Title

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 III 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 [**?**] while the Xilinx Virtex series employs dedicated multiplexers and encoders for the fast carry logic [**?**]. 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.

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.

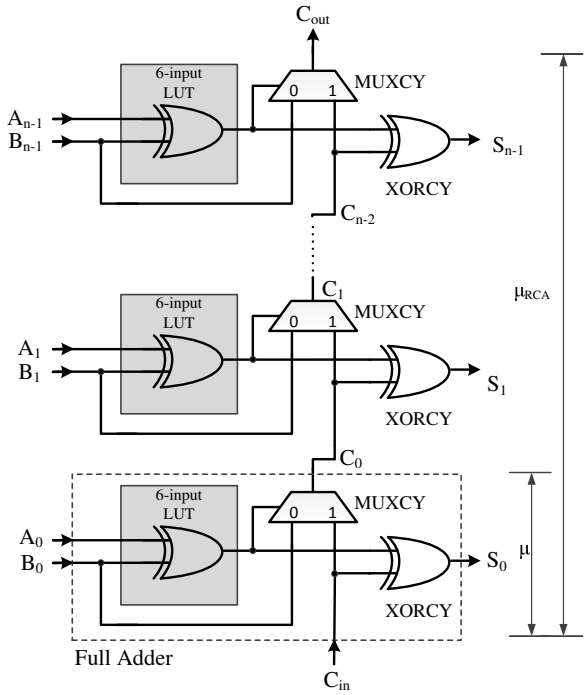


Fig. 1. An n-bit ripple carry adder in Virtex-6 FPGA.

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 **??** 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 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).

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

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 [**?**] 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.

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

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. 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 III-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 2.

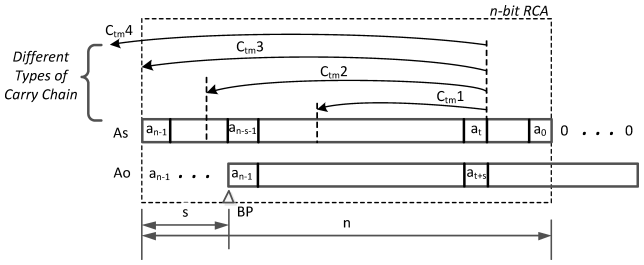


Fig. 2. Different types of carry chain in constant coefficient multiplier. The notion s denotes the shifted bits and BP denotes the binary point.

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

$$E_{Tout} = |coe| \cdot E_{Tin} \tag{15}$$

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 (16).

$$e_{tm} = 2^{t+b-n+s} \tag{16}$$

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 2, 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 (17) can be obtained by substituting this into (8).

$$P_{tm} = (1/2)^{m+1}, \quad \text{if } t+m-1 \leqslant n-2 \tag{17}$$

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 (18).

$$P_{tm} = (1/2)^{m}, \quad \text{if } t+m-1 = n-1 \tag{18}$$

$C_{tm}4$ 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 (19).

$$P_{tm} = \begin{cases} (1/2)^{m+1} & \text{if } t+m-1 < n-1 \\ (1/2)^{m} & \text{if } t+m-1 = n-1 \end{cases} \tag{19}$$

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 (20) and (21).

$$0 \leqslant t \leqslant n-b-1 \tag{20}$$

$$b < m \leqslant n-t \tag{21}$$

Finally, by substituting (19) and (16) with modified bounds of t and m into (10), we obtain the expectation of overclocking error for a CCM to be given by (22).

$$E_O = \begin{cases} 2^{s-b-1} - 2^{s-n-1}, & \text{if } b \leq n-1 \\ 0, & \text{otherwise} \end{cases} \tag{22}$$

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 (22) can be obtained through (23).

$$s = |s_1 - s_2| \tag{23}$$

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.

IV. Carry Select Adder

A. Introduction

Since the delay of RCA is determined by the length of carry chain, there are several alternative adder architectures proposed to shorten the carry chain. For instance, the carry select adder (CSA) is designed to overlap carry propagation in sections such that the operating speed can be boosted. In a CSA, the carry chain is divided into multiple stages. Each stage contains two RCAs and two multiplexers. For a given input, computations are performed twice simultaneously, with the carry input as zero and one respectively. The results are

then selected according to the actual carry input. Although this structure brings timing benefits, it costs extra hardware resources than RCA as the carry chain is duplicated and multiplexers are expensive especially in the FPGA technology. In this section, silicon area is incorporated as the third metric besides accuracy and performance. We explore the trade-offs between these three factors for both adder structures. The objective is to demonstrate the best design choice for a given set of design constraints.

B. Timing Models for Carry Select Adder

We initially describe the modelling method for the CSA timing, with the aim of forming the relationship between the operating frequency and the corresponding maximum word-length of CSA. This information can be employed to determine the truncation error based on the models presented in Section.xxx.

In a CSA with s stages ($s \geqslant 2$), let the stage delay be denoted by $d_0 \dots d_{s-1}$, where d_0 and d_{s-1} represent the delay of the most significant stage and the least significant stage, respectively. Unlike other stages, the least significant stage is built by one RCA only, since it is directly driven by the carry input. In our analysis, we follow the previous assumption that delay is due to carry propagation as well as, in this case, multiplexing the carry output. Therefore the delay of the i^{th} stage can be obtained through (24), where μ_c denotes the delay of 1-bit carry propagation and μ_{mux} denotes the delay of multiplexing.

$$d_i = \begin{cases} n_i \cdot \mu_c + (i+1) \cdot \mu_{mux}, & \text{if } i \in [0, s-2] \\ d_{i-1}, & \text{if } i = s-1 \end{cases} \tag{24}$$

Under the timing-driven design environment, the delay of each stage of CSA is set to be approximately uniform for the fastest operation. In this case we obtain (25).

$$d_0 = d_1 = \cdots = d_{s-1} \tag{25}$$

Substituting (24) into (25) yields (26), which denotes the CSA word-length of stage i.

$$n_i = \begin{cases} n_0 - i \cdot \frac{\mu_{mux}}{\mu_c}, & \text{if } i \in [0, s-2] \\ n_{i-1}, & \text{if } i = s-1 \end{cases} \tag{26}$$

The word-length of CSA is then given by (27).

$$n_{CSA} = \sum_{i=0}^{s-1} n_i = s \cdot n_0 - \frac{\mu_{mux}}{\mu_c} \cdot \frac{(s+1)(s-2)}{2} \tag{27}$$

In the conventional situation, the word-length of both RCA and CSA should be properly selected in order to meet timing. In this case, (28) can be obtained, where n_{RCA} is determined by current timing information through (xxx).

$$\mu_c \cdot n_{RCA} = \mu_c n_0 + \mu_{mux} \tag{28}$$

Based on (27) and (28), we may obtain the representation of the word-length of CSA in terms of a given timing constraint, as presented in (29).

$$n_{CSA} = s \cdot n_{RCA} - \frac{\mu_{mux}}{\mu_c} \cdot \frac{(s+2)(s-1)}{2} \tag{29}$$

It can be seen that $s = 1$ leads to $n_{CSA} = n_{RCA}$. This is because RCA forms the least significant stage of CSA. In addition, combine (26) and (28) to ensure $n_{s-1} > 0$, we obtain the upper bond of the stage number by (30).

$$s < n_{RCA} \cdot \frac{\mu_c}{\mu_{mux}} + 1 \tag{30}$$

C. Model Verification

As seen in (29), the value of μ_{mux}/μ_c should be determined before applying the timing model. This ratio can be obtained through timing analysis. Specifically, varying the word-length of a single stage n_i and recording the corresponding delay value d_i through the timing analysis tool. According to (24), the ratio can be obtained by fitting those values using linear polynomials.

Using this information, we verify our timing model with experimental results, which are obtained from post place and route simulations on Xilinx Virtex-6 FPGAs. Fig. 3 demonstrates both the modelled value and the experimental results of the maximum word-length of the 2-stage CSA for various operating frequencies. In this experiment, the input data are randomly sampled from a 16-bit data set, which follows uniform distribution. It can be seen that the the modelled value match well with the experimental results.

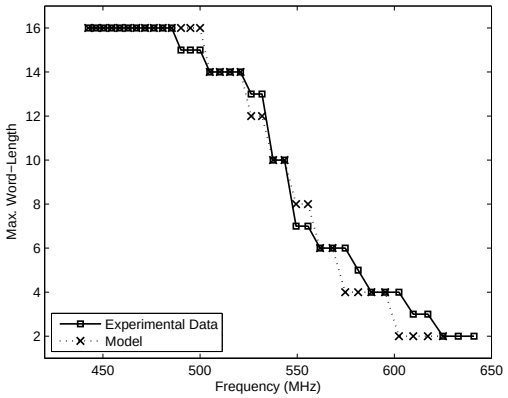
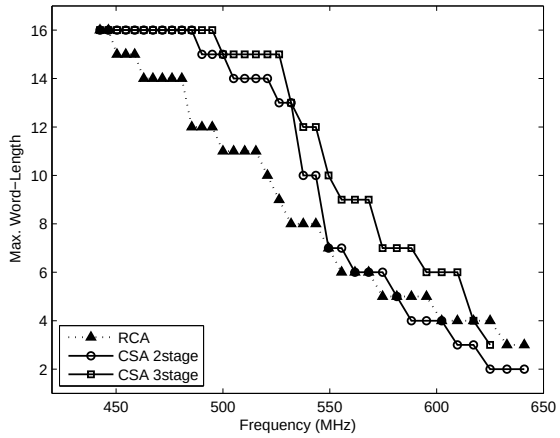


Fig. 3. Comparison of the maximum word-length for a given operating frequency between the modelled value and the experimental results from FPGA simulations.

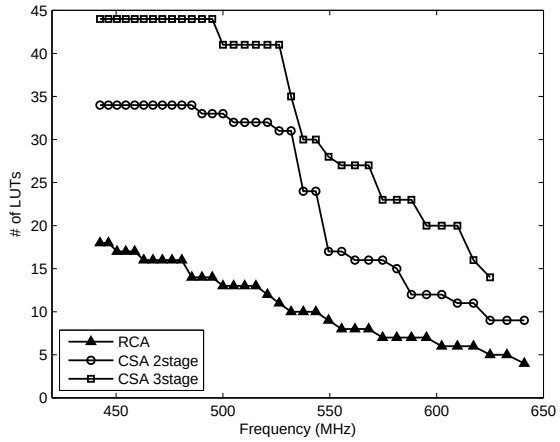
D. Area Overhead

Fig. 4(a) demonstrates the maximum word-lengths for RCA and CSA with 2 stages and 3 stages under a range of operating frequencies. We only investigate 3 stages as the maximum stages number predicted in (30) is 3. It can be seen that in comparison to RCA, CSA achieves greater word-length when frequency is initially increased. This corresponds to smaller truncation errors. RCA only outperforms than CSA when very high frequency is applied. In addition, the word-length of 3-stage CSA is always greater than 2-stage CSA across the entire frequency domain, as expected.

However, the accuracy benefits brought by CSA comes with the cost of large area overhead. Fig. 4(b) depicts the resource



(a) Maximum word-length.



(b) Hardware resource usage.

Fig. 4. Comparison between RCA and CSA across a variety of frequency values in terms of the maximum word-length of input signal and the area consumption.

usage (in terms of the number of Look-Up Tables (LUTs) in the FPGA) used for all three structures. It can be seen that for a given frequency, the 3-stage CSA consumes $2.4\times \sim 3.7\times$ area than RCA, while the number of the 2-stage CSA is $1.7\times \sim 3.1\times$. This finding poses a question of whether CSA with higher stage numbers still offer better accuracy than RCA under a limited area budget.

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

To answer this question, we explore the trade-offs between accuracy, performance as well as area in this section. If the available hardware resources are limited, the full word-length of both CSA and RCA might not be implemented. The precision lose might generates large errors even at low frequencies. This is demonstrated by experiments where area constraints are applied besides the timing requirement. Conventionally, both of the constraints are met by reducing the word-length of the input signal. The errors at the outputs are recorded with the reference to the original word-length, which is 16 bits in the following experiments. In addition, we propose another designing method where RCA is implemented with the maximum possible word-length under the given area budget, while the timing constraints are met by overclocking. For a certain pair of $\{Area, Frequency\}$ constraint, the optimum design method is selected based on the following criteria:

- Design with the minimum error expectation is the optimum design;
- If multiple designs achieve the same error expectation, then the design with minimum area is the optimum design;
- If the error expectation and area are identical for multiple designs, they are all treated as the optimum design.

The optimum design method for a variety of frequency and area constraints is demonstrated in Fig. 5, from which several observations can be made. If the available area is large enough, CSA serves as the optimum design option because it inherently operates faster than RCA, and it can be implemented with full precision in this situation. 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, whilst RCA still keeps full precision. In this case, area instead of timing becomes the dominate factor for small frequency values. This leads to precision loss for CSA initially. Therefore we see RCA with overclocking is the best design across almost the whole frequency domain. For an even stringent area constraint, the word-length of RCA is also limited. This results in truncation error initially for all design scenarios. Meanwhile, CSA with high stage numbers could not even be implemented at all.

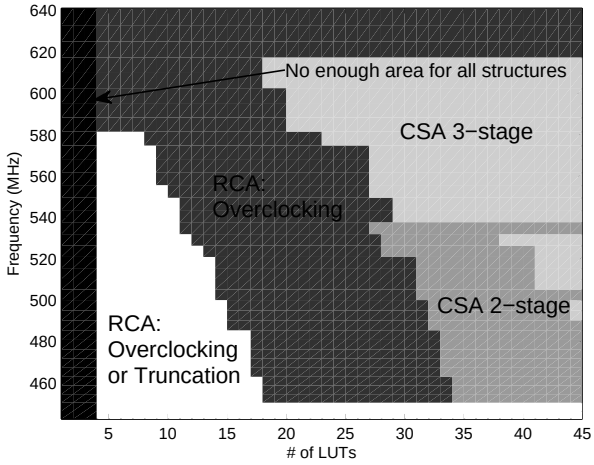


Fig. 5. Demonstration of the optimum design methodology for a variety frequency and area constraints.

V. Conclusion

The conclusion goes here.

Acknowledgment

The authors would like to thank...

REFERENCES

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.

PLACE PHOTO HERE

Author Biography text here.

Author Biography text here.

Author Biography text here.