm} = P_{(A_t=B_t=1)} P_{(A_{t+m-1}=B_{t+m-1})} \cdot \prod_{i=t+1}^{t+m-2} P_{(A_i \neq B_i)} \tag{9}$$

Under the assumption that A and B are mutually independent and uniformly distributed, we have $P_{(A_i=B_i=1)} = 1/4$, $P_{(A_i \neq B_i)} = 1/2$ and $P_{(A_i=B_i)} = 1/2$, so P_{tm} can be obtained by (10). Note that (10) takes into account the carry annihilation always occurs when $t+m-1 = n$.

$$P_{tm} = \begin{cases} (1/2)^{m+1} & \text{if } t+m-1 < n \\ (1/2)^m & \text{if } t+m-1 = n \end{cases} \tag{10}$$

Furthermore, it can be seen that the probabilistic models for the magnitude and probability of overclocking errors can be easily generalized for any input distributions with mutually independent bits, since (7) and (8) are not related to the input distribution, while (9) can be employed as long as the value of $P_{(A_i=B_i=1)}$ and $P_{(A_i \neq B_i)}$ is known.

3.3.4. Expectation of Overclocking Error. Expectation of overclocking error can be expressed by (11).

$$E_O = \sum_t \sum_m P_{tm} \cdot e_{tm} \tag{11}$$

Using P_{tm} and e_{tm} from (8) and (10) respectively, E_O can be obtained in (12).

$$E_O = \begin{cases} 2^{-b} - 2^{-n-1}, & \text{if } b \leq n \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

3.4. Comparison between Two Scenarios

In the traditional scenario, the word-length of RCA must be truncated, using $n = b-1$ bits, in order to meet a given f_S. The error expectation is then given by (13).

$$E_{trad} = 2^{-b+1} - 2^{-k} \tag{13}$$

Overclocking errors are allowed to happen in our proposed new scenario. Therefore the word-length of RCA is set to be equal to the input word-length, that is, $n = k$. Hence we obtain (14) according to (12).

$$E_{new} = 2^{-b} - 2^{-k-1} \tag{14}$$

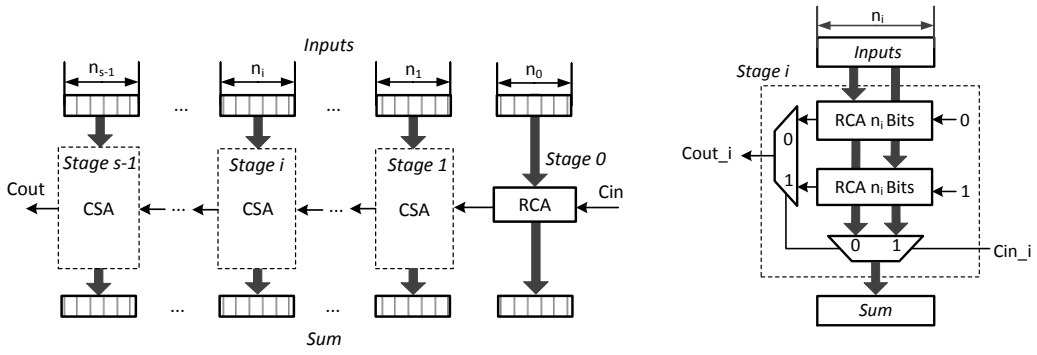
Comparing (14) and (13), we have (15). This equation indicates that by allowing timing violations, the overall error expectation of RCA outputs drops by a factor of 2 in comparison to traditional scenario. This provides the first hint that our approach is useful in practice.

$$\frac{E_{new}}{E_{trad}} = \frac{2^{-b} - 2^{-k-1}}{2^{-b+1} - 2^{-k}} = \frac{1}{2} \tag{15}$$

4. CARRY SELECT ADDER

4.1. Introduction

As seen in (15), a factor of 2 reduction in error expection can be achieved by using the proposed approach in RCA. However, implementing our approach costs extra silicon area in comparison to the conventional approach, because we keep the original word-length k in the new scenario instead of truncating to only $b-1$ bits in the traditional scenario. Besides our approach, there are existing approaches that could be employed to trade silicon area for low latency. For instance, alternative adder architectures, such as carry select adder (CSA), have been proposed to boost performance. In a CSA, the carry chain is divided into multiple overlapped stages, while each stage contains two RCAs and two multiplexers, as shown in Fig. 2(a) and Fig. 2(b), respectively. For a given input, two additions are performed simultaneously within a single stage where the carry input is zero and one seperately. 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, in FPGA technology, multiplexers are expensive to implement. Due to this reason, we explore the trade-offs between silicon area, accuracy and performance of RCA and CSA in this section.



(a) A CSA with s stages. (b) The i^{th} stage in the CSA.

Fig. 2. The structure diagram of CSA.

4.2. Timing Models for Carry Select Adder

We initially model the CSA in order to understand the relationship between the operating frequency and the maximum word-length of the CSA. This information can then be employed to determine the truncation error based on the models presented in Section 3.2.

In our analysis, the stage delay of the i^{th} stage in the CSA refers to the combination of the i-bit carry propagation delay and the delay of multiplexing the carry from the carry input of the i^{th} stage to the carry output of the CSA. For a CSA with s stages ($s \geqslant 2$), let the stage delay be denoted by $d_{s-1},\ \ldots,\ d_0$, where d_{s-1} and d_0 represent the delay of the most significant and the least significant stages, respectively. We still follow the aforementioned assumption that the critical path delay of the CSA is due to carry propagation and multiplexing the carry output, instead of generating the sum outputs. It should be noted that unlike other stages, the least significant stage of the CSA can be built only by one RCA without multiplexers, since it is directly driven by the carry input. Hence we can obtain the delay of the i^{th} stage as presented in (16),

where μ_c, μ_{mux} and n_i denote the delay of 1-bit carry propagation, the delay of multiplexing and the word-length of the i^{th} stage of the CSA, respectively.

$$d_i = \begin{cases} n_i \cdot \mu_c + (s-i) \cdot \mu_{mux}, & \text{if } i \in [1, s-1] \\ n_0 \cdot \mu_c + (s-1) \cdot \mu_{mux}, & \text{if } i = 0 \end{cases} \tag{16}$$

Under the timing-driven design environment, the delay of each stage of CSA is equalized in order to achieve the fastest operation, as presented in (17).

$$d_{s-1} = d_{s-2} = \cdots = d_0 \tag{17}$$

In this case, combining (16) and (17) yields n_i, which is represented by the word-length of the most significant stage n_{s-1}, in (18).

$$n_i = \begin{cases} n_{s-1} - (s-1-i) \cdot \frac{\mu_{mux}}{\mu_c}, & \text{if } i \in [1, s-1] \\ n_{s-1} - (s-2) \cdot \frac{\mu_{mux}}{\mu_c}, & \text{if } i = 0 \end{cases} \tag{18}$$

Sum up n_i to give the total word-length of the CSA in (19).

$$n_{CSA} = \sum_{i=0}^{s-1} n_i = s \cdot n_{s-1} - \frac{\mu_{mux}}{\mu_c} \cdot \frac{(s+1)(s-2)}{2} \tag{19}$$

Conventionally under a given frequency constraint, the word-length of RCA is truncated by using $n_{RCA} = b - 1$ bits, where b is determined by (4). Similarly for CSA, the word-length of each stage should be selected in order to satisfy (20).

$$\forall i \in [0, s-1] \ , \ d_i \leqslant \tfrac{1}{f_s} \tag{20}$$

Hence for the same frequency requirement, we can form the relationship between the delay of the most significant stage of CSA and the delay of RCA, as given by (21),

$$\mu_c \cdot (b-1) = n_{s-1} \cdot \mu_c + \mu_{mux} \tag{21}$$

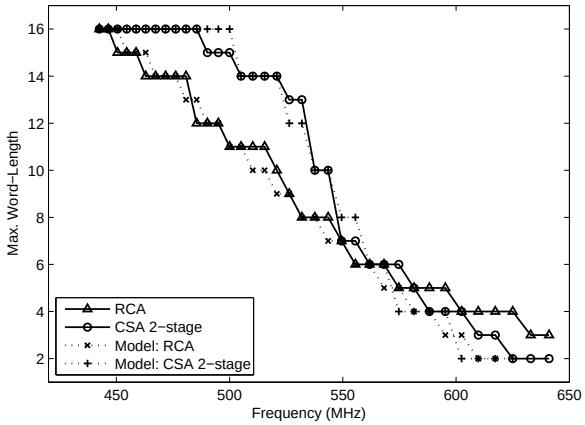
Substitute (21) into (19) to replace n_{s-1}, we derive the representation of the word-length of CSA in terms of b, as presented in (22).

$$n_{CSA} = s \cdot (b-1) - \frac{\mu_{mux}}{\mu_c} \cdot \frac{(s+2)(s-1)}{2} \tag{22}$$

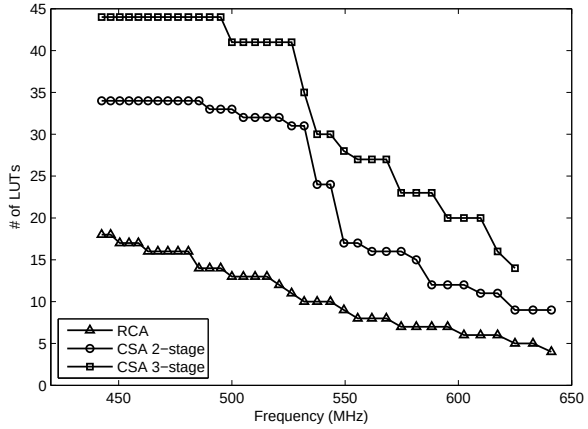
4.3. Accuracy Benefits and Area Overhead in CSA

We first verify the models for the RCA and CSA in terms of the maximum word-length under the given operating frequencies. For the CSA, the ratio μ_{mux}/μ_c can be computed experimentally. We perform post place-and-route simulations on the CSA with 2 stages using Xilinx Virtex-6 FPGA. The delay of the i^{th} stage d_i in (16) is recorded with respect to different word-lengths. The total word-length of CSA can be predicted through (22). In addition, the maximum word-lengths of the 2-stage CSA and RCA are obtained experimentally by increasing n_{CSA} and n_{RCA} respectively until errors are observed at the output. The comparison between the modeled value and the empirical results is illustrated in Fig. 3(a). It can be seen that our models for both RCA and CSA match well with the experimental results.

Fig. 3(a) also highlights that for a relatively relexed frequency cosntraint, the CSA achieves greater word-length than the RCA. This is because the stage parallelism in the CSA enables a greater overall word-length, even though the multiplexer delay limits the word-length of each stage in the CSA when compared to the RCA. When more stringent latency requirement is applied, however, the word-length of each stage is largely limited such that the multiplexer delay becomes comparable to the delay of the carry chain, and this inhibits the benefits of parallelism.



(a) The modeled value and the experimental results of the maximum word-length of RCA and CSA, with respect to various frequency requirements.



(b) Hardware resource usage for an RCA and a CSA with 2 and 3 stages.

Fig. 3. A comparison between RCA and CSA in terms of the maximum word-length of input signal and the area consumption.

However, the accuracy benefits brought by CSA comes at the cost of a large area overhead. Fig. 3(b) depicts the number of LUTs in the FPGA used for an RCA, a 2-stage CSA and a 3-stage CSA. It can be seen that in order to meet a given frequency, the 3-stage CSA consumes $2.4\times \sim 3.7\times$ area than RCA, while the 2-stage CSA requires $1.7\times \sim 3.1\times$ extra area. This finding poses a question of which is the best adder structure for a specific area budget.

5. CHOOSING THE OPTIMUM ADDER STRUCTURE

In Section 3, we discussed two design scenarios for the RCA when considering timing constraints. In this section, we expand our analysis by incorporating silicon area as another evaluation metric, and investigate the accuracy, performance and area trade-offs for different adder structures. In the conventional design scenario, the word-length of RCA and CSA is limited by the given frequency constraint, or/and the available hardware resources. The precision loss potentially generates large errors even without timing violations. However in the new design scenario, we use the RCA with the maxi-

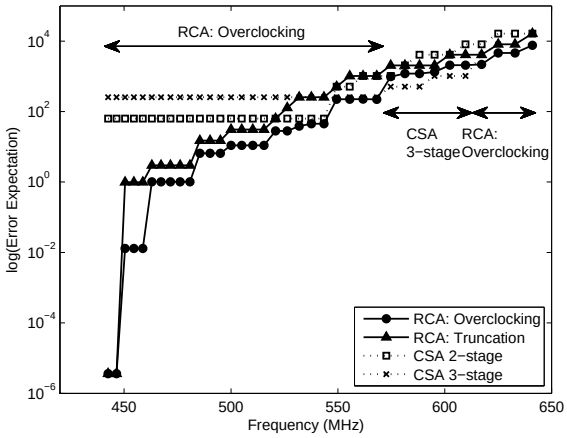


Fig. 4. A comparison between two design scenarios when the number of available LUT is 25. The RCA and CSA are investigated in the conventional scenario, while the RCA is explored in the proposed new scenario. The results are obtained from post place-and-route simulations on Xilinx Virtex-6 FPGAs.

mum possible word-length under the given area budget, and the timing constraints are allowed to be violated. This process might result in timing errors as well as truncation errors due to area limitation. We compare these two scenarios with different design goals with the aim of finding the optimum design methodology under each situation.

5.1. Determination of the Optimum Adder Structure for Given Frequency Requirements

First, suppose the algorithm designer wished to create a circuit that can run at a given frequency in a given resource constraint while achieving the minimum possible output error. A decision must then be taken over which adder structure achieves this minimum, and whether the structure should be overclocked

As an example slice through the design space, in Fig. 4 we record the mean value of error at outputs with respect to different operating frequencies for both design scenarios when the number of the available LUTs is set to 25. In this graph, we have labeled the optimum adder structure. Note that in this and all the following experiments within this section, in order to apply our models the input data is randomly generated following the uniform distribution.

We first notice that for all frequency values, the overclocked RCA achieves no larger error expectation than the RCA with truncated operand word-lengths, as predicted by the models for the RCA in Section 3.4. It can also be observed that the CSA cannot be implemented with the original word-length due to the limited area budget and this leads to large truncation errors. Although the CSA with 3 stages is best for some frequencies, the overclocked RCA is still the optimum design when high operating frequencies are applied.

We then perform similar experiments with a variety of area constraints. The optimum design methods with respect to different operating frequencies and area consumptions are demonstrated in Fig. 5. From this figure, several observations can be made. Firstly, if the frequency requirement is not large whilst the available area is large enough to implement a CSA in full precision, it will be the optimum design. This is expected from our earlier analysis in Fig. 3(a). Secondly, the 2-stage CSA is better than the 3-stage CSA when frequency is low, as it consumes less area although both achieve the same error expectation. Thirdly, only part of the CSA can be implemented under a tighter area budget, whereas the RCA still keeps full precision. In this case, area becomes the dominant factor and precision is lost for the CSA, meaning the RCA

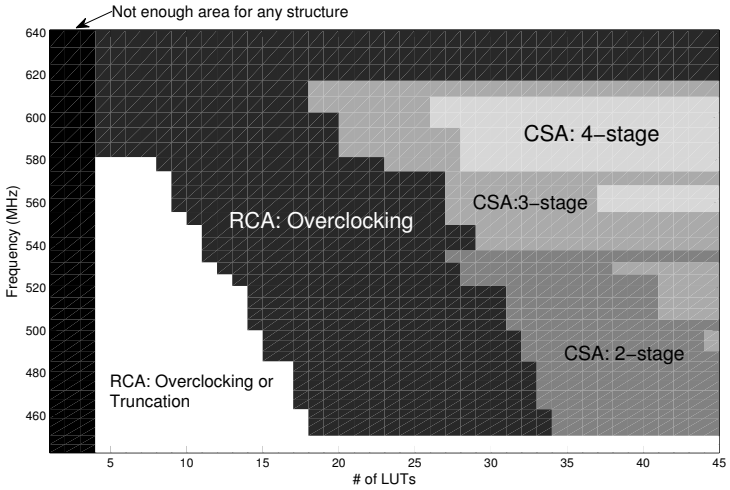


Fig. 5. A demonstration of the optimum design methodology which achieves the minimum error at outputs with respect to a variety of frequency and area constraints.

with overclocking is the optimum design method across almost the whole frequency domain in this situation. Finally, for very stringent area constraints, the word-length of RCA is also limited. This results in truncation errors initially for all design scenarios. However, the RCA with overclocking can still be employed as the optimum design, because it loses less precision than the CSA under the same area constraint.

5.2. Determination of the Optimum Adder Structure for Given Accuracy Requirements

If the design goal is to operate the circuit as fast as possible with the minimum area whilst a certain error budget can be tolerated, the optimum design methodology can be decided as illustrated in Fig. 6. In this situation, the error specifications are evaluated in terms of the mean relative error (MRE), as given by (23), where E_{error} and E_{out} refer to the mean value of error and the mean value of outputs, respectively.

$$MRE = \left| \frac{E_{error}}{E_{out}} \right| \times 100\% \tag{23}$$

In our experiments, MRE is set ranging from 0.001% to 50%. For a certain value of MRE, the design with the maximum operating frequency is selected as the optimum design. Moreover, the smallest design is the optimum one if multiple structures operate at the same frequency, with a certain area requirement. Fig. 6 can thus be obtained based on these criteria.

For a tight accuracy requirement, i.e. $MRE < 0.005\%$, CSA serves as the optimum design choice for large area constraints, as it intrinsically operates faster than RCA. Once again, when the area budget shrinks, the RCA performs best because the precision of the CSA is limited. Similarly to the results in Fig. 5, we see that the overclocked RCA achieves the fastest operating frequencies under most area constraints when the accuracy requirement is released.

5.3. Design Guidance

To sum up, the experiments reveal that for both design goals, the CSA is not compatitive with the overclocking approach unless one has very relexed area budget. As a result, for the remainder of this paper, we will focus on the RCA in our following analysis and experiments.

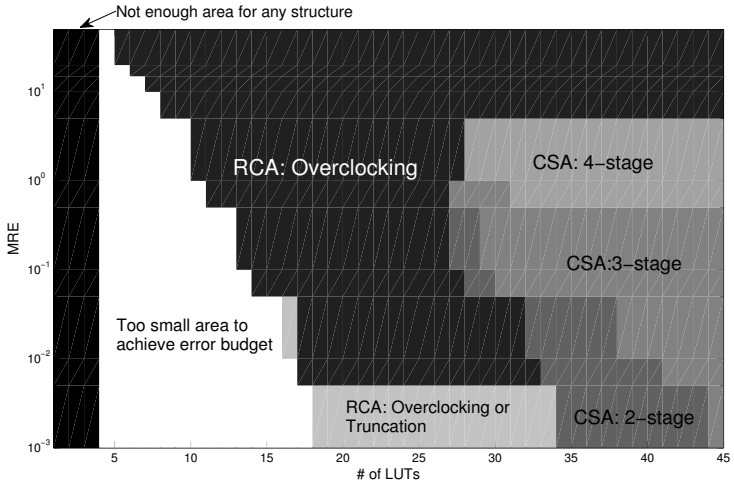


Fig. 6. A demonstration of the optimum design methodology which runs at the fastest frequency with respect to a variety of accuracy and area constraints.

6. CONSTANT COEFFICIENT MULTIPLIER

As another key primitive of arithmetic operations, CCM can be implemented using RCA and shifters. For example, operation $B = 9A$ is equivalent to $B = A + 8A = A + (A << 3)$, which can be built using one RCA and one shifter. We first focus on a single RCA and single shifter structure. We describe how more complex structures consisting of multiple RCAs and multiple shifters can be built in accordance with this baseline structure in Section 6.3.

In this CCM structure, let the two inputs of the RCA be denoted by A_S and A_O respectively, which are both two's complement numbers. A_S denotes the "shifted signal", with zeros padded after LSB, while A_O denotes the "original signal" with MSB sign extension. For an n-bit input signal, it should be noted that an n-bit RCA is sufficient for this operation, because no carry will be generated or propagated when adding with zeros, as shown in Fig. 7.

6.1. Probabilistic Model of Truncation Error

Let E_{Tin} and E_{Tout} denote the expectation of truncation error at the input and output of CCM respectively. We then have (24), where coe denotes the coefficient value of the CCM, and E_{Tin} can be obtained according to (3).

$$E_{Tout} = |coe| \cdot E_{Tin} \tag{24}$$

6.2. Probabilistic Model of Overclocking Error

6.2.1. Absolute Value of Overclocking Error. The absolute value of overclocking error of carry chain C_{tm} is increased by a factor of 2^s due to shifting, when compared to RCA. Hence e_{tm} in CCM can be modified from (8) to give (25).

$$e_{tm} = 2^{t+b-n+s} \tag{25}$$

6.2.2. Probability of Overclocking Error. Due to the dependencies in a CCM, carry generation requires $a_t = a_{t-s} = 1$, propagation and annihilation of a carry chain is best considered separately for four types of carry chain generated at bit t. We label these by $C_{tm}1$ to $C_{tm}4$ in Fig. 7, defined by the end region of the carry chain. For $C_{tm}1$, we have:

— Carry propagation: $a_i \neq a_{i+s}$ where $i \in [t+1, n-s-2]$;

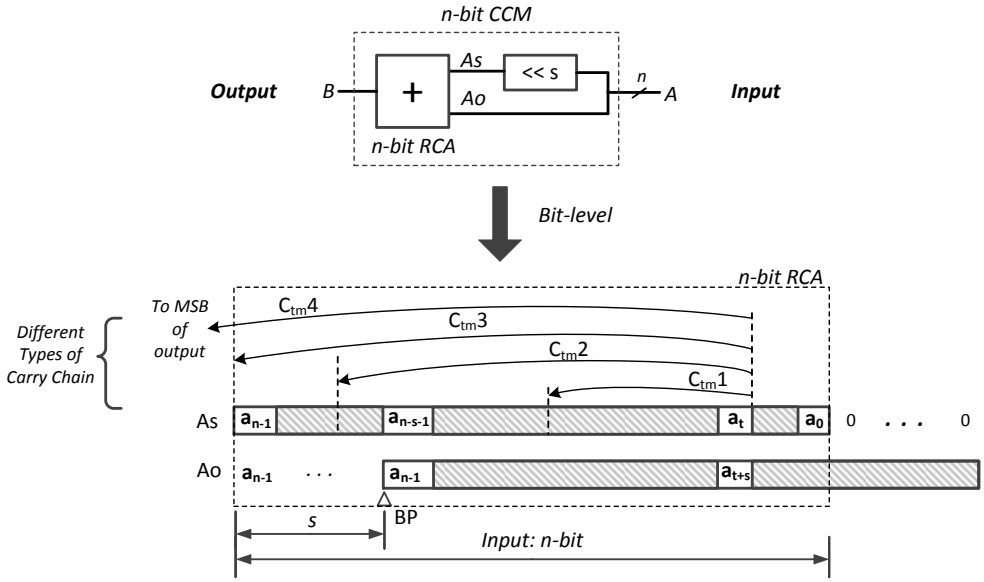


Fig. 7. Four possible carry chain types in a constant coefficient multiplier with n-bit inputs. The notion s denotes the shifted bits and BP denotes the binary point.

— Carry annihilation: $a_j = a_{j+s}$ where $j \in [t+1, n-s-1]$.

Similarly for $C_{tm}2$, we have:

— Carry propagation: $a_i \neq a_{n-1}$ where $i \in [n-s-1, n-3]$; or $a_i \neq a_{i+s}$ where $i \in [t+1, n-s-2]$;
— Carry annihilation: $a_j = a_{n-1}$ where $j \in [n-s-1, n-2]$.

For the first two types of carry chain $C_{tm}1$ and $C_{tm}2$, the probability of carry propagation and annihilation is $1/2$ and the probability of carry generation is $1/4$, under the premise that all bits of input signal are mutually independent. Therefore (26) can be obtained by substituting this into (9).

$$P_{tm} = (1/2)^{m+1}, \text{ if } t+m-1 \leqslant n-2 \tag{26}$$

For carry annihilation of $C_{tm}3$, it requires $a_{n-1} = a_{n-1}$, which is always true. Thus the probability of $C_{tm}3$ is given by (27).

$$P_{tm} = (1/2)^{m}, \text{ if } t+m-1 = n-1 \tag{27}$$

$C_{tm}4$ represents carry chain annihilates over a_{n-1}, therefore carry propagation requires $a_{n-1} \neq a_{n-1}$. This means $C_{tm}4$ never occurs in a CCM.

Altogether, P_{tm} for a CCM is given by (28).

$$P_{tm} = \begin{cases} (1/2)^{m+1} & \text{if } t+m-1 < n-1 \\ (1/2)^{m} & \text{if } t+m-1 = n-1 \end{cases} \tag{28}$$

6.2.3. Expectation of Overclocking Error. Since the carry chain of a CCM will not propagate over a_{n-1}, the upper bound of parameter t and m should be modified from (5) and (6) to give (29) and (30).

$$0 \leqslant t \leqslant n-b-1 \tag{29}$$

$$b < m \leqslant n-t \tag{30}$$

Finally, by substituting (28) and (25) with modified bounds of t and m into (11), we obtain the expectation of overclocking error for a CCM to be given by (31).

$$E_O = \begin{cases} 2^{s-b-1} - 2^{s-n-1}, & \text{if } b \leq n-1 \\ 0, & \text{otherwise} \end{cases} \tag{31}$$

6.3. CCM with Multiple RCAs and Shifters

In the case where a CCM is composed of two shifters and one RCA, such as operation $B = 20A = (A << 2)+(A << 4)$, let the shifted bits be denoted as s_1 and s_2 respectively. Hence the equivalent s in (31) can be obtained through (32).

$$s = |s_1 - s_2| \tag{32}$$

For those operations such as $B = 37A = (A << 5) + (A << 2) + (A << 1)$, the CCM can be built using a tree structure. Each root node is the baseline CCM and the errors are propagated through an adder tree, of which the error can be determined based on our previous RCA model.

7. TEST PLATFORM

In our experiments, we compare two design perspectives. In the first scenario, the word-length of the input signal is truncated before propagating through the datapath in order to meet a given latency. In our proposed overclocking scenario, the circuit is overclocked while keeping the original operand word-length. The benefits of the proposed methodology are demonstrated over a set of DSP example designs, which are implemented on the Xilinx ML605 board with a Virtex-6 FPGA.

7.1. Experimental Setup

We initially build up a test framework on an FPGA. The general architecture is depicted in Fig. 8. The main body of the test framework consists of the circuit under test (CUT), the test frequency generator and the control logic, as shown in the dotted box in Fig. 8. The I/Os of the CUT are registered by the launch registers (LRs) and the sample registers (SRs), which are all triggered by the test clock. Input test vectors are stored in the on-chip memory during initialization. The results are sampled using Xilinx ChipScope. Finally, we perform an offline comparison of the output of the original circuit at the rated frequency with the output of the overclocked as well as the truncated designs using the same input vectors.

The test frequency generator is implemented using two cascaded mixed-mode clock managers (MMCMs), created using Xilinx Core Generator [Xilinx a]. Besides the outputs, the corresponding input vectors and memory addresses are also recorded into the comparator, as can be seen in Fig. 8, in order to ensure that the recorded errors arise from overclocking the CUT rather than the surrounding circuitry when high test frequencies are applied.

7.2. Benchmark Circuits

Three types of DSP designs are tested: digital filters (FIR, IIR and Butterworth), a Sobel edge detector and a direct implementation of a Discrete Cosine Transformation (DCT). The filter parameters are generated through MATLAB filter design toolbox, and they are normalized to integers for implementation. Table I summarizes the operating frequency of each implemented design in Xilinx ISE14.1 when the word-length of input signal is 8-bit.

The input data are generated from two sources. One is called “uniform independent inputs”, which are randomly sampled from a uniform distribution of 8-bit numbers.

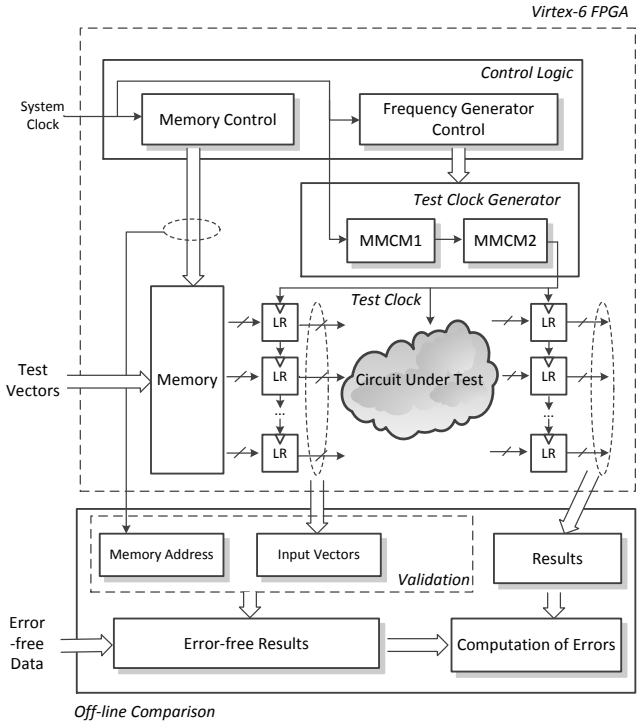


Fig. 8. Test framework, which is composed of a measurement architecture (the dotted box) on an FPGA and an off-line comparator using software. The error-free data are obtained by either pre-computation or initial run with low frequencies.

Table I. Rated Frequencies of example Designs.

Design	Frequency (MHz)	Description
FIR Filter	126.2	5^{th} order
Sobel Edge Detector	196.7	3×3
IIR Filter	140.3	7^{th} order
Butterworth Filter	117.1	9^{th} order
DCT	176.7	4-point

The other is referred to as “real inputs”, which denote 8-bit pixel values of the 512×512 Lena image.

7.3. Correcting for Conservative Timing Margin

Generally, the operating frequency provided by EDA tools tends to be conservative to ensure the correct functionality under a wide range of operating environments and workloads. In a practical situation, this may result in a large gap between the predicted frequency and the actual frequency under which the correct operation is maintained [Gojman et al. 2013].

For example, the predicted frequencies and the actual frequencies of a 5^{th} order FIR filter using different word-lengths are depicted in Fig. 9. The “actual” maximum frequencies are computed by increasing the operating frequency from the rated value until errors are observed at the output; the maximum operating frequency with correct output is recorded for the current word-length. As can be seen in Fig. 9, the circuit can operate without errors at a much higher frequency in practice than predicted according

to our experiments. A maximum speed differential of $3.2\times$ is obtained when the input signal is 5-bit.

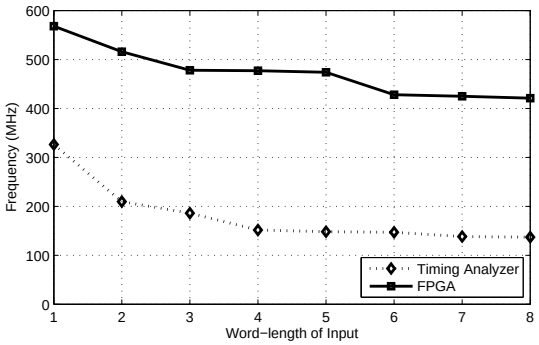


Fig. 9. The maximum operating frequencies for different input word-lengths of an FIR filter. The dotted line depicts the rated frequency reported by the timing analysis tool. The solid line is obtained through real FPGA tests using our platform.

In our experiments in Section 8, the conservative timing margin is removed in the traditional scenario for a fairer comparison to the overclocking scenario. To do this, for each truncated word-length, we select the maximum frequency at which we see no overclocking error on the FPGA board in our lab. For example, in Fig. 9, the operating frequency of the design when the word-lengths are truncated to 8, 5 and 2 bits are 400MHz, 450MHz and 500MHz respectively.

Fig. 9 also demonstrates that when the circuit is truncated, it allows the circuit to operate at a higher frequency than the frequency of full precision implementation. However, a non-uniform period change can be observed for both results. For instance, the maximum operating frequency keeps almost constant when the operand word-length reduces from 8 to 6 or from 5 to 3 in both the experimental results and those of timing analyzer. This will cause a slight deviation between our analytical model which assumes that the single bit carry propagation delay to be a constant value, as discussed in (13) with expression $n = b - 1$. This deviation will be influenced by many factors such as how the architecture has been packed onto LUTs and CLBs. In addition, process variation might cause non-uniform interconnection delays [Wong et al. 2005]. However, we shall see that our model remains close to the true empirical results in Section 8.

7.4. Computing Model Parameters

The accuracy of our proposed models is examined with practical results on Virtex-6 FPGA. We first determine the model parameters. There are two types of parameters in the models of overclocking error. The first is based on the circuit architecture. For example, the word-length of RCAs and CCMs (n), the shifted bits of the shifters in CCM (s), and the word-length of the input signal (k). This is determined through static analysis. The second depends on timing information, such as the single bit carry propagation delay μ_c. In order to keep consistency with the assumption made in models that μ_c is a fixed value, it is obtained according to the actual FPGA measurement results.

Initially the maximum error-free frequency f_0 is applied. In this case we have (33) where d_c is a constant value which denotes the interconnection delay, and $d_0 = 1/f_0$. The frequency is then increased such that (34) is obtained. This process repeats until the maximum frequency f_{n-1} is applied in (35). Based on these frequency values, μ_c

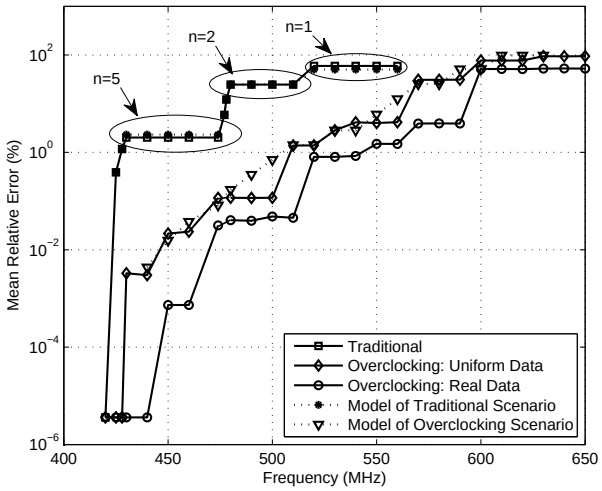


Fig. 10. A demonstration of two design perspectives with a 5^{th} order FIR filter, which is implemented on Virtex-6 FPGA. The modeled values of both overclocking errors and truncation errors are presented as dotted lines. The actual FPGA measurements are depicted using solid lines. Two types of inputs are employed in the overclocking scenario: the uniformly distributed data and the real image data from Lena.

can be determined.

$$d_0 = n\mu + d_c \tag{33}$$

$$d_1 = (n-1)\mu + d_c \tag{34}$$

$$\cdots$$

$$d_{n-1} = \mu + d_c \tag{35}$$

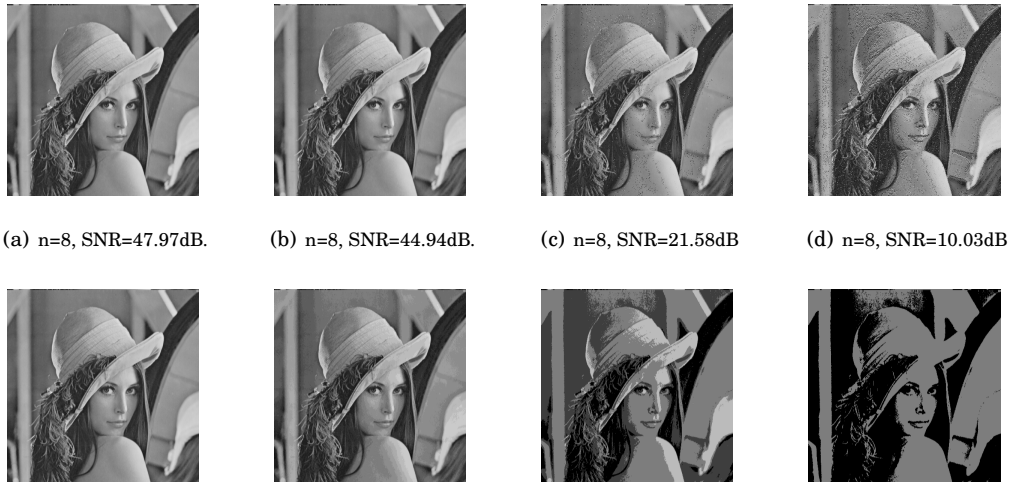
8. RESULTS AND DISCUSSION

8.1. Case study: FIR filter

We first assess the accuracy of our proposed models of errors. We record the value of MRE based on (23) in a 5^{th} order FIR with respect to different operating frequencies. The modeled values of both overclocking error and truncation error of the FIR filter are presented in Fig. 10 (dotted lines), as well as the actual measurements on the FPGA (solid lines) with two types of input data. The results demonstrate that our models match well with the practical results obtained using the uniform independent inputs.

According to Fig. 10, output errors are reduced in the overclocking scenario for both input types in comparison to the traditional scenario, as the analytical model validates. In addition, we see that using real data, more significant reduction of MRE are achieved, and that no errors are observed when frequency is initially increased. This is because for real data, long carry chains are typically generated with even smaller probabilities, and the longest carry chain rarely occurs.

The output images of the FIR filter for both of the two scenarios with increasing frequencies are presented in Fig. 11, from which we can clearly see the differences between the errors generated in these two scenarios. In the overclocking scenario, we observe errors in the MSBs for certain input patterns. This leads to “salt and pepper noise”, as shown on the images in the top row of Fig. 11. In the traditional scenario, truncation causes an overall degradation of the whole image, as can be seen in the bottom row of Fig. 11. Furthermore, it is worth noting that recovering from the latter type of error is more difficult, since it is generated due to precision loss.



(a) n=8, SNR=47.97dB. (b) n=8, SNR=44.94dB. (c) n=8, SNR=21.58dB (d) n=8, SNR=10.03dB

(e) n=7, SNR=26.02dB (f) n=5, SNR=21.76dB (g) n=2, SNR=6.57dB (h) n=1, SNR=3.95dB

Fig. 11. Output images of the FIR filter for both overclocking scenario (top row) and traditional scenario (bottom row) when operating frequencies are $425MHz$, $450MHz$, $480MHz$ and $520MHz$ respectively (from left to right).

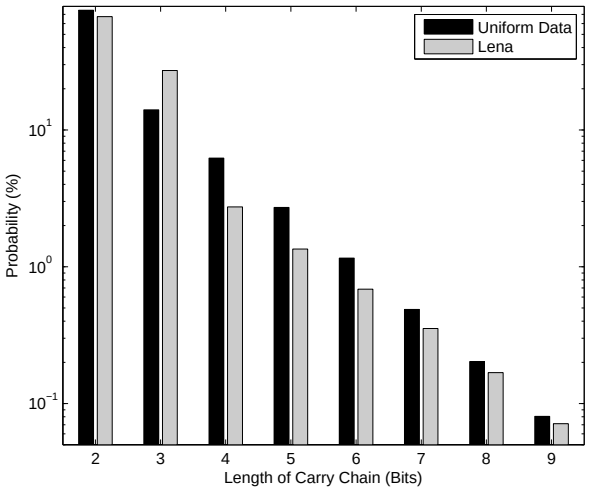


Fig. 12. Probability distribution of different length of carry chains in a 8-bit RCA. For the uniform data, two inputs of RCA are randomly sampled from a uniform distribution. For the image data Lena, one input of RCA uses the original data and the other uses the delayed original data for several clock cycles.

In addition, we record the probability distribution of different length of carry chains using an 8-bit RCA with both data types, as shown in Fig. 12. As the input data is 8-bit, the longest possible carry chain is of length 9-bit. It can be observed that when using Lena data, the probability is only higher for carry chain with a length of 3-bit. While in other situations, using uniform data leads to higher probability, especially for long carry chain length. This finding explicitly demonstrates that longer carry chains happens with an even smaller probability when utilizing real data.

Table II. Relative Reduction of MRE (%) in Overclocking Scenario for Various Normalized Frequencies Based on (36) for Two Types of Input Data: the Uniform Data (Uni) and Real Image Data from Lena (Lena).

Normalized	**FIR**		**Sobel**		**IIR**		**Butterworth**		**DCT4**		**Geo.Mean**	
Frequency	Uni	Lena	Uni	Lena	Uni	Lena	Uni	Lena	Uni	Lena	Uni	Lena
1.04	99.85	100.00	99.51	99.74	72.28	90.09	79.03	100.00	83.58	98.06	86.14	97.50
1.08	98.93	99.97	96.26	93.75	71.64	90.50	78.81	100.00	83.26	98.45	85.15	96.46
1.12	94.27	98.82	96.25	93.62	73.63	88.25	81.88	84.87	89.44	99.56	86.68	92.84
1.16	99.66	99.91	73.73	93.62	73.10	89.92	79.30	84.23	79.07	99.44	80.44	93.23
1.20	96.03	99.90	81.55	81.52	70.76	75.67	64.96	84.50	N/A*	N/A*	77.46	84.95
1.24	98.46	99.32	81.43	81.67	70.47	76.12	63.66	84.23	N/A*	N/A*	77.44	84.92
1.28	95.39	99.29	60.41	78.24	N/A*	N/A*	54.38	75.15	N/A*	N/A*	67.92	83.58
1.32	95.37	98.75	N/A*	N/A*	N/A*	N/A*	N/A*	N/A*	N/A*	N/A*	95.37	98.75

* Current frequency cannot be achieved in the traditional scenario. These points are excluded from the calculation of geometric means.

Table III. Frequency Speedups (%) in Overclocking Scenario Under Various Error Budgets for Two Types of Input Data: Uniform Data (Uni) and Real Image Data from Lena (Lena).

Error	**FIR**		**Sobel**		**IIR**		**Butterworth**		**DCT4**		**Geo.Mean**	
Budget(%)	Uni	Lena	Uni	Lena	Uni	Lena	Uni	Lena	Uni	Lena	Uni	Lena
0.05	4.76	21.43	6.82	6.82	0.95	1.26	12.40	24.03	0.72	0.96	3.07	5.32
0.5	19.05	21.43	13.64	6.82	0.63	10.06	24.03	24.03	0.48	12.44	4.52	13.45
1	19.05	28.57	13.64	18.18	10.06	16.35	24.03	24.03	0.48	12.44	7.86	19.10
5	19.15	25.53	18.18	18.18	0.54	0.82	0.63	0.94	7.66	12.44	3.91	5.36
10	19.15	25.53	10.64	10.64	0.54	1.09	6.92	13.21	4.91	4.911	5.19	7.19
20	19.15	25.53	5.77	15.39	8.70	8.70	3.26	3.26	6.70	10.88	7.32	10.39
50	15.69	15.69	10.53	19.30	42.50	50.00	46.74	54.89	15.06	19.25	21.8	27.59

8.2. Potential Benefits in Datapath Design

As we mentioned in Section 5, our results could be of interest to a circuit designer in two ways. Typically, either the designer will want to create a circuit that can run at a given frequency with the minimum possible MRE, or the algorithm designer will wish to run as fast as possible whilst maintaining a specific error tolerance. For the first design target, the experimental results for all five example designs on FPGA are summarized in Table II in terms of the relative reduction of MRE as given in (36) where MRE_{Trad} and MRE_{ovrc} denote the value obtained in the traditional scenario and in the overclocking scenario, respectively.

$$\frac{MRE_{Trad} - MRE_{ovrc}}{MRE_{Trad}} \times 100\% \tag{36}$$

In this table, the frequency is normalized to the maximum error-free frequency for each design when the input signal is 8-bit. The N/A in Table II refers to the situations where a certain frequency simply cannot be achieved using the traditional scenario. It can be seen that a significant reduction of MRE can be achieved using the proposed overclocking scenario, and the geometric mean reduction varies from 67.9% to 95.4% using uniform input data. Even larger differences of MRE can be observed when testing with real image data for each design, ranging from 83.6% to 98.8%, as expected given the results shown in Fig. 10.

Table III illustrates the frequency speedups for each design when the specified error tolerance varies from 0.05% to 50%. For all designs, we see that the overclocking scenario still outperforms the traditional scenario for each MRE budget in terms of operating frequency. Likewise, the frequency speedup is higher for real image inputs than uniform inputs. The geometric mean of frequency speedups of 3.1% to 21.8% can be achieved by using uniform data, while 5.3% to 27.6% when using real image data.

Table IV. Area Overhead of Our Approach with respect to Given Frequency Requirements

Normalized	**FIR**		**Sobel**		**IIR**		**Butterworth**		**DCT4**		**Geo.Mean**	
Frequency	LUTs	Slices	LUTs	Slices	LUTs	Slices	LUTs	Slices	LUTs	Slices	LUTs	Slices
1.04	1.24	1.05	1.12	1.22	1.26	1.42	1.16	1.11	1.13	1.00	1.18	1.15
1.08	1.24	1.05	1.03	1.08	1.26	1.42	1.16	1.11	1.11	1.15	1.16	1.16
1.12	1.24	1.05	1.03	1.08	1.26	1.42	1.37	1.54	2.30	2.17	1.38	1.40
1.16	2.95	2.56	1.03	1.08	1.61	1.71	1.99	2.03	4.60	2.89	2.14	1.95
1.20	2.95	2.56	1.66	1.39	2.77	3.48	3.17	3.50	N/A*	N/A*	2.56	2.57
1.24	4.42	3.15	4.00	3.55	7.87	6.73	6.92	5.25	N/A*	N/A*	5.57	4.46
1.28	4.42	3.15	4.00	3.55	N/A*	N/A*	6.92	5.25	N/A*	N/A*	4.97	3.89
1.32	4.42	3.15	N/A*	N/A*	N/A*	N/A*	N/A*	N/A*	N/A*	N/A*	4.42	3.15

* Current frequency cannot be achieved in the traditional scenario. These points are excluded from the calculation of geometric means.

Table V. Area Overhead of Our Approach with respect to Given Error Budgets.

Error	**FIR**		**Sobel**		**IIR**		**Butterworth**		**DCT4**		**Geo.Mean**	
Budget(%)	LUTs	Slices	LUTs	Slices	LUTs	Slices	LUTs	Slices	LUTs	Slices	LUTs	Slices
0.05	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
0.5	1.02	1.15	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.03
1	1.02	1.15	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.03
5	1.24	1.05	1.00	1.00	1.26	1.42	1.65	1.11	1.00	1.00	1.13	1.11
10	1.24	1.05	1.12	1.22	1.26	1.42	1.65	1.11	1.11	1.15	1.18	1.18
20	1.24	1.05	1.03	1.08	1.26	1.42	6.92	5.25	3.45	2.17	2.07	1.79
50	2.95	2.56	4.00	3.55	7.87	6.73	6.92	5.25	3.45	2.12	4.67	3.70

8.3. Area Overhead of Our Approach

For both aforementioned design goals, it should be noted that an extra benefit of using the traditional approach is that reducing datapath precision would potentially lead to a smaller design, in comparison to our proposed overclocking scenario. Consequently, the area overheads of our approach are summarized in Table IV and Table V for both design goals. In our experiments, the area overhead is evaluated in terms of the number of LUTs and Slices in the FPGA technology, while similar metrics can also be employed if applying our approach into other hardware platforms. It can be seen that for a given frequency requirement with minimum possible errors, the geometric mean of area overhead ranges from $1.16\times$ and $1.15\times$ to $5.57\times$ and $4.46\times$ for LUTs and Slices, respectively. This result illustrates that although using traditional approach would lead to smaller design, the area benefits are less attractive than the performance or accuracy benefits brought by utilizing the our proposed new approach, especially when the clock frequency is initially increased.

An even smaller area overhead can be found for a specified error budget with the maximum clock frequencies. As seen in Table IV, the geometric mean of area overhead in terms of LUTs and Slices is $1.00\times \sim 4.67\times$ and $1.00\times \sim 3.70\times$ respectively. Notice that an area overhead of $1\times$ means both the overclocking scenario and the traditional scenario use the original precision, however the former achieves frequency speedups as seen in Table V, because a specific error budget can be tolerated.

In general, a very small area overhead of our approach can be observed when the design is initially operated beyond the safe region with higher frequencies or with a small error budget (i.e. $\sim 10\%$). In addition at even higher operating frequencies or larger error specifications, our approach achieves large performance or accuracy improvements at a cost of a relatively small extra area.

9. CONCLUSION

In this paper, we have explored the probabilistic behavior of key arithmetic primitives in an FPGA when operating beyond the deterministic region. Two design scenarios

that might generate errors: violating timing constraints and reducing precisions, have been investigated. It has been found that simply allowing timing violations to happen would potentially lead to either reductions in average errors or speedups in clock frequencies, in comparison to the traditional approach where the target timing is met by limiting the precisions throughout the datapath. To verify this hypothesis, probabilistic models have been developed for errors generated from both design methodologies, and empirical results on a Virtex-6 FPGA have also been obtained over a set of DSP applications. Besides, the silicon area has also been taken into account for the tradeoffs. We have shown that applying our approach on basic arithmetic primitives would potentially outperform utilizing the traditional approach on either the same architectures or structures specifically designed for high performance, of which the precision loss is even greater under a tight area budget.

While this research has shown a promising direction for sustaining performance scaling, in the future we wish to apply this approach onto arithmetic units based on alternative number representations, and explore the probabilistic behavior of these operators under a variety of process, voltage and temperature variations.

REFERENCES

Altera. 2008. Cyclone Device Handbook. (2008).

Semiconductor Industry Association. 2007. *International technology roadmap for semiconductors (ITRS)*.

T. Austin, V. Bertacco, D. Blaauw, and T. Mudge. 2005. Opportunities and challenges for better than worst-case design. *Proc. ASP-Design Automation Conf.* (2005), 2–7.

T. Austin, D. Blaauw, T. Mudge, and K. Flautner. 2004. Making typical silicon matter with razor. *IEEE Trans. on Computer* 37, 3 (2004), 57–65.

D. Boland and G. A. Constantinides. 2008. An FPGA-based implementation of the MINRES algorithm. In *Proc. Int. Conf. Field Programmable Logic and Applications*. IEEE, 379–384.

D. Boland and G. A. Constantinides. 2011. Bounding variable values and round-off effects using handelman representations. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* 30, 11 (2011), 1691–1704.

B. Colwell. 2004. We may need a new box. *IEEE Trans. on Computer* 37, 3 (2004), 40–41.

G. A. Constantinides, N. Nicolici, and A. B. Kinsman. 2011. Numerical Data Representations for FPGA-Based Scientific Computing. *IEEE Design Test of Computers* 28, 4 (2011), 8–17.

F. De, H. Luiz, and J. Stolfi. 2004. Affine arithmetic: concepts and applications. *Numerical Algorithms* 37, 1-4 (2004), 147–158.

D. Ernst, N.S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and others. 2003. Razor: A low-power pipeline based on circuit-level timing speculation. In *Proc. Int. Symp. on Microarchitecture*. 7–18.

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

B. Gojman, S. Nalmela, N. Mehta, N. Howarth, and A. DeHon. 2013. GROK-LAB: generating real on-chip knowledge for intra-cluster delays using timing extraction. In *Proc. Int. Symp. on Field Programmable Gate Arrays*. 81–90.

V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy. 2013. Low-Power Digital Signal Processing Using Approximate Adders. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* 32, 1 (2013), 124–137.

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

K. Keutzer and M. Orshansky. 2002. From blind certainty to informed uncertainty. In *Proc. Int. workshop on Timing Issues in the Specification and Synthesis of Digital Systems*. 37–41.

A. B. Kinsman and N. Nicolici. 2010. Bit-width allocation for hardware accelerators for scientific computing using SAT-modulo theory. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* 29, 3 (2010), 405–413.

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

S. L. Lu. 2004. Speeding up processing with approximation circuits. *IEEE Trans. on Computer* 37, 3 (2004), 67–73.

R. E. Moore. 1966. *Interval analysis*. Vol. 60. Prentice-Hall Englewood Cliffs, NJ.

J. M. Rabaey, A. P. Chandrakasan, and B. Nikolic. 2003. *Digital integrated circuits: a design perspective (2nd edition)*. Prentice-Hall.

A. Roldao-Lopes, A. Shahzad, G. A. Constantinides, and E. C. Kerrigan. 2009. More flops or more precision? accuracy parameterizable linear equation solvers for model predictive control. In *Proc. Int. Symp. Field Programmable Custom Computing Machines*. 209–216.

A. Sampson, W. Dietl, E. Fortuna, D. Gnanapragasam, L. Ceze, and D. Grossman. 2011. EnerJ: Approximate data types for safe and general low-power computation. In *Proc. ACM SIGPLAN Notices*, Vol. 46. 164–174.

W. Sung and K. Kum. 1995. Simulation-based word-length optimization method for fixed-point digital signal processing systems. *IEEE Trans. on Signal Processing* 43, 12 (1995), 3087–3090.

A. K. Uht. 2004. Going beyond worst-case specs with TEAtime. *IEEE Trans. on Computer* 37, 3 (2004), 51–56.

K. Underwood. 2004. FPGAs vs. CPUs: trends in peak floating-point performance. In *Proc. Int. Symp. Field Programmable Gate Arrays*. 171–180.

H. Y. Wong, L. Cheng, Y. Lin, and L. He. 2005. FPGA device and architecture evaluation considering process variations. In *Proc. Int. Conf. on Computer-Aided Design*. 19–24.

Xilinx. *Virtex-6 FPGA Clocking Resources User Guide*.

Xilinx. *Virtex-6 FPGA Configurable Logic Block User Guide*.

Online Appendix to: Imprecise Datapath Design: An Overclocking Approach

KAN SHI, DAVID BOLAND
and GEORGE A. CONSTANTINIDES, Imperial College London

DOI: http://dx.doi.org/10.1145/0000000.0000000