Imprecise Computing in Datapath Design: Precision Loss or Timing Violations

Kan Shi, David Boland, *Member, IEEE,* and George A. Constantinides, *Senior Member, IEEE*

***Abstract*—The abstract.**

***Index Terms*—**

I. Introduction

THIS demo file is intended to serve as a "starter file" for IEEE journal papers produced under LaTeX using IEEEtran.cls version 1.8 and later. I wish you the best of success.

mds
December 27, 2012

A. Subsection Heading Here

Subsection text here.

1) Subsubsection Heading Here: Subsubsection text here.

II. Ripple Carry Adder

A. Adder Structures in FPGAs

Adders serve as a key building block for arithmetic operations. Generally speaking, 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 in Section IV by discussing the CCM that is commonly used in DSP applications and numerical algorithms.

Typically the maximum frequency of a RCA is determined by the longest carry propagation. Consequently, modern FPGAs offer built-in architectures for very fast ripple carry addition. For instance, the Altera Cyclone series uses fast tables [1] while the Xilinx Virtex series employs dedicated multiplexers and encoders for the fast carry logic [2]. Figure 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 generation of each LUT with increasing operand word-lengths. For our initial analysis, we assume that the carry propagation delay of each FA is a constant value μ, which is a combination of logic delay and routing delay, and hence the critical path delay of the RCA is $\mu_{RCA} = n\mu$, as shown in Figure 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, $T_S < \mu_{RCA}$, intermediate results will be sampled, potentially generating errors.

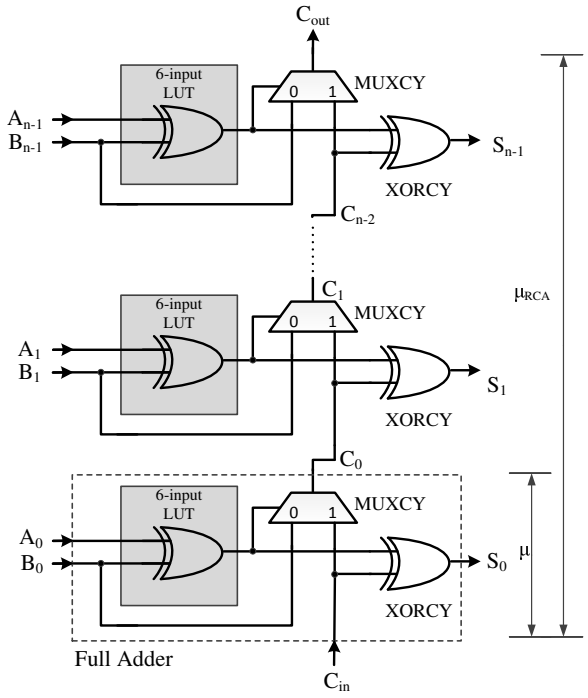


Fig. 1. An n-bit ripple carry adder in Virtex-6 FPGA.

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 results in truncation or roundoff error. In our proposed new scenario, circuits are implemented with greater word-length, but are clocked beyond the safe region so that timing violations sometimes occur. This process generates "overclocking error".

B. 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 $[-1, 1)$. For our initial analysis, we assume every bit of each input is uniformly and independently generated. However, this assumption will be relaxed in Section VI where the predictions

K. Shi, D. Boland and G. A. Constantinides are with the Department of Electrical and Electronic Engineering, Imperial College London, London, SW7 2BT, U.K. (email: {k.shi11, david.boland03, g.constantinides}@imperial.ac.uk)

Manuscript received April 19, 2005; revised December 27, 2012.

are verified using real image data. The errors at the output are evaluated in terms of the absolute value and the probability of their occurring. These two metrics are combined as the error expectation.

If the input signal of a circuit is 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 (1).

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

Since we assume there are two mutually independent inputs to the RCA, the overall expectation of truncation error for the RCA is given by (2).

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

C. Probabilistic Model of Overclocking Error

1) Generation of Overclocking Error: For a given T_S, the maximum length of error-free carry propagation is described by (3), where f_S denotes the sampling frequency.

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

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 [3] to the following statements, which examine carry generation, propagation and annihilation, as well as the corresponding summation results of a single bit i, according to the relationship between its input patterns A_i and B_i:

- If $A_i = B_i = 1$, a new carry chain is generated at bit i, and $S_i = C_{i-1}$;
- If $A_i \neq B_i$, the carry propagates for this carry chain at bit i, and $S_i = 0$;
- If $A_i = B_i$, the current carry chain annihilates at bit i, and S_i=1.

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 (3). 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 (4) and (5):

$$0 \leq t \leq n - b \tag{4}$$

$$b < m \leq n + 1 - t \tag{5}$$

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 (6), where $\hat{S}_i$ and S_i denote the actual and error-free output of bit i respectively.

$$e_{tm} = \frac{\left|\sum_{i=t+b}^{n} (S_i - \hat{S}_i) \cdot 2^i\right|}{2^n} \tag{6}$$

S_i and $\hat{S}_i$ can be determined using the equations from the previous statements in Section II-C1. 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. Substituting these values into (6) yields (7). Interestingly, the value of overclocking error has no dependence on the length of carry chain m.

$$\begin{aligned} e_{tm} &= \frac{\left|2^{t+m-1} - 2^{t+m-2} - \cdots - 2^{t+b}\right|}{2^n} \\ &= 2^{t+b-n} \end{aligned} \tag{7}$$

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 (8).

$$P_{tm} = P_{(A_t = B_t = 1)} P_{(A_{t+m-1} = B_{t+m-1})} \cdot \prod_{i=t+1}^{t+m-2} P_{(A_i \neq B_i)} \tag{8}$$

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 (9). Note that (9) takes into account the carry annihilation always occurs when $t + m - 1 = n$.

$$P_{tm} = \begin{cases} (1/2)^{m+1} & \text{if } t + m - 1 < n \\ (1/2)^m & \text{if } t + m - 1 = n \end{cases} \tag{9}$$

4) Expectation of Overclocking Error: Expectation of overclocking error can be expressed by (10).

$$E_O = \sum_t \sum_m P_{tm} \cdot e_{tm} \tag{10}$$

Using P_{tm} and e_{tm} from (7) and (9) respectively, E_O can be obtained by (11).

$$E_O = \begin{cases} 2^{-b} - 2^{-n-1}, & \text{if } b \leq n \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

D. 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 (12).

$$E_{trad} = 2^{-b+1} - 2^{-k} \tag{12}$$

Overclocking errors are allowed to happen in the second scenario, therefore the word-length of RCA is set to be equal to the input word-length, that is, $n = k$. Hence we obtain (13) according to (11).

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

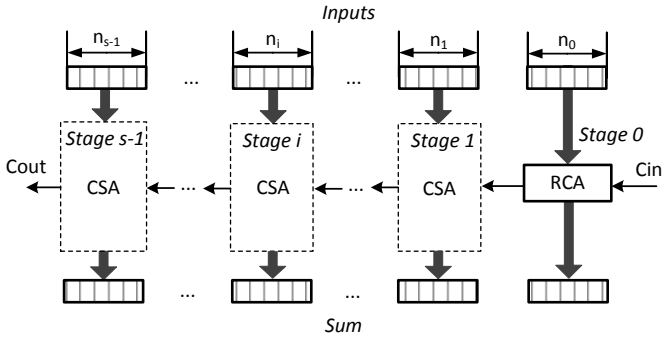
Comparing (13) and (12), we have (14). This equation indicates that by allowing timing violations, the overall error expectation of RCA outputs drops by a factor of 2 in comparison to traditional scenario. This provides the first hint that our approach is useful in practice.

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

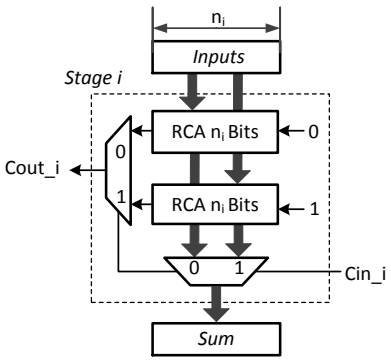
III. Carry Select Adder

A. Introduction

Since the delay of RCA is determined by the length of carry chain, the carry select adder (CSA) is designed to shorten the carry chain by overlapping carry propagation in sections such that the operating speed can be boosted. The structure of a CSA is presented in Fig. 2(a). In a CSA, the carry chain is divided into multiple stages. Each stage contains two RCAs and two multiplexers, as seen in Fig. 2(b). For a given input, two computations are performed simultaneously where the carry input is zero and one respectively. One of these two results is then selected according to the actual carry input. Although this structure brings timing benefits, it costs extra hardware resources compared to a standard RCA because the carry chain is duplicated. Furthermore, in FPGA technology, multiplexers are expensive. Due to this reason, we explore the trade-offs between silicon area, accuracy and performance of these two adder structures in this section.



(a) The structure diagram of a CSA with s stages.



(b) The structure of the i^{th} stage in the CSA.

Fig. 2. A comparison between RCA and CSA in terms of the maximum word-length of input signal and the area consumption.

B. 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 II-B.

In 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. In our analysis, we follow the assumption that the critical path delay is due to carry propagation and multiplexing the carry output. Note that unlike other stages, the least significant stage is only built by one RCA without multiplexers, since it is directly driven by the carry input. Hence we obtain the delay of the i^{th} stage as presented in (15), 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.

$$d_i = \begin{cases} n_i \cdot \mu_c + (s-i) \cdot \mu_{mux}, & \text{if } i \in [1, s-1] \\ n_0 \cdot \mu_c + (s-1) \cdot \mu_{mux}, & \text{if } i = 0 \end{cases} \tag{15}$$

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 (16).

$$d_{s-1} = d_{s-2} = \cdots = d_0 \tag{16}$$

In this case, combining (15) and (16) yields n_i, which is represented by the word-length of the most significant stage n_{s-1}, in (17).

$$n_i = \begin{cases} n_{s-1} - (s-1-i) \cdot \frac{\mu_{mux}}{\mu_c}, & \text{if } i \in [1, s-1] \\ n_{s-1} - (s-2) \cdot \frac{\mu_{mux}}{\mu_c}, & \text{if } i = 0 \end{cases} \tag{17}$$

Summing up n_i gives the total word-length of the CSA in (18).

$$\begin{aligned} n_{CSA} &= \sum_{i=0}^{s-1} n_i \\ &= s \cdot n_{s-1} - \frac{\mu_{mux}}{\mu_c} \cdot \frac{(s+1)(s-2)}{2} \end{aligned} \tag{18}$$

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 (3). Similarly for CSA, the word-length of each stage should be selected in order to satisfy (19).

$$d_i \leqslant \tfrac{1}{f_s} \quad , \quad \forall i \in [0, s-1] \tag{19}$$

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 (20),

$$\mu_c \cdot (b-1) = n_{s-1} \cdot \mu_c + \mu_{mux} \tag{20}$$

Substitute (20) into (18) to replace n_{s-1}, we derive the representation of the word-length of CSA in terms of a given timing constraint, as presented in (21).

$$n_{CSA} = s \cdot (b-1) - \frac{\mu_{mux}}{\mu_c} \cdot \frac{(s+2)(s-1)}{2} \tag{21}$$

C. Model Verification

The ratio μ_{mux}/μ_c can be computed from experiments. 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 (15) is recorded with respect to different word-lengths of this given stage (n_i). Then the total word-length of CSA can be predicted through (21). In addition, the maximum word-length of the 2-stage CSA is obtained experimentally by increasing

n_{CSA} until errors are observed at the output. The comparison between the modeled value and the empirical results of n_{CSA} is illustrated in Fig. (3). It can be seen that our model match well with the experimental results.

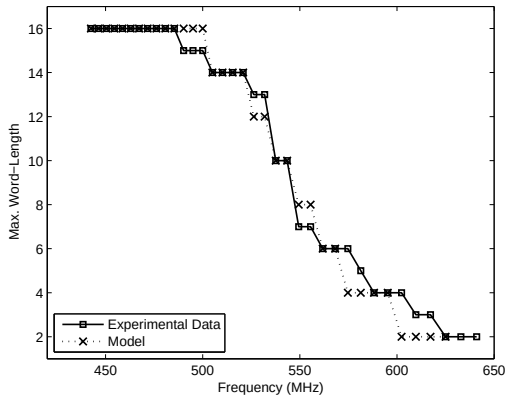


Fig. 3. Comparison of the maximum word-length between the modeled value and the experimental results from FPGA simulations.

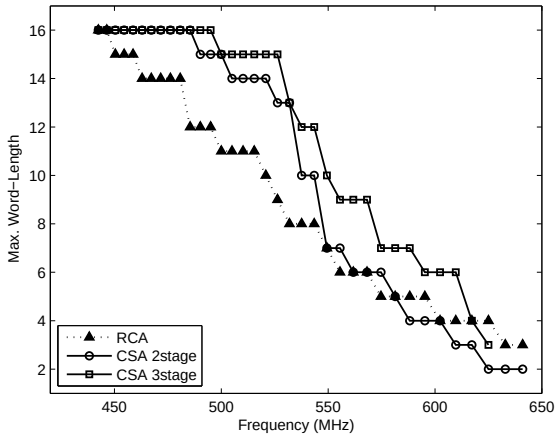
D. Accuracy Benefits and Area Overhead in CSA

As greater word-length corresponds to smaller truncation errors, we compare the maximum word-length of RCA and CSA with 2 stages and 3 stages over a range of operating frequencies, as presented in Fig. 4(a). It can be seen from Fig. 4(a) that in comparison to RCA, CSA achieves greater word-length when frequency is initially increased. RCA only outperforms than CSA when very high frequency is applied. This is because at low frequencies, although the multiplexer delay limits the word-length of each stage in CSA when compared to RCA, the stage parallelism in CSA enables a greater word-length. However when frequency increases, the multiplexer delay becomes comparable to the delay of the carry chain, and this inhibits the benefits of parallelism. In addition, we can see that the word-length of 3-stage CSA is always greater than 2-stage CSA across the entire frequency domain.

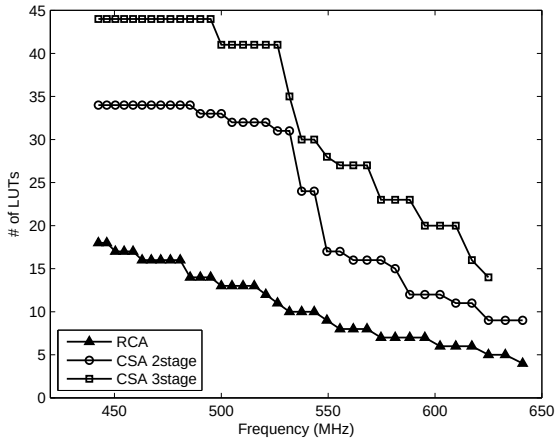
However, the accuracy benefits brought by CSA comes at the cost of a large area overhead. Fig. 4(b) depicts the number of Look-Up Tables (LUTs) in the FPGA used for all three structures. It can be seen that in order to meet a given frequency, the 3-stage CSA consumes $2.4\times \sim 3.7\times$ area than 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.

E. Exploring Trade-offs Between Accuracy, Performance and Area

In Section II, we have discussed two design scenarios when considering timing constraints. In this section, we expand our analysis by incorporating the 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 RCA with the maximum 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 targeting the optimum design methodology under each situation.



(a) Maximum word-length.



(b) Hardware resource usage.

Fig. 4. A comparison between RCA and CSA in terms of the maximum word-length of input signal and the area consumption.

First, suppose the designer would wish to create a circuit that can run at a given frequency with the minimum achievable output errors and resource usage. That is, a pair of $\{Area, Frequency\}$ constraint is applied. In this case, the optimum design method is selected based on the following criteria:

- Design with the best accuracy at outputs is the optimum design;
- If multiple designs achieve the same accuracy, then the design with minimum area is the optimum design;
- If the accuracy and area are identical for multiple designs,

they are all treated as the optimum design.

For instance let the available LUTs number be 25, the accuracy with respect to different operating frequencies for both design scenarios are presented in Fig. (5), on which the optimum design metric is labeled. Note that the accuracy is defined in terms of the error expectation at the output.

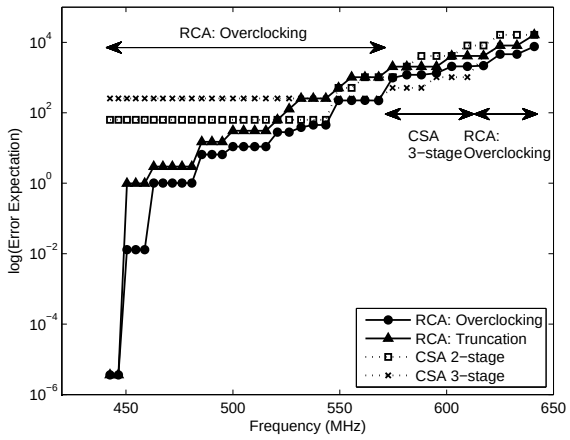


Fig. 5. A comparison between two design scenarios when the number of LUT is set to 25. The RCA and CSA are investigated in the conventional scenario, while the RCA is explored in the proposed new scenario. The results are obtained from post place-and-route simulations on Xilinx Virtex FPGAs.

We first notice that for all frequency values, the overclocked RCA achieves smaller error expectation than the RCA with truncated operand word-lengths, as predicted by the models in Section II-D. It can also be observed that CSA cannot be implemented with full word-length due to the limited area budget, and this leads to large truncation errors initially for CSA.

We then perform the similar experiments with a variety of area constraints. The optimum design method with respect to different operating frequencies and area consumptions is demonstrated in Fig. 6. From this figure, several observations can be made. If 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. 4(a). The 2-stage CSA is better than the 3-stage CSA when frequency is initially increased, as it consumes less area although both of them achieve the same error expectation. For a tighter area budget, only part of the CSA can be implemented, whereas the RCA still keeps full precision. In this case, area becomes the dominate factor and precision is lost for the CSA. We see the RCA with overclocking is the optimum design method across almost the whole frequency domain. For a more stringent area constraint, 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 CSA under the same area constraint.

Similarly 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. 7. In this situation, the error

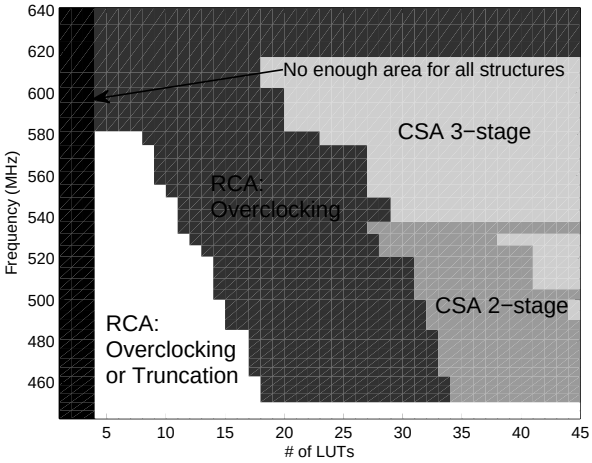


Fig. 6. A demonstration of the optimum design methodology which achieves the minimum error at outputs with respect to a variety of frequency and area constraints.

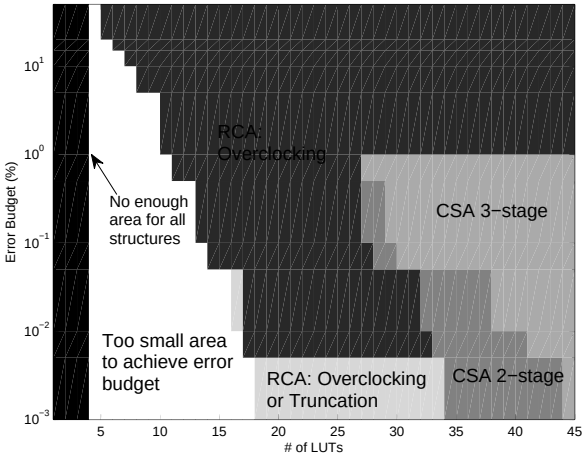


Fig. 7. A demonstration of the optimum design methodology which runs at the fastest frequency with respect to a variety of accuracy and area constraints.

specifications are evaluated in terms of the mean relative error (MRE), as presented in (32), where E_{error} and E_{out} refer to the mean value of error and the mean value of outputs, respectively.

$$MRE = \left| \frac{E_{error}}{E_{out}} \right| \times 100\% \tag{22}$$

In our experiments, MRE is set ranging from 0.001% to 50%. For a certain MRE, the design with the maximum operating frequencies 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. 7 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 with respect to large accessible area, 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 previous section, we see that the overclocked RCA achieves the fastest operating frequencies under most area constraints when the accuracy requirement is released.

To sum up, it can be seen that for both design goals, CSA is the best option only when there is enough hardware resources. However this circumstance is unlikely to happen, especially with the continued technology scaling nowadays. Due to this reason, in the following sections we will focus on the situation where the hardware resources are limited, and therefore RCA is the design option in our following analysis and experiments.

IV. 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 IV-C.

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 Figure 8.

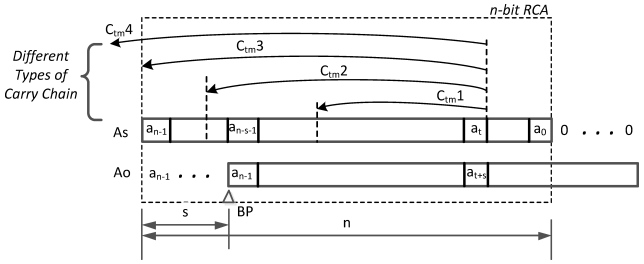


Fig. 8. Different types of carry chain in constant coefficient multiplier. The notion s denotes the shifted bits and BP denotes the binary point.

A. 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 (23), where *coe* denotes the coefficient value of the CCM, and E_{Tin} can be obtained according to (2).

$$E_{Tout} = |coe| \cdot E_{Tin} \tag{23}$$

B. Probabilistic Model of Overclocking Error

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, compared to RCA. Hence e_{tm} in CCM can be modified from (7) to give (24).

$$e_{tm} = 2^{t+b-n+s} \tag{24}$$

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 Figure 8, defined by the end region of the carry chain. For $C_{tm}1$, we have:

- Carry propagation: $a_i \neq a_{i+s}$ where $i \in [t+1, n-s-2]$;
- 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 (25) can be obtained by substituting this into (8).

$$P_{tm} = (1/2)^{m+1}, \quad \text{if } t+m-1 \leqslant n-2 \tag{25}$$

For carry annihilation of $C_{tm}3$, $a_{n-1} = a_{n-1}$, which is always true. Thus the probability of $C_{tm}3$ is given by (26).

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

$C_{tm}4$ 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 (27).

$$P_{tm} = \begin{cases} (1/2)^{m+1} & \text{if } t+m-1 < n-1 \\ (1/2)^{m} & \text{if } t+m-1 = n-1 \end{cases} \tag{27}$$

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 (4) and (5) to give (28) and (29).

$$0 \leqslant t \leqslant n-b-1 \tag{28}$$

$$b < m \leqslant n-t \tag{29}$$

Finally, by substituting (27) and (24) with modified bounds of t and m into (10), we obtain the expectation of overclocking error for a CCM to be given by (30).

$$E_O = \begin{cases} 2^{s-b-1} - 2^{s-n-1}, & \text{if } b \leq n-1 \\ 0, & \text{otherwise} \end{cases} \tag{30}$$

C. 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 (30) can be obtained through (31).

$$s = |s_1 - s_2| \tag{31}$$

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.

V. 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 XC6VLX240T-1FFG1156.

A. Experimental Setup

We initially build up a test framework on an FPGA. The general architecture is depicted in Fig. 9. 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. 9. 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.

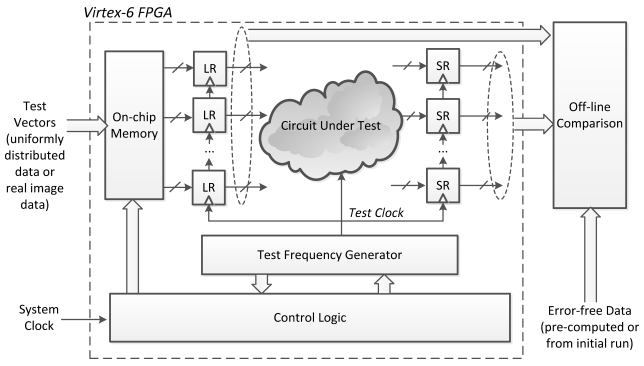


Fig. 9. FPGA Test framework, which is composed of a measurement architecture (the dotted box) and an off-line comparator.

The test frequency generator is implemented using two cascaded mixed-mode clock managers (MMCMs), created using Xilinx Core Generator [4]. Besides the outputs, the corresponding input vectors and memory addresses are also recorded into the comparator, as can be seen in Fig. 9, in order to ensure that the recorded errors arise from overclocking the CUT rather than the surrounding circuitry when high test frequencies are applied.

B. 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.

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

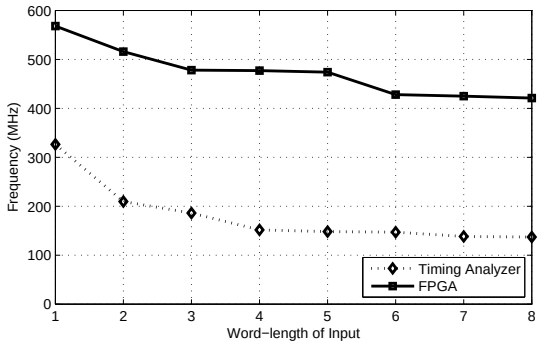


Fig. 10. The maximum operating frequencies for different input word-lengths of an FIR filter. The dotted line depicts the rated frequency reported by the Xilinx Timing Analyzer. The solid line is obtained through real FPGA tests using our platform.

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

C. Exploring the 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 [5].

For example, the predicted frequencies and the actual frequencies of a 5^{th} order FIR filter using different word-lengths are depicted in Fig. 10. 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. 10, 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.

In our experiments in Section VI, 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. 10, 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. 10 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 (12) with expression $n = b-1$. This deviation will be influenced by many factors including how the architecture has been packed onto LUTs and CLBs and process variation causing non-uniform interconnection delays [6]. However, we shall see that our model remains close to the true empirical results in Section VI.

D. Evaluation Metric of Outputs

The results are evaluated in terms of mean relative error (MRE), which represents the percentage of error at outputs. MRE is given by (32), where E_{error} and E_{out} refer to the mean value of error and the correct output respectively.

$$MRE = \left| \frac{E_{error}}{E_{out}} \right| \times 100\% \tag{32}$$

E. 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. In order to keep consistency with the assumption made in models that μ 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. 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 can be determined.

$$f_0 = n\mu + d_c \tag{33}$$

$$f_1 = (n-1)\mu + d_c \tag{34}$$

$$\dots$$

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

VI. Results and Discussion

A. Case study: FIR filter

We first assess the accuracy of our proposed models of error. The modeled values of both overclocking error and truncation error of the FIR filter are presented in Fig. 11 (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.

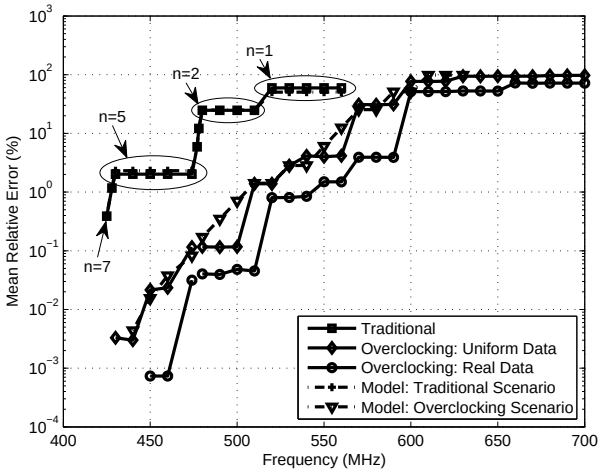


Fig. 11. A demonstration of two design perspectives with a 5^{th} order FIR filter, which is implemented on Virtex-6 FPGA. The modeled values of both overclocking errors and truncation errors are presented as dotted lines. The actual FPGA measurements are depicted using solid lines. Two types of inputs are employed in the overclocking scenario: the uniformly distributed data and the real image data from Lena.

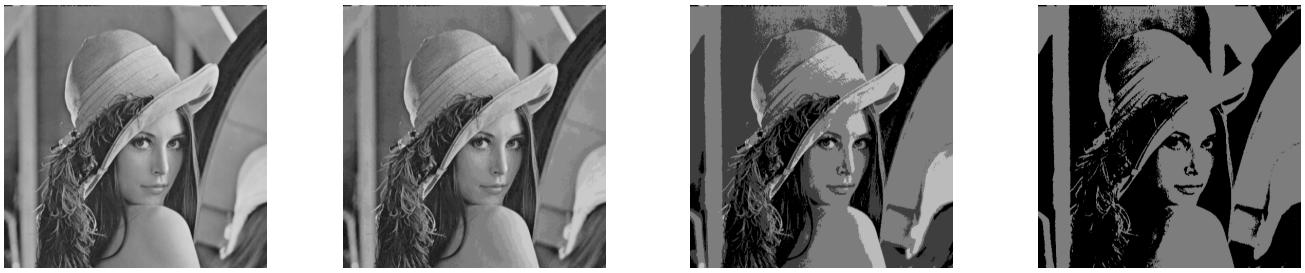
According to Fig. 11, output errors are reduced in the overclocking scenario for both input types in comparison to the traditional scenario, as expected by our models. In addition, we see that using real data, more significant reduction of MRE are achieved, and that no errors are observed when frequency 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. 12, 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. 12. In the traditional scenario, truncation causes an overall degradation of the whole image, as can be seen in the bottom row of Fig. 12. Furthermore, it is difficult to recover from the latter type of error, since it is generated due to precision loss.

B. Potential Benefits in Circuit Design

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

(a) 425MHz, n=8 , no errors observed (b) 430MHz, n=8, SNR=47.15dB (c) 480MHz, n=8, SNR=24.1dB (d) 520MHz, n=8, SNR=10.86dB



(e) 425MHz, n=7, SNR=26.06dB (f) 430MHz, n=5, SNR=24.85dB (g) 480MHz, n=2, SNR=6.57dB (h) 520MHz, n=1, SNR=3.95dB

Fig. 12. Output images of the FIR filter for both overclocking scenario (top row) and traditional scenario (bottom row) under various operating frequencies.

fast as possible whilst maintaining a specific error tolerance. In the first case, 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. 11.

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.

VII. Conclusion

The conclusion goes here.

Acknowledgment

The authors would like to acknowledge the support of the EPSRC (Grants EP/I020557/1 and EP/I012036/1).

References

[1] Altera, "Cyclone device handbook," 2008.
[2] Xilinx, "Virtex-6 FPGA configurable logic block user guide," 2009.
[3] J. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital integrated circuits: a design perspective (2nd edition)*. Prentice-Hall, 2003.
[4] Xilinx, "Virtex-6 FPGA clocking resources user guide," 2011.
[5] B. Gojman, S. Nalmela, N. Mehta, N. Howarth, and A. DeHon, "GROK-LAB: generating real on-chip knowledge for intra-cluster delays using timing extraction," in *Proc. Int. Symp. on Field Programmable Gate Arrays*, 2013, pp. 81–90.
[6] H. Wong, L. Cheng, Y. Lin, and L. He, "FPGA device and architecture evaluation considering process variations," in *Proc. Int. Conf. on Computer-Aided Design*, 2005, pp. 19–24.

PLACE PHOTO HERE

Author Biography text here.

Author Biography text here.

Author Biography text here.

TABLE II
RELATIVE REDUCTION OF MRE IN OVERCLOCKING SCENARIO FOR VARIOUS NORMALIZED FREQUENCIES BASED ON (36).

Normalized	**FIR**		**Sobel**		**IIR**		**Butterworth**		**DCT4**		**Geo.Mean**	
Frequency	Uniform	Lena	Uniform	Lena	Uniform	Lena	Uniform	Lena	Uniform	Lena	Uniform	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.

Error Budget	**FIR**		**Sobel**		**IIR**		**Butterworth**		**DCT4**		**Geo.Mean**	
%	Uniform	Lena	Uniform	Lena	Uniform	Lena	Uniform	Lena	Uniform	Lena	Uniform	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%