Algorithm and Design of a Fully Parallel Approximate Coordinate Rotation Digital Computer (CORDIC)

Linbin Chen, *Student Member IEEE*, Jie Han, *Member IEEE*, Weiqiang Liu, Senior *Member IEEE* and Fabrizio Lombardi, *Fellow, IEEE*

Abstract— This paper proposes a new approximate scheme for coordinate rotation digital computer (CORDIC) design; this scheme is based on modifying the existing Para-CORDIC architecture with an approximation that is inserted in multiple parts and made possible by relaxing the CORDIC algorithm itself. A fully parallel approximate CORDIC (FPAX-CORDIC) scheme is proposed; this scheme avoids the memory register of Para-CORDIC and makes the generation of the rotation direction fully parallel. A comprehensive analysis and the evaluation of the error introduced by the approximation together with different circuit-related metrics are pursued using HSPICE as the simulation tool. This error analysis also combines existing figures of merit for approximate computing (such as the Mean Error Distance (MED) and MED Power Product (MPP)) with CORDIC specific parameters; it is shown that a good agreement between expected and simulated error values is found. The Discrete Cosine Transformation (DCT) and the Inverse DCT (IDCT) transformations as case study of approximate computing to image processing are investigated by utilizing the proposed approximate FPAX-CORDIC architecture with different accuracy requirements. The results confirm the viability of the proposed scheme.

Index Terms— Inexact computing, CORDIC, Error distance, Power dissipation

1 INTRODUCTION

Modern computers rely heavily on fast arithmetic computation to solve complex problems with a high degree of accuracy in the results generated. However, accurate hardware implementations for arithmetic processing often incur in large overheads as related to circuit complexity, delay and power consumption. These overheads are more evident at the nanometric scales in which computation encounters physical limitations due to the reduced feature size.

The paradigm of inexact computing relies on relaxing fully precise and completely deterministic computation to balance often contradicting figures of merit such as power consumption and performance [1].

The tradeoffs that are available for inexact computing are very complex once arithmetic processing is considered at a higher level than just a single operation (such as addition or multiplication). Many inexact or approximate adders, multipliers and dividers have been proposed in the technical literature [2-4]; however, these designs are considered and compared often with respect to the implications of an approximation to the operation and its ability to deliver an output of acceptable accuracy (such as image processing or filtering) [5]. This paper addresses the different scenario in which approximation is still implemented in hardware, but it is considered as part of an algorithm, namely for CORDIC implementation [6, 7].

CORDIC is an iterative algorithm for the calculation of the rotation of a 2-dimensional vector in different coordinate systems; the benefits of the CORDIC algorithms are that only additions and shift operations are employed. A hardware implementation of the CORDIC algorithm usually employs a finite number of iterations and a finite level of precision. Therefore, the objective of this paper is to design, evaluate and assess an approximate CORDIC design; the approximation is inserted in multiple parts and is made possible by relaxing the CORDIC algorithm itself. So, power dissipation and accuracy can be reduced when properly selecting the parameters (such as the so-called error control parameter *p*) of the proposed scheme. This paper extends the initial findings of [8]; a comprehensive evaluation of *p* and different circuit metrics including complexity and power dissipation are presented. An error analysis that combines traditional figures of merit (such as the MED [3]) with CORDIC specific parameters is analytically pursued. Simulation shows that when either DCT, or IDCT, or both are inexact, the accuracy and power dissipation are affected. The case in which a single inexact DCT or IDCT is present, shows the most sensitivity to *p*; when both DCT and IDCT are inexact, the error variation and dependency with respect to *p* are reduced.

This paper is organized as follows. Section 2 presents a brief review of the basic and the advanced parallel CORDIC algorithms; this latter algorithm partially elimi-

- *Linbin Chen is with the Electrical and Computer Engineering Department, Northeastern University, Boston, MA 02115. E-mail: chen.lin@husky.neu.edu.*
- *Jie Han is with the Electrical and Computer Engineering Department, University of Alberta Edmonton, AB, Canada. E-mail: jhan8@ualberta.ca.*
- *Weiqiang Liu is with the College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, China. E-mail: liuweiqiang@nuaa.edu.cn.*
- *Fabrizio Lombardi is with the Electrical and Computer Engineering Department, Northeastern University, Boston, MA 02115. E-mail: lombardi@ece.neu.edu.*

This manuscript is an extended version of the conference article listed in the bibliography as [8].

nates the iterative nature of the basic CORDIC algorithm; Section 3 proposes a new fully parallel approximate CORDIC algorithm, as the main contribution of this manuscript. In section 4, The error analysis of the proposed approximate algorithm is derived and compared to simulated data. A detailed evaluation of the proposed hardware scheme is pursued in Section 5; the Synopsys Design Compiler is used as simulation tool. Section 6 deals with a case study of image processing, namely the Discrete Cosine Transformation (DCT) and the Inverse DCT (IDCT). Conclusion is dealt in Section 7.

2 Review

2.1 Radix-2 Circular System CORDIC Algorithm

The focus of this paper is on computing trigonometric functions using the radix-2 CORDIC algorithm in the circular coordinate system. As shown in Fig. 1, a two-dimensional vector (x_0, y_0) is rotated through an angle θ to calculate a rotated vector(x'_n, y'_n). This computation can be performed by the matrix product, $(x'_n, y'_n). = R \cdot (x_0, y_0)$, in which R is the rotation matrix:

$$R = \begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix} \tag{1}$$

By factoring out $cos\theta$, the roatation matrix R becomes

$$R = K\begin{bmatrix} 1 & -tan\theta \\ tan\theta & 1 \end{bmatrix} \tag{2}$$

where $K = (1 + \tan^2\theta)^{-1/2}$.

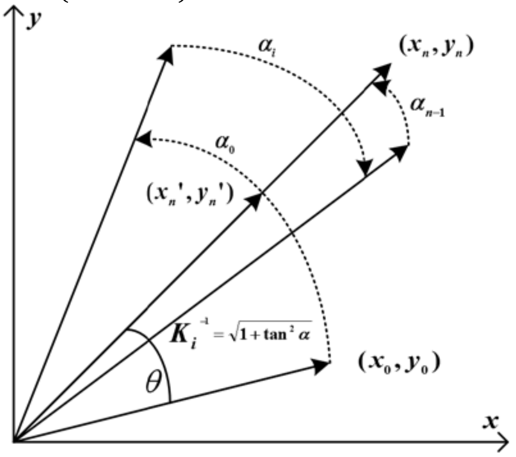


Fig. 1 Two-dimensional vector rotation.

To compute the rotation in hardware, CORDIC decomposes the rotations into a sequence of elementary rotations through predefined angles. In Fig. 1, the rotation through angle θ is decomposed by a sequence of microrotations through the elementary angle α_i. The values of α_i are chosen such that $\tan(\alpha_i) = 2^{-i}$ and the multiplication of the tangent term in Eq. (2) is reduced to a simple shift operation through i bit positions. Instead of performing the rotation directly through an angle θ, CORDIC utilizes a certain number of microrotations through angle α_i, as

$$\theta = \sum_{i=0}^{n-1} \sigma_i\alpha_i, \text{and } \sigma_i = \pm 1 \tag{3}$$

where n indicates the number of microrotations, α_i is the elementary angle for the ith iteration and σ_i is the direction of the ith microrotation. The rotation matrix for the i-th iteration is given by

$$R_i = K_i\begin{bmatrix} 1 & -\sigma_i 2^{-i} \\ \sigma_i 2^{-i} & 1 \end{bmatrix} \tag{4}$$

where $K_i = 1/\sqrt{(1 + 2^{-2i})}$ being the scale factor. K_i for a microrotations does not depend on the direction of microrotations and decreases monotonically; so rather than scaling during each microrotation, the magnitude of final output could be scaled by K as,

$$K = \prod_{i=0}^{n} K_i.$$

Therefore, the iterative equations of the CORDIC algorithm for radix-2 in circular coordinate systems are as follows:

$$\begin{gathered} x_{i+1} = x_i - \sigma_i y_i 2^{-i}, \\ y_{i+1} = y_i + \sigma_i x_i 2^{-i}, \\ \theta_{i+1} = \theta_i - \sigma_i \, tan^{-1}(2^{-i}), \\ \sigma_i \in \{-1, 1\};\ i = 0,2, \dots, n-1. \end{gathered} \tag{5}$$

Finally, the vector (x'_n, y'_n) is obtained as,

$$(x'_n, y'_n) = K \cdot (x_n, y_n)$$

$$K = \prod_{i=0}^{n}(1/\sqrt{1 + 2^{-2i}}).$$

Eq. (5) is often referred to as the CORDIC equation and can be used in either rotation mode and vectoring mode. The direction of the iterative rotation is determined using θ_i or y_i depending on the rotation mode or the vectoring mode respectively. The rotation mode is used when calculating trigonometric functions. The direction of rotation in any iteration is determined using the sign of the residual angle θ_i found in the previous iteration. Let $x_0 = K, y_0 = 0$, after n iterations, the final outputs are the sine and cosine functions, $x_n = cos\theta$ and $y_n = sin\theta$.

2.2 Advanced Partially Parallel CORDIC Algorithm

Eq. (5) shows that the performance bottleneck is the sequential calculation of σ_i. The operations in each stage can be executed only after the corresponding stage has selected the correct rotation direction. If the direction is found in all stages and can be parallelized or pre-computed, the corresponding CORDIC rotations in the microrotation stage can also be executed concurrently.

Different solutions have been proposed in the technical literature [9-13] for the parallel execution of σ_i through the θ-path. In [14], a so-called Para-CORDIC parallelizes the generation of the rotations direction *i.e.*, σ_i from the binary value of the input angle θ by employing a binary to bipolar representation (BBR) and a microrotation angle recoding (MAR) techniques.

The two's complement N+1 bits binary representation of the input angle θ is assumed to be in the range $|\theta| \le \pi/4$ and is given by $(-b_0) + \Sigma_{j=1}^{N} b_j 2^{-j}$, where $b_j \in \{0,1\}$. The input angle θ is divided into the higher part θ^H and the lower part θ^L:

$$\begin{aligned}\theta &= \theta^L + \theta^H \\ &= (-b_0) + \Sigma_{j=1}^{l-1} b_j 2^{-j} + \Sigma_{j=l}^{N} b_j 2^{-j}\end{aligned} \quad (6)$$

In Eq. (6), l is the smallest index value such that $2^{-l} - \tan^{-1} 2^{-l} < 2^{-N}$. It has been shown in [15] that $l = \lceil (N - \log_2 3)/3 \rceil$. Next, a brief treatment of the BBR and MAR method to predict the σ_i for θ^L and θ^H, respectively, is provided.

1) Binary to Bipolar Representation (BBR)

The BBR method converts the first l-1 bits of the input angle (*i.e.,* θ^L) and obtains the corresponding rotation directions (σ_1 *to* σ_l). The binary value $b_j \in \{0,1\}$ is converted to the corresponding bipolar representation $r_k \in \{-1,1\}$ as follows:

$$\begin{aligned}\theta^L &= (-b_0) + \Sigma_{j=1}^{l-1} b_j 2^{-j} \\ &= (-b_0) + \Sigma_{j=1}^{l-1}[2^{-j-1} + (2b_j - 1)2^{-j-1}] \\ &= \Sigma_{i=1}^{l} r_i 2^{-i} - 2^{-l},\end{aligned}$$

where,

$$\begin{aligned}r_1 &= 1 - 2b_0 \\ r_i &= 2b_{i-1} - 1\,, i = 2,3,\dots,l\end{aligned} \quad (7)$$

The first l rotation directions (σ_1 *to* σ_l) are directly derived from Eq. (7) and the bipolar values of r_1 to r_l. Then, θ^L is written as

$$\begin{aligned}\theta^L &= \Sigma_{i=1}^{l} r_i 2^{-i} - 2^{-l} \\ &= \Sigma_{(i=1)}^{l} \sigma_i \left\{ \Sigma_{j=1}^{n(i)} tan^{-1}\left(2^{-s_i^j}\right) + e_i \right\} - 2^{-l}, \\ &\quad where,\ \sigma_i = r_i;\ i = 1,2,\dots,l.\end{aligned} \quad (8)$$

In Eq. (8), 2^i is expressed as the sum of arctangent values and an error term, *i.e.,* $\Sigma_{j=1}^{n(i)} \tan^{-1}\left(2^{-s_i^j}\right) + e_i$ (discussed next in the MAR algorithm).

2) Microrotation Angle Recoding (MAR)

The decomposing of each positional binary weighting $2^{-i}, i = 1,2,\dots,l-1$ into a combination of arctangent terms and a nonnegative error term e_i, yields the following expression:

$$\begin{aligned}2^{-i} &= tan^{-1}(2^{-i}) + \Sigma_{j=2}^{n(i)} tan^{-1}(2^{-s_i^j}) \\ &= \Sigma_{j=1}^{n(i)} tan^{-1}(2^{-s_i^j}) + e_i, \\ s_i^1 &= i,\ i = 1,2,\dots,l-1.\end{aligned} \quad (9)$$

The above equation is generally known as MAR. $n(i)$ is the number of microrotations required in the MAR recording of 2^{-i}, and s_i^j is the shift sequences for $j = 1,2,\dots,n(i)$ with the first shift $s_i^1 = s_i = i, i = 1,2,\dots,l-1$. The detailed algorithm of MAR recording can be found in [14]. By combining Eq (6) and Eq. (8-9), the corrected rotation angle $\hat{\theta}^H$ is given by

$$\hat{\theta}^H = \theta^H + \Sigma_{i=1}^{l-1} \sigma_i e_i - 2^{-l}. \quad (10)$$

Following the BBR for $\theta^L$and the MAR for the first binary positional weightings, another BBR is applied to the binary representation of the corrected $\hat{\theta}^H$ as follows:

$$\begin{aligned}\hat{\theta}^H &= (-\hat{b}_{l-1})2^{-l+1} + \Sigma_{k=l}^{N} \hat{b}_k 2^{-k} \\ &= (-\hat{b}_{l-1}) + \Sigma_{k=l+1}^{N+1}[(2\hat{b}_{k-1} - 1)2^{k+2^{-l}} + 2^{-N-1}] \\ &= \Sigma_{i=l}^{N+1} \hat{r}_i 2^{-i} - 2^{-N-1},\end{aligned}$$

where,

$$\begin{aligned}\hat{r}_l &= 1 - 2\hat{b}_{i-1}, & i &= l \\ \hat{r}_i &= (2\hat{b}_{i-1} - 1), & i &= l+1,\dots,N+1\end{aligned} \quad (11)$$

From Eq. (11) the last N-l+2 rotation direction ($\hat{\sigma}_l$ *to* $\hat{\sigma}_{N+1}$) is found directly from the bipolar value of $\hat{r}_l$ to $\hat{r}_{N+1}$.

3) Para-CORDIC architecture

The Para-CORDIC rotation architecture is shown in Fig. 2. There are two BBRs in the para-CORDIC rotation. In Fig. 2, the notation BBR$_L$ and BBR$_H$ are used to represent the operations in Eq. (7) and Eq. (11) to determine the rotation direction σ_1 to σ_l for phase 1 rotation stages and $\hat{\sigma}_l$ to $\hat{\sigma}_{N+1}$ in phase 2 rotation stages, respectively. The operations in Eq. (10) are denoted as $Add_{prediction}$ block. The errors given by the e_i terms, are precomputed and stored in the memory register.

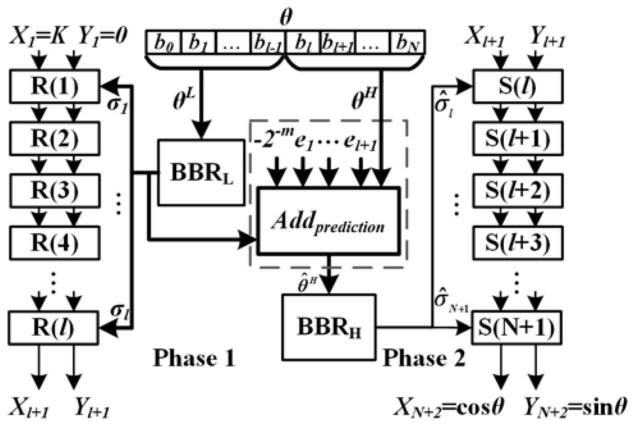


Fig. 2 The Para CORDIC Architecture [14].

3 Proposed Fully Parallel Approximate CORDIC (FPAX-CORDIC)

3.1 FPAX-CORDIC

Although Para-CORDIC parallelizes the computation of σ_i in two phases, this cannot eliminate the dependency between Phase 1 and Phase 2; the generation of the rotation direction σ_i is still not fully parallel. To achieve the goal of fully parallel generation of σ_i, the relation between Phase 1 and Phase 2 must be further analyzed. The parallel generation of σ_i is restricted by the error compensation mechanism in Para-CORDIC, i.e. the error must be assessed following the BBR in Phase 1 and added back by using $Add_{prediction}$ block. So, for a fully parallel execution either error compensation must be performed differently, or the error should be tolerated. A fully parallel approach allows $Add_{prediction}$ to be completely eliminated and have a fully parallel execution of BBR$_L$ and BBR$_H$. In this latter case, there is no need of additional memory to store the error compensation terms e_i; this condition makes the operation of the circuit fully combinational, so also improving its performance. Therefore, a fully parallel approximate CORDIC (FPAX-CORDIC) design is proposed to meet this requirement. By slightly modifying MAR algorithm, the error produced by generation of σ_i can be controlled and recovered in the rotation blocks instead. This makes the whole circuit be a pure multi-operand additions or subtractions structure, and enable flexible control over error and performance of FPAX-CORDIC.

Consider Fig. 2, the block $Add_{prediction}$ is eliminated (denoted now as a dashed rectangle) as per the following analysis. The main function of the $Add_{prediction}$ block is to compensate the error introduced by the first BBR operation in Eq. (10). In MAR for $2^{-i}, i = 1,2,\dots,l-1$, the value of $\hat{\theta}^H$ must satisfy the following constraint:

$$|\hat{\theta}^H| < 2^{-(l-1)} \quad (12)$$

If this condition is met, then each binary weighting of 2^{-i} in the remaining angle can be approximated by $\tan^{-1}(2^{-i})$ within the accuracy allowed by the N fractional bits. As outlined in [14], if $\Sigma_{i=1}^{l-1} e_i < 2^{-l}$, then the value of $|\hat{\theta}^H|$ will satisfy the inequality of Eq. (12). 2^{-l} is the upper bound for $\Sigma_{i=1}^{l-1} e_i$; the higher order terms $\tan^{-1}(2^q), q > i$ in MAR of 2^{-i} can be found to let $\Sigma_{i=1}^{l-1} e_i < 2^{-l}$. So, the number of microrotations (with the shift sequences $s_i^j, j = 1,2,\dots,n(i)$) is directly controlled by $\Sigma_{i=1}^{l-1} e_i$, *i.e.*, the smaller $\Sigma_{i=1}^{l-1} e_i$ is, the larger $n(i)$ is [14].

The complete elimination of the $Add_{predition}$ block in Fig. 2 (so making CORDIC fully parallel) requires that the equation $\Sigma_{i=1}^{l-1} e_i = 0$ to be satisfied: As shown in Eq. (10), if $\Sigma_{i=1}^{l-1} \sigma_i e_i = 0$, then $\hat{\theta}^H = \theta^H - 2^{-l}$ and error compensation is not necessary, *i.e.*, $Add_{prediction}$ can be eliminated. In general, $\Sigma_{i=1}^{l-1} \sigma_i e_i = 0$ is not applicable, because it is only possible to make $\Sigma_{i=1}^{l-1} \sigma_i e_i$ approximately equal to 0. Let $\theta^E = \Sigma_{i=1}^{l-1} \sigma_i e_i$; as each e_i is not negative in MAR, then $|\theta^E| = |\Sigma_{i=1}^{l-1} \sigma_i e_i| \le \Sigma_{i=1}^{l-1} e_i$. So, let $\Sigma_{i=1}^{l-1} e_i \to 0$, hence $\theta^E \to 0$. For an input angle θ with N-bit precision, let $\Sigma_{i=1}^{l-1} e_i < 2^{-N}$ so that the error θ^E can be ignored for N-bit precision.

An algorithm to find the high-order terms $\tan^{-1}(2^q), q > i$ to make $\Sigma_{i=1}^{l-1} e_i < 2^{-N}$ is rather intuitive as it is quite similar to MAR [14]. However, the condition $2^{-N} (\ll 2^{-l})$ for an error-free fully parallel CORDIC is significantly more complex than in Para-CORDIC in terms of the number of microrotation stages, i.e., the number of microrotation $n(i)$ in this case is larger than for Para-CORDIC. When the θ error term $|\theta^E| < 2^{-p}$ is tolerated for a specific application, then $|\theta^E| \le \Sigma_{i=1}^{l-1} e_i < 2^{-p}$, where $p \in [l,\dots,N]$ is the so-called *error tolerant parameter*. In the case of 16-bit CORDIC design to achieve accuracy similar to Para-CORDIC, FPAX-CORDIC should designed with p=16; in general, for an approximate design, the selection of p is between l and N. The proposed FPAX-CORDIC algorithm is presented in pseudo-code in Fig. 3.

```
For a N-bit input angle $\theta$,
//Initial Values
    Find $l = \lceil (N - log_2 3)/3 \rceil$
    Perform Eq.(10) and $\theta^H - 2^{-l}$
    Use the modified MAR to find $s_i^j$ and $e_i$ ($\Sigma_{i=1}^{l-1} e_i < 2^{-p}$).
// Full Parallel Execution
    Perform BBR (for the full range of the N-bit input angle $\theta$)
    From stage 1 to stage $l$
    Perform R(i) using $\sigma_i = r_i, i = 1 \dots l$.
    From stage (l+1) to stage (N+2)
    Perform S(i) using $\sigma_i = r_l, i = l \dots N+1$
```

Fig. 3 FPAX-CORDIC Algorithm.

The architecture of the proposed FPAX-CORDIC is shown in Fig. 4. The two paths, namely θ-path and X/Y path, are discussed as follows.

θ-path: Consider Fig. 2 and Fig. 4; the operations of the data path in the conventional CORDIC are replaced by the BBR in the proposed FPAX-CORDIC. The delay and the hardware overhead of the two BBRs are negligible because they perform a simple signal mapping (*i.e.*, 0 (1) is considered as subtraction (addition) signal).

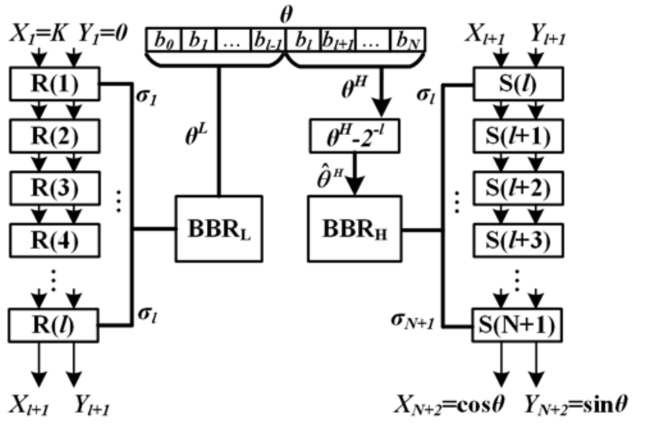


Fig. 4 Architecture of the proposed fully parallel approximate CORDIC.

X/Y-path: In the proposed architecture (Fig. 4), the number of total microrotations is denoted as $Rot(N) = \Sigma_{i=1}^{l-1} n(i) + N - l + 3$. $n(i)$ is directly related to $\Sigma_{i=1}^{l-1} e_i$, and therefore it is also related to the error tolerant parameter p. So, the area and delay are $(4N \times Rot(N))A_{FA}$ and $(2 \times Rot(N))T_{FA}$, i.e. it is assumed that each microrotation stage is implemented using a Binary Signed Digit Adder (BSDA) and the area and delay with a word size of N bits are $4N \times A_{FA}$, where $2T_{FA}$. A_{FA} and T_{FA} are the area and delay of a full adder. The relation between the number of total microrotations and p is shown in Fig. 5; as the delay is proportional to the number of microrotation stages, the delay nearly follows the same trend as the area.

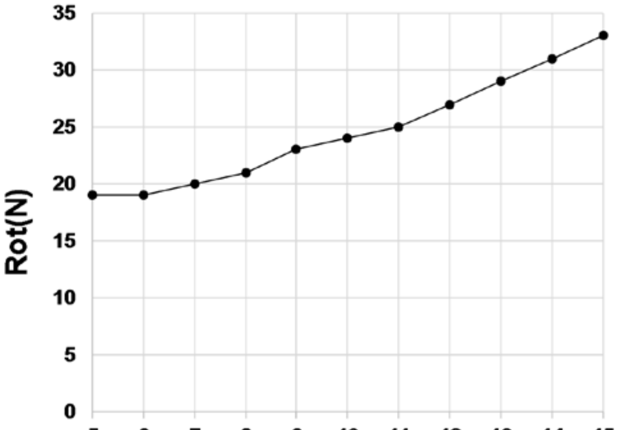


Fig. 5 16-bit FPAX-CORDIC number of the total microrotations stages.

3.2 FPAX-CORDIC with Truncation

By increasing p, the number of microrotation stages $S(i)$ in each *R(i)* can potentially increase depending on the modification of the MAR algorithm. At a specific value of p, the power and area due to the additional microrotation stages may exceed those saved by eliminating the $Add_{prediction}$ block, i.e. the power dissipation and area may be larger than for Para-CORDIC if p is larger than the critical value. Also, a higher p means a lower error at the output; so a different approximate technique must be utilized to increase the critical value of p [16]. As FPAX-CORDIC consists of adders and subtractors, the truncation scheme of [16] can be used; a Vertical Truncation (VT) scheme with a depth of d (the truncation parameter d indicates the number of BSDA adder cells that are removed from the LSB) is applied to the last N-l+2 rotation stages $S(i)$ (i.e. the stages controlled by the second BBR$_H$ block) of FPAX-CORDIC. While truncation saves area and power dissipation, it also introduces an error in the CORDIC computation; however, at a low depth d this error should not be significant compared to the error introduced by p. So, for FPAX-CORDIC, p is important for controlling the first l stages of the *R(i)* blocks, while truncation is utilized only in the last several *S(i)* stages. As shown in later sections, the increase in power dissipation due to the additional stages for p, can be mitigated by truncation (as controlled through d), while still maintaining the error nearly constant. In some application like DCT-IDCT, truncation is an additional scheme to control the error and performance together with p.

4 Error Analysis of FPAX-CORDIC

4.1 Errors

The error of the FPAX-CORDIC with respect to a (real) trigonometric function consists of two parts: (1) the approximation error due to the inertial computation algorithm itself; and (2) the rounding error.

Approximation Error: Initially, assume the input θ and X/Y have an infinite number of bits, i.e., the rounding error is zero. The approximation error of the proposed FPAX-CORDIC algorithm is controlled by the error tolerant parameter p. As $|\theta^E| < 2^{-p}$ is ignored in θ, then

$$sin\,\theta' = sin(\theta - \theta^E) \tag{13}$$

Thus,

$$\begin{aligned} E_{sin}^{AX}(\theta,p) &= \sin\theta - \sin\theta' \\ &= (1-\cos\theta^E)\sin\theta + \sin\theta^E\cos\theta \\ E_{cos}^{AX}(\theta,p) &= \cos\theta - \cos\theta' \\ &= (1-\cos\theta^E)\cos\theta - \sin\theta^E\sin\theta \end{aligned} \tag{14}$$

Rounding Error: The error due to a finite input bit width (for example a 16-bit θ and X/Y) is generally known as the rounding error and denoted as E_{sin}^{RD} and E_{cos}^{RD}. The total error is then given by

$$\begin{aligned} E_{sin} &= E_{sin}^{AX} + E_{sin}^{RD} \\ E_{cos} &= E_{cos}^{AX} + E_{cos}^{RD} \end{aligned} \tag{15}$$

MED Due to Approximation Error: If the rounding error is not considered, then the MED *[3]* can be derived by the integral of the approximation error for $\theta \in [0, 2\pi)$ as,

Eq. (16) has been plotted against the simulated results. Fig. 6(b), Fig. 7(b) and Fig. 8(b) show that the error equation provides a good estimate of the MED for FPAX-CORDIC architecture.

$$\begin{aligned} \text{MED} &= \int_0^{2\pi} E_{sin}^{AX}(\theta,p)d\theta = \int_0^{2\pi} E_{cos}^{AX}(\theta,p)d\theta \\ &= \int_0^{2\pi} |(1-\cos\theta^E)\sin\theta + \sin\theta^E\cos\theta| d\theta \\ &= \begin{cases} \dfrac{\sin\theta^E - \cos\theta^E + 1}{\pi}, & E_{sin}^{AX}(\theta,p) \geq 0 \\ \dfrac{\cos\theta^E - \sin\theta^E - 1}{\pi}, & E_{sin}^{AX}(\theta,p) < 0 \end{cases} \end{aligned} \tag{16}$$

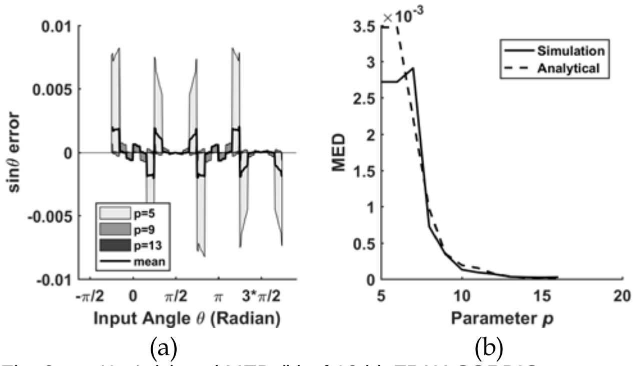


Fig. 6 $E_{sin}(\theta,p)$ (a) and MED (b) of 16-bit FPAX-CORDIC vs. p

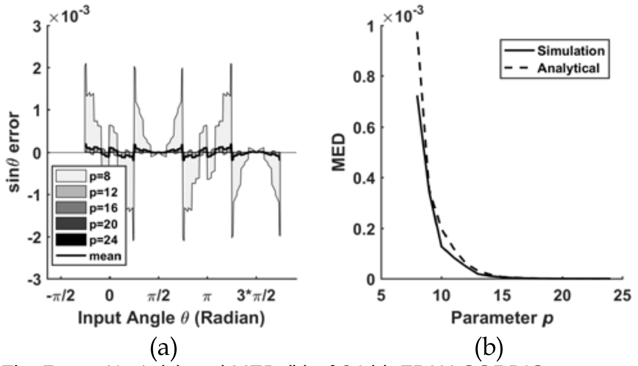


Fig. 7 $E_{sin}(\theta,p)$ (a) and MED (b) of 24-bit FPAX-CORDIC vs. p

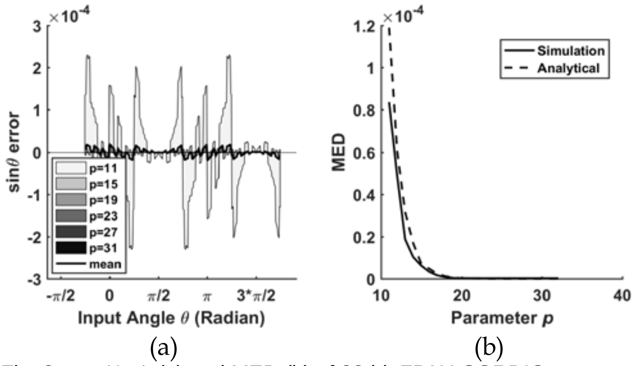


Fig. 8 $E_{sin}(\theta,p)$ (a) and MED (b) of 32-bit FPAX-CORDIC vs. p

The simulated error $E_{sin}(\theta,p)$ versus p for the proposed 16-bit, 24-bit and 32-bit FPAX-CORDIC are plotted in Fig. 6(a), Fig. 7(a) and Fig. 8(a). As p increases, the output error variation decrease to reach nearly zero; the black solid line shows the mean error over the range of p $\in$ [5, 16]; as the word width increases from 16 to 32, the mean error of $E_{sin}(\theta,p)$ decreases by nearly 30 times in magnitude. Fig. 6(b) shows the simulated 16-bit FPAX-CORDIC MED as function of p; the MED drops rather fast between $p = 8$ and 10, and is nearly constant at $p > 10$; however, the simulated MED value is deviates a little from the estimated value. The simulated and analytically estimated MEDs of the 24-bit and 32-bit FPAX-CORDIC (Fig. 7(b) and Fig. 8(b)) show a better agreement than for the 16-bit case; this is caused by the larger rounding error occurring at a smaller bit-width.

4.2 Truncation Error

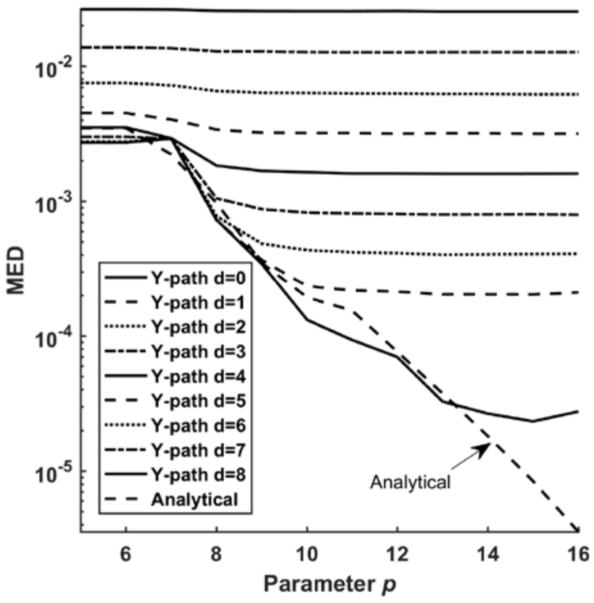


Fig. 9 MED of 16-bit FPAX-CORDIC with Truncation Scheme: Y-Path Truncation

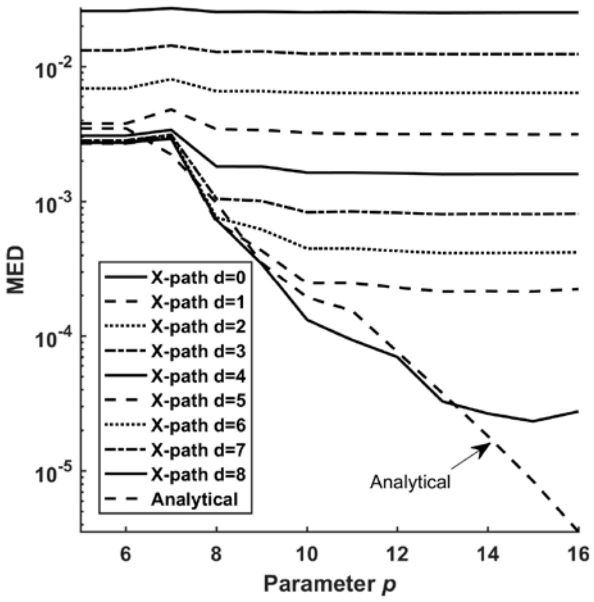


Fig. 10 MED of 16-bit FPAX- CORDIC with Truncation Scheme: X-Path Truncation

The MED of a truncated FPAX-CORDIC either in X-Path or Y-Path is shown in Fig. 9 and Fig. 10. These plots show that when the value of p is lower than 8, the truncation depth d does not significantly affect the MED until d increases beyond 4; when p is higher than 8 and d increases, the MED starts to be different from the analytically estimated value; the impact of d is greater when p has a high value. This suggested that while high truncation depth would save lots of area and power consumption as shown in the following section, however, d has a critical value for impacting mostly the error of a truncated FPAX-CORDIC.

5 Hardware Implementation and Evaluation

A 16-bit FPAX-CORDIC is implemented using Verilog HDL and synthesized using NCSU FreePDK45[17] design kit; as an angle (in radian) θ is represented by 17 bit 2's complementary binary values. The angle information is in the format U(1,16), where bit 16 is the sign bit and the bits [15:0] are the fractional parts. X/Y use a 17-bit 2's complementary binary number representation; its format is given by U(1,16) with an additional MSB guard bit.

1) BBR Block

The BBR_L block is shown in Fig. 11; only a two transistor MUX and an inverter are required per digit positon. The output is given by σ_i for the first l rotation directions. BBR_H is implemented in a similar manner as BBR_L. For the 16-bit input θ, $l = \lceil(16 - \log_2 3)/3\rceil = 5$. So, 5 MUXes and 5 inverters are need for $\text{BBR}_\text{L;}$ and 13 MUXes and 13 inverters are needed for BBR_H.

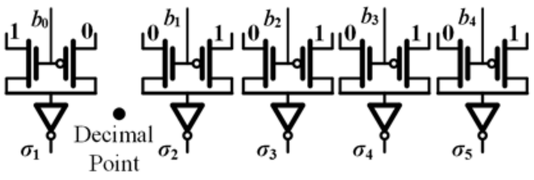


Fig. 11 BBR_L block for 16 bit FPAX-CORDIC.

2) θ-Path

The θ-Path for FPAX-CORDIC is reduced to one single subtraction of $\hat{\theta}^H = \theta^H - 2^{-l}$. The input is given by 17-bits for θ^H, The definition of the θ-path is shown in Fig. 12; a carry-look ahead adder (CLA) is used for this subtraction.

$$\begin{array}{r} \theta^H = 0.\overbrace{00...0}^{(l-1)bits}b_l...b_N \\ -2^{-l} = -0.0...0100...0 \\ \hline \left|\hat{\theta}^H\right| = 0.\underbrace{00...0}_{(l-1)bits}\hat{b}_l\hat{b}_{l+1}...\hat{b}_N \end{array}$$

Fig. 12 The Diagram of θ PATH

3) X/Y Path

As σ_i is fixed a-priori, the X/Y path can be realized using a Binary Signed Digit Adder (BSDA), so addition without propagating a carry. The intermediate rotation results are represented using a Binary Signed-Digit (BSD) numbering system [18] with a digit set given by {-1,0,1}; positive/negative flag encoding (Table 1) is used for each BSD digit.

TABLE 1 ENCODING OF BSD DIGITS

BSD Digits *d*	d^+	d^-
-1	0	1
0	1 or 1	0 or 0
1	1	0

$R(i)$ in Fig. 4 consists of $S(s_i^j)$ with $j = 1,2,\ldots,n(i)$ (as shown in Fig. 13). The basic block of the X/Y path is the microrotation block $S(s_i^j)$; it is implemented using a BSDA (Fig. 14).

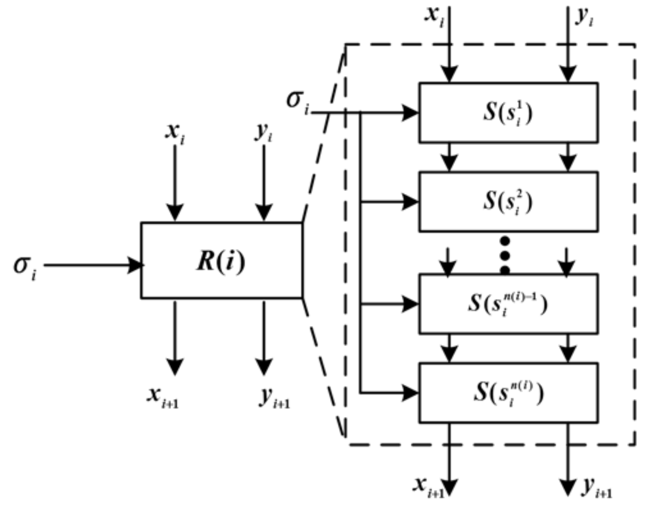


Fig. 13 Implementation of R(i).

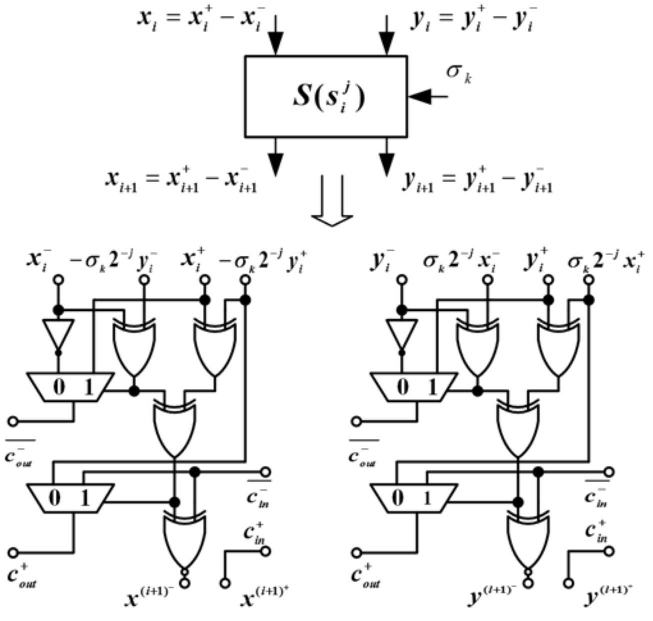

$x_i = x_i^+ - x_i^-$
$y_i = y_i^+ - y_i^-$
$S(s_i^j)$
σ_k
$x_{i+1} = x_{i+1}^+ - x_{i+1}^-$
$y_{i+1} = y_{i+1}^+ - y_{i+1}^-$
x_i^- $-\sigma_k 2^{-j} y_i^-$ x_i^+ $-\sigma_k 2^{-j} y_i^+$
y_i^- $\sigma_k 2^{-j} x_i^-$ y_i^+ $\sigma_k 2^{-j} x_i^+$
0 1
$\overline{c_{out}^-}$
c_{out}^+
$\overline{c_{in}^-}$
c_{in}^+
$x^{(i+1)-}$
$x^{(i+1)+}$
$y^{(i+1)-}$
$y^{(i+1)+}$

Fig. 14 Implementation of the X/Y path $S(s_i^j)$ using BSDA.

4) Power, Delay and area

The power, delay and area of the proposed FPAX-CORDIC are simulated and plotted in Fig. 15, Fig. 16 and Fig. 17. The power dissipation of FPAX-CORDIC is lower than Para-CORDIC when p is lower than the critical value of 8; so, the proposed FPAX-CORDIC performs well in term of power consumption (with p=5, FPAX-CORDIC has a reduction in power dissipation of 10%). When p increases beyond 8, the number of total microrotation stages also increases; so, the power dissipation of FPAX-CORDIC increases beyond the power dissipation of Para-CORDIC. The MED is also plotted as function of p in the same figures; the MED of FPAX-CORDIC is high when p is lower than 8, however an abrupt decrease of 80% occurs when p = 8. The delay of FPAX-CORDIC is better than for Para-CORDIC when p is lower than 10 (13% lower at p=5) and area is smaller when p is lower than 8 (10% lower at p=5). However, the delay and area are higher than those of Para-CORDIC at a higher value of p (i.e. when a high accuracy is preferred). For an error-free FPAX-CORDIC, the power, delay and area increase by 60%, 6% and 66% respectively. However, by using truncation, the delay and area penalties can be mitigated. Fig. 18, Fig. 19 and Fig. 20 show the results for FPAX-CORDIC with truncation (p fixed at 5). If the truncation depth d is below 5, the delay remains the same, the power dissipation and area drop by at most 16% and 12.5% respectively, while maintaining the MED at nearly the same level of magnitude as for a scheme with no truncation.

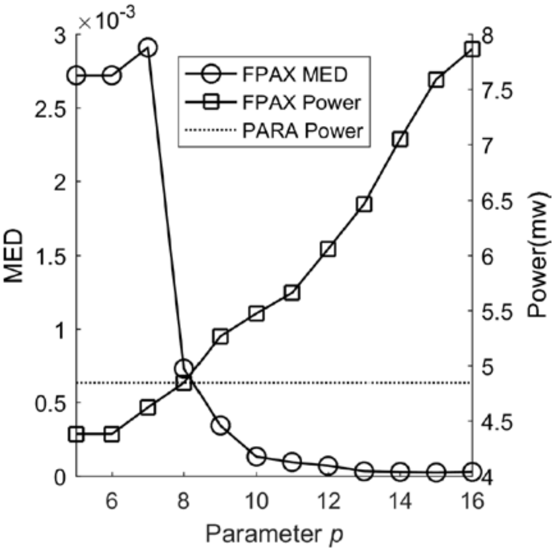


Fig. 15 Power and MED of 16-bit FPAX-CORDIC (d=0)

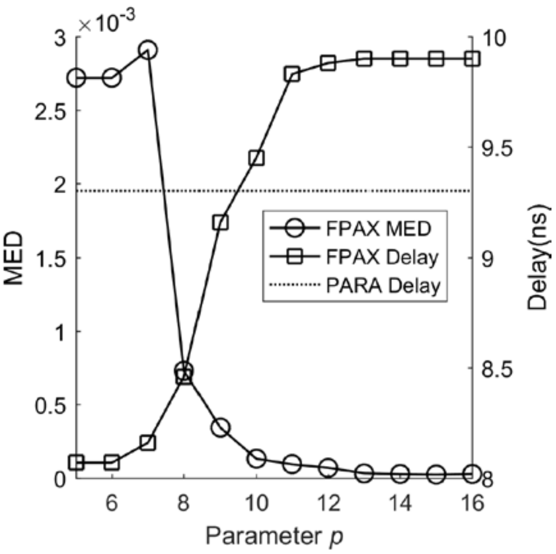


Fig. 16 Critical path delay and MED of 16-bit FPAX-CORDIC (d=0)

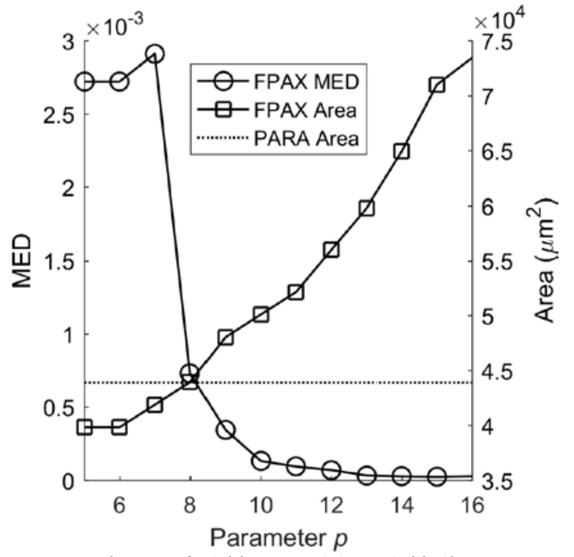


Fig. 17 Area and MED of 16-bit FPAX-CORDIC (d=0)

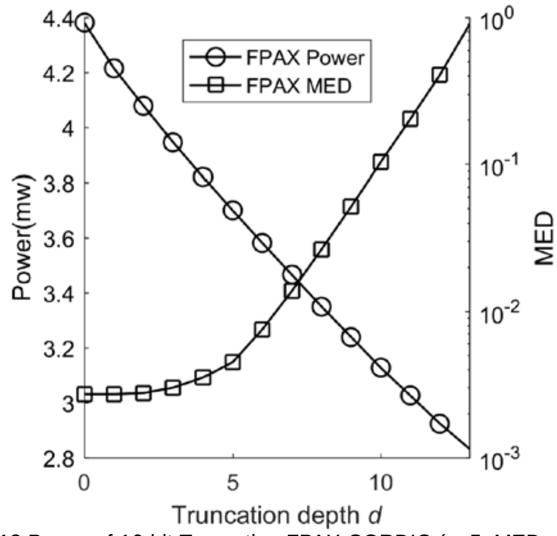


Fig. 18 Power of 16-bit Truncation FPAX-CORDIC (p=5, MED axis is in log scale)

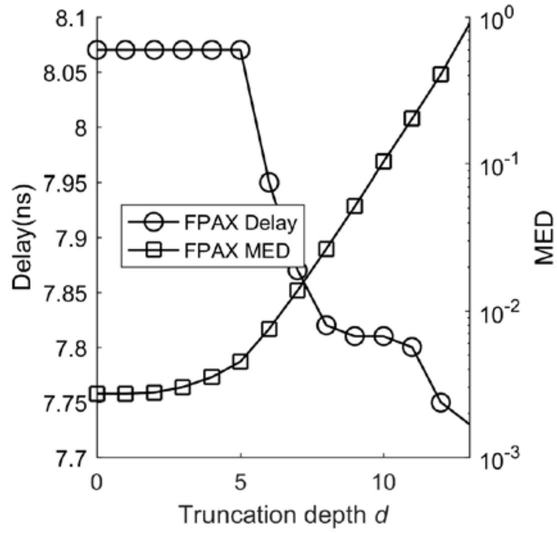


Fig. 19 Delay of 16-bit Truncation FPAX-CORDIC (p=5, MED axis is in log scale)

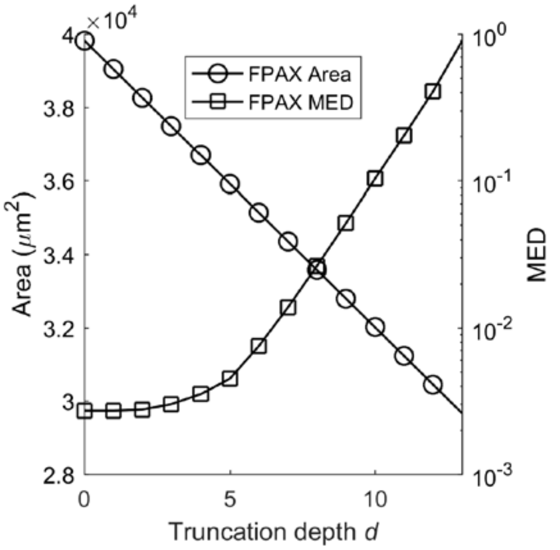


Fig. 20 Area of 16-bit Truncation FPAX-CORDIC (p=5, MED axis is in log scale)

5) Power and MED tradeoff

An approximate design has to address the trade-off between accuracy and power. The MED Power Product (MPP) is introduced in [16] for assessing this trade-off for an approximate design. A lower MPP implies better accuracy and/or power consumption. As shown in Fig. 21, the MPP drops by increasing *p*; at a lower value of *p*, the MPP is stable for a truncation depth smaller than 5. At higher *p*, the MPP increases very rapidly by increasing *d*. Thus, by not considering the delay and area penalties, *p*=15 and *d*=0 are the best values for attaining low power and high accuracy for a 16-bit design.

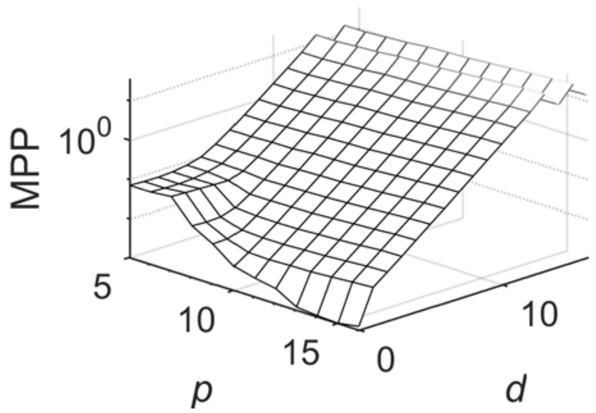


Fig. 21 MPP of 16bit-FPAX CORDIC as a function of parameter p and truncation depth d (z-axis is in log scale)

6 Case Study: Discrete Cosine Transformation (DCT) and Inverse DCT (IDCT)

A wide range of algorithms (such as image enhancement in the spatial domain, frequency transform, image rotation, edge detection) can be implemented using a CORDIC architecture [19]; a discrete cosine transform

(DCT) [20] expresses a finite sequence of data points in terms of a sum of cosine functions oscillating at different frequencies. The focus of this section is on the 2-D 8×8 DCT-IDCT, which is widely used in image compression applications. As shown in Fig. 22, the source image (Lena) is divided into 8×8 small blocks; the DCT and IDCT computations are applied to each 8×8 block.

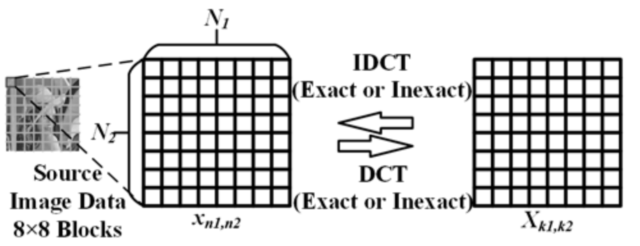


Fig. 22 2D DCT-IDCT process for image compression application.

Throughout this section, the FPAX-CORDIC architecture uses a 17-bit U(1,16) for the θ-path, 32-bit U(16,16) for the X/Y-path (each input image pixel is in the range [0,255]).

6.1 Inexact 2-D 8×8 DCT based on FPAX-CORDIC

In this section, the DCT computation is implemented using the proposed FPAX-CORDIC. The 2-D DCT is decomposed into a 1-D DCT (row-wise DCT) followed by another 1-D DCT (column-wise DCT). The process with the separable 1-D DCTs is shown as Fig. 23; the resulting DCT image is compared against the results of Para-CORDIC.

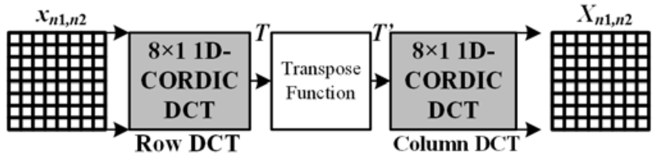


Fig. 23 Process for 2-D 8×8 DCT.

The 8×1 1-D DCT transform is given by [21]

$$X(k) = \frac{c(k)}{2}\sum_{i=0}^{7} x(i)\cos\frac{(2i+1)k\pi}{16} \qquad (17)$$

where

$$k = 0,1,2,\ldots 7$$
$$c(k) = \begin{cases} \frac{1}{\sqrt{2}}, & k = 0 \\ 1, & k > 0 \end{cases}$$

By Eq. (17) and using the DCT and trigonometric symmetric property, the 8×1 1-D DCT in matrix form is as follows:

$$\begin{bmatrix} X(4) \\ X(0) \end{bmatrix} = \frac{1}{2}\begin{bmatrix} c_4 & -s_4 \\ s_4 & c_4 \end{bmatrix}\begin{bmatrix} x(0)+x(7)+x(3)+x(4) \\ x(1)+x(6)+x(2)+x(5) \end{bmatrix}$$
$$\begin{bmatrix} X(6) \\ X(2) \end{bmatrix} = \frac{1}{2}\begin{bmatrix} c_6 & -s_6 \\ s_6 & c_6 \end{bmatrix}\begin{bmatrix} x(0)+x(7)-x(3)-x(4) \\ x(1)+x(6)-x(2)-x(5) \end{bmatrix}$$
$$\begin{bmatrix} X(1) \\ X(7) \end{bmatrix} = \frac{1}{2}\begin{bmatrix} c_7 & s_7 \\ -s_7 & c_7 \end{bmatrix}\begin{bmatrix} x(3)-x(4) \\ x(0)-x(7) \end{bmatrix} + \frac{1}{2}\begin{bmatrix} c_3 & s_3 \\ -s_3 & c_3 \end{bmatrix}\begin{bmatrix} x(1)-x(6) \\ x(2)-x(5) \end{bmatrix} \qquad (18)$$
$$\begin{bmatrix} X(3) \\ X(5) \end{bmatrix} = \frac{1}{2}\begin{bmatrix} c_3 & -s_3 \\ s_3 & c_3 \end{bmatrix}\begin{bmatrix} x(0)-x(7) \\ x(3)-x(4) \end{bmatrix} - \frac{1}{2}\begin{bmatrix} c_1 & s_1 \\ -s_1 & c_1 \end{bmatrix}\begin{bmatrix} x(2)-x(5) \\ x(1)-x(6) \end{bmatrix}$$

where $c_k = cos(k\pi/16) = s_m = \sin(m\pi/16), m = 8-k$.

The implementation of Eq. (18) for an 8×1 1D-CORDIC DCT is shown in Fig. 24; six fix-angle FPAX-CORDICs are used to complete the multiplication of the trigonometric terms in Eq. (18) because for an input $\theta \in [\pi/4, -\pi/4]$ to the FPAX-CORDIC design, all rotation angles are converted to this restricted range.

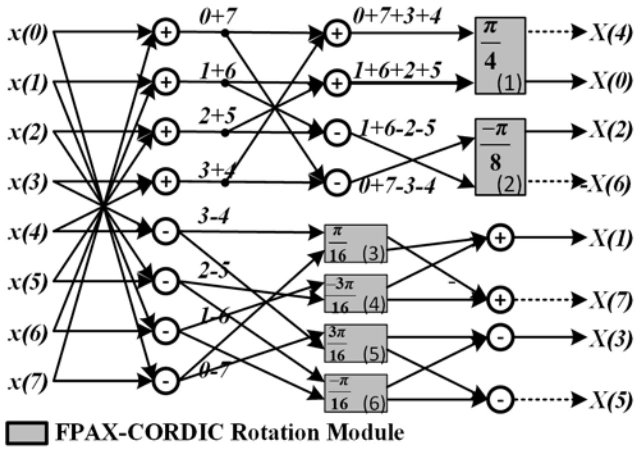


Fig. 24 Architecture of 8×1 1D-CORDIC DCT.

In the architecture of the 8×8 DCT, the parameter p controls the accuracy of the FPAX-CORDIC modules, so accuracy of the inexact 2-D 8×8 DCT is affected in two respects. (1) for each 8×1 1D-CORDIC DCT (Fig. 23), six parallel FPAX-CORDIC modules can be configured by changing p; (2) the two 8×1 1D-CORDIC DCT are connected in series (Fig. 24), but they can be configured using different values of p. For simplicity, the value of p is kept the same for all FPAX-CORDIC modules in the following analysis.

6.2 Configurations of 8×8 DCT based on FPAX-CORDIC

To further reduce the complexity of the proposed architecture for an 8×8 DCT and attain a better DCT accuracy, two features are considered for the six FPAX-CORDIC modules.

1) Non-equal precision path

When considering the reconstruction of the original input image $x(i)$ from the DCT compressed image $X(k)$, the high frequency components of $X(k)$ (for example $X(4)$~$X(7)$) have a small impact on the reconstructed image *x(i)*. So, the X/Y path of the six FPAX-CORDIC modules in the 1D-CORDIC architecture can have non-equal precision, i.e. truncation scheme is used at the X(4)~X(7) computation path; for example, the X-path of the FPAX-CORDIC module (1) (3) and the Y-path of the FPAX-CORDIC module (2) (4) (5) (6) can either have a small precision (*i.e.,* a smaller number of bits), or simply be truncated. The high frequency or non-critical outputs are shown in Fig. 24 with dashed arrows.

2) p sets

On the basis of an unequal precision path configuration to further control the error for the low-frequency or critical outputs, p of each individual FPAX-CORDIC module can be adjusted to meet the error specifications of an application. In each FPAX-CORDIC module, the output usually consists of a low-high frequency component pair; for example, for module (1), the output pair is $(X(4), X(0))$; as the accuracy of $X(4)$ is reduced, so p affects solely $X(0)$. Similar conditions may also apply to the other five FPAX-CORDIC modules.

3) Image test

Fig. 25 shows the error impact of different configurations of FPAX-CORDIC in an inexact DCT application; the original image $x(i,j)$ is transformed using a FPAX-CORDIC based 2-D DCT architecture to generate an inexact DCT $\hat{X}(i,j)$, then the original $\hat{x}(i,j)$ is recovered using an exact IDCT transformation. The PSNR of $\hat{x}(i,j)$ against $x(i,j)$ is measured and the results are plotted in Fig. 26.

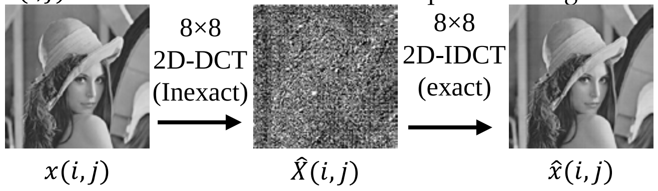


Fig. 25 Evaluation of the error of FPAX-CORDIC based inexact DCT scheme.

The PSNR (Fig. 26) shows that higher are the values of p and lower the truncation depth d, the higher the PSNR is. At lower truncation depth (below 12), the PSNR is not impact by the truncation. The PSNR begins to decrease after truncation depth reaches 12. When p>8, the PSNR increases rapidly and reaches the largest value.

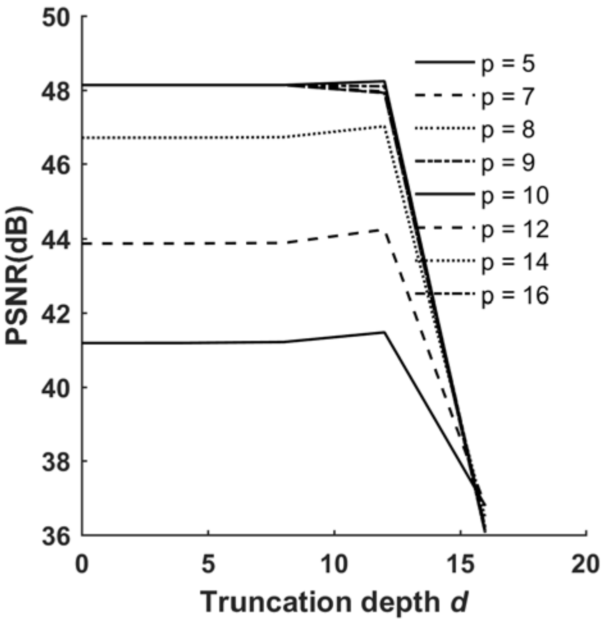


Fig. 26 PSNR for inexact DCT application.

6.3 Inexact 2-D 8×8 IDCT

The IDCT computation is implemented using the proposed FPAX-CORDIC; the original image as reconstructed by both FPAX-CORDIC and Para-CORDIC is compared. The 8 points 1-D IDCT transform are given in [22] as follows:

$$x(i) = \frac{1}{2}\sum_{k=0}^{7} c(k)X(k)\cos\frac{(2i+1)k\pi}{16} \tag{19}$$

where

$$i = 0,1,2,\dots 7$$

$$c(k) = \begin{cases} \frac{1}{\sqrt{2}}, & k = 0 \\ 1, & k > 0 \end{cases}$$

By decomposing the sum in Eq. (19), and using the DCT and trigonometric symmetric property, the 8×1 1-D DCT is expressed in matrix form as follows:

$$\begin{bmatrix} x(6) \\ x(0) \\ x(2) \\ x(3) \\ x(1) \\ x(7) \\ x(5) \\ x(4) \end{bmatrix} = \frac{1}{2}\begin{bmatrix} c_4 & -s_4 & c_6 & -s_6 & c_5 & -s_5 & s_1 & c_1 \\ s_4 & c_4 & s_6 & c_6 & s_1 & c_1 & s_5 & c_5 \\ c_4 & -s_4 & -c_6 & s_6 & s_5 & c_5 & -c_1 & s_1 \\ s_4 & c_4 & -s_6 & -c_6 & -c_1 & s_1 & -c_5 & s_5 \\ c_4 & -s_4 & c_6 & -s_6 & -c_5 & s_5 & -s_1 & -c_1 \\ s_4 & c_4 & s_6 & c_6 & -s_1 & -c_1 & -s_5 & -c_5 \\ c_4 & -s_4 & -c_6 & s_6 & -s_5 & -c_5 & c_1 & -s_1 \\ s_4 & c_4 & -s_6 & -c_6 & c_1 & -s_1 & c_5 & -s_5 \end{bmatrix} \times \begin{bmatrix} X(0) \\ X(4) \\ X(2) \\ X(6) \\ X(7) \\ X(1) \\ X(3) \\ X(5) \end{bmatrix} \tag{20}$$

where $c_k = \cos\left(\frac{k\pi}{16}\right) = s_m = \sin\left(\frac{m\pi}{16}\right), m = 8 - k$.

Similar to DCT, the architecture of 8×1 1D-CORDIC IDCT is shown as Fig. 27.

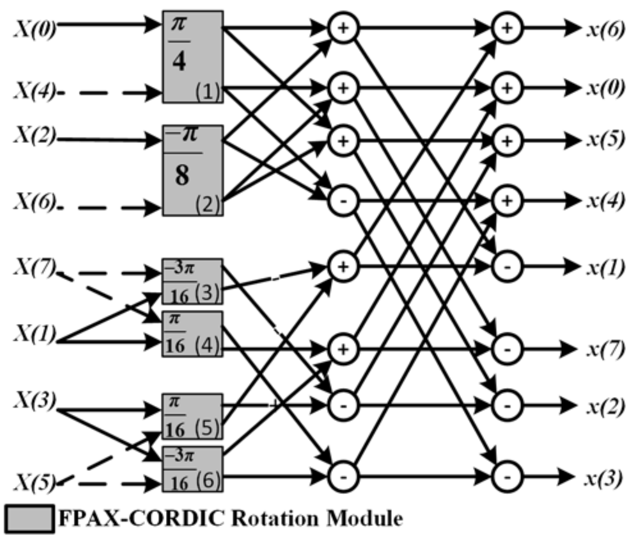


Fig. 27 Architecture of 8×8 1D-CORDIC IDCT.

6.4 Configurations of 8×8 IDCT based on FPAX-CORDIC

Similar to DCT, the IDCT can be also configured according to two features: 1) Non-equal precision path and 2) Different p sets.

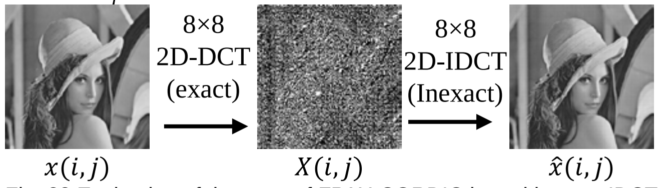


Fig. 28 Evaluation of the error of FPAX-CORDIC based inexact IDCT scheme.

As shown in Fig. 28, to find the error impact of different configurations of FPAX-CORDIC in the IDCT application, the original image $x(i,j)$ is transformed using the exact DCT to obtain $X(i,j)$, and then the original $\hat{x}(i,j)$ is recovered using FPAX-CORDIC IDCT to obtain the inexact IDCT result $\hat{x}(i,j)$. The PSNR of $\hat{x}(i,j)$ against $x(i,j)$ is measured and the results are plotted in Fig. 29. Consider the measured PSNR; similar as the inexact DCT application the higher p and the lower the d are, the higher the PSNR is. At lower truncation depth (below 8), the PSNR is not impact by the truncation. The PSNR begins to decrease after truncation depth reaches 8. When p>8, the PSNR reaches the largest value. Compared to using an inexact DCT and exact IDCT, the use of an exact DCT and inexact IDCT is more sensitive to the truncation depth in X/Y-path.

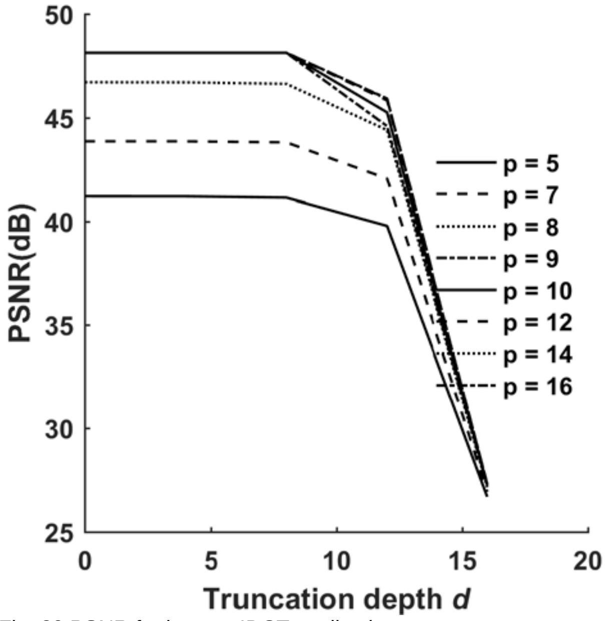


Fig. 29 PSNR for inexact IDCT application

6.5 All Inexact DCT and IDCT Using FPAX-CORDIC

To further decrease the circuit complexity and power dissipation, both DCT and IDCT are computed approximately using the FPAX-CORDIC (Fig. 30). The PSNR of $\hat{x}(i,j)$ against $x(i,j)$ is measured; the results are plotted in Fig. 31.

The results show that compared to the former two cases of using only a single inexact processing, the use of both inexact DCT and IDCT makes the PSNR less sensitive to p. This occurs because DCT and IDCT are inverse transformations, the error introduced by the DCT stage could be properly compensated by IDCT. Thus, the proposed FPAX-CORDIC based architecture is quite suitable for low power DCT-IDCT pair computation.

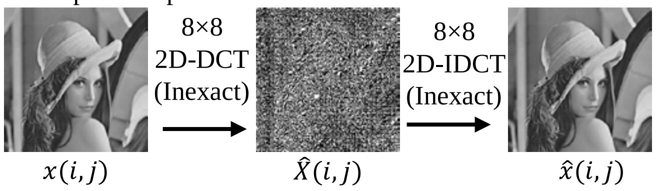


Fig. 30 Evaluation of the error of FPAX-CORDIC based inexact DCT-IDCT scheme.

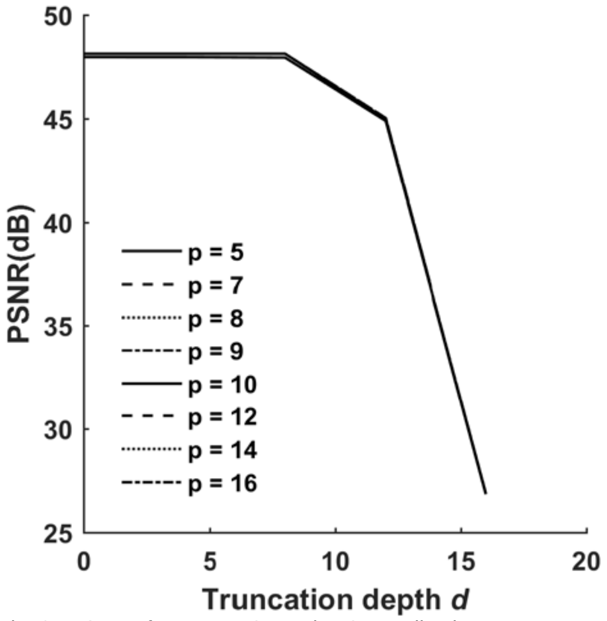


Fig. 31 PSNR of Inexact DCT and IDCT application.

6.6 Hardware implementation of DCT and IDCT

For the DCT-IDCT application presented previously, FPAX-CORDIC at different values of p has been described in a previous section. For the non-equal precision path introduced solely for the DCT-IDCT application, a simple gate [23] (shown in Fig. 32(a)) is used to turn off the data paths of the LSB of the Y-path. An example of the proposed scheme is shown in Fig. 32(b); the bit-width of the data path is controlled by the dynamic bit-width control (DBC) circuit. In this scheme, the hardware overhead consists of the pull-up and pull-down turn-off gate transistors for each X/Y-path bit in the positional logic.

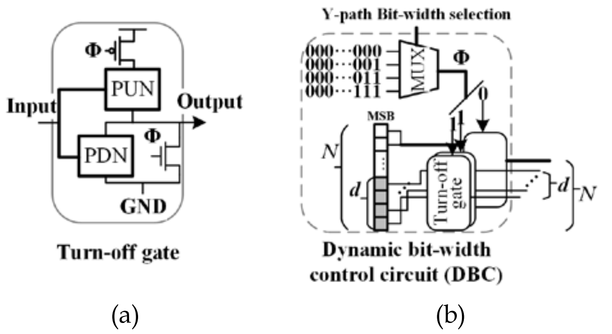
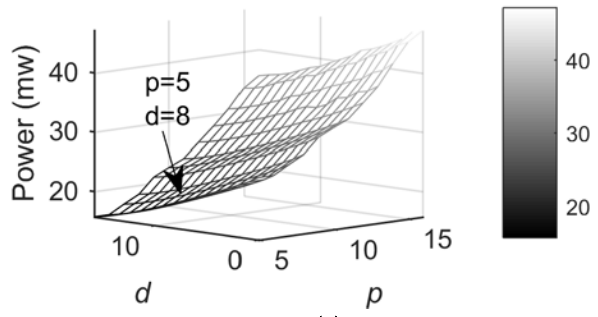
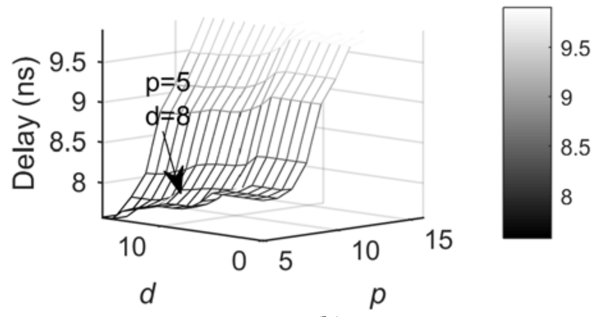


Fig. 32 (a) Turnoff gate [20]. (b) Dynamic bit-width control using turnoff gate.

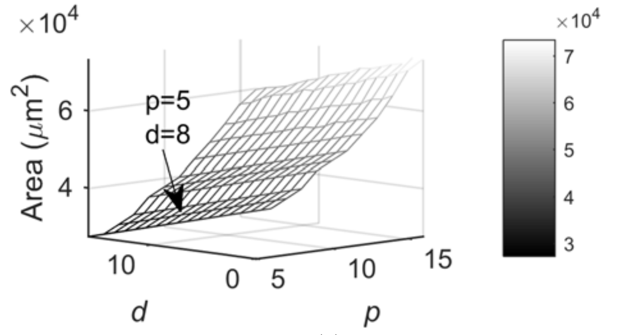
The performance of a single 8×8 DCT-IDCT computation is measured and the results are plotted in Fig. 33. When considering the PSNR (shown in Fig. 31), a better accuracy is achieved at a higher power consumption, delay and area. However, as p does not significantly affect the PSNR, a lower value of p results in a lower power, delay and area. Performance is improved when d is chosen to be not higher than 8 for a high PSNR. Therefore, to achieve both low power dissipation and high accuracy in DCT-IDCT a scheme with p=5 and d=8 offers the best combined performance.



(a)



(b)



(c)

Fig. 33 (a) Power (b) Delay (c) Area of single 8×8 DCT-IDCT pair.

7 Conclusion

This paper has proposed an approach for approximate CORDIC designs. By modifying the Para-CORDIC architecture, a full parallel approximate CORDIC (FPAX-CORDIC) scheme has been proposed. The proposed approximate CORDIC avoids the memory register of Para-CORDIC and makes the generation of the rotation direction fully parallel. Two parameters namely the error-tolerant parameter p and the truncation depth d are proposed and utilized for error control. Power consumption can be reduced when properly selecting the parameters in the proposed scheme, while not results in a great degradation in accuracy. A comprehensive evaluation of the impact of design parameter p and d on circuit metrics including power, delay and area has been presented. An error analysis that combines traditional figures of merit (such as MED) with CORDIC specific parameters is analytically pursued. Therefore, the following conclusive evidence is applicable:

- The parameters p and d are used to control the error of FPAX-CORDIC. As p increases more accuracy is obtained till an error-free computation is reached at p equal to the bit-width N. The truncation depth d is less significant in the error control process; at a depth d less than 5, truncation can save power, delay and area while still keeping a nearly constant accuracy.
- The circuit metrics of FPAX-CORDIC are also a function of p and d. The power dissipation and area increase with p, with similar values as Para-CORDIC at p=8. The delay is smaller than for Para-CORDIC when $p < 10$.
- The error parameters p and d are important when assessing the trade-off between power consumption and accuracy for the proposed FPAX-CORDIC; the values of p=15 and d=0 yield the best MPP.
- For image compression and decompression process using DCT-IDCT, in addition p, more power saving can be gained by tolerating more errors for the high frequency components. The additional power saving can be realized by using the DBC circuit on the X/Y-path of each FPAX-CORDIC module.
- When either DCT, or IDCT, or both are inexact, the accuracy and power dissipation are affected. The case in which a single inexact DCT or IDCT is present, shows the most sensitivity to p and d; when both DCT and IDCT are inexact, then the error variation and dependency with p are reduced. p=5, d=8 are the best design parameter for DCT-IDCT pair computation.

Acknowledgment

This work is partially supported by a grant from the National Natural Science Foundation of China (No. 61401197).

References

[1] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," in *Proc. 18th IEEE European Test Symposium (ETS)*, 2013, pp. 1-6.
[2] H. Jiang, J. Han, and F. Lombardi, "A Comparative Review and Evaluation of Approximate Adders," in *Proc. 25th Great Lakes Symposium on VLSI*, Pittsburgh, Pennsylvania, USA, 2015, pp. 343-348.
[3] J. Liang, J. Han, and F. Lombardi, "New Metrics for the Reliability of Approximate and Probabilistic Adders," *IEEE Trans. on Computers*, vol. 62, pp. 1760-1771, 2013.
[4] M. J. Schulte and E. E. Swartzlander, "Truncated multiplication with correction constant," *VLSI Signal Processing VI*, pp. 388-396, 1993.
[5] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-Power Digital Signal Processing Using Approximate Adders," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, pp. 124-137, 2013.
[6] J. E. Volder, "The CORDIC Trigonometric Computing Technique," *IRE Trans. on Electronic Computers*, vol. EC-8, pp. 330-334, 1959.
[7] J. S. Walther, "A unified algorithm for elementary functions," in *Proc. Spring Joint Computer Conference*, Atlantic City, New Jersey,

1971, pp. 379-385.

[8] L. Chen, J. Han, W. Liu, and F. Lombardi, "A Fully Parallel Approximate CORDIC Design," in *Proc. ACM/IEEE Symposium on Nano Architectures*, Beijing, 2016, pp. 197-202.

[9] D. Timmermann, H. Hahn, and B. J. Hosticka, "Low latency time CORDIC algorithms," *IEEE Trans. on Computers*, vol. 41, pp. 1010-1015, 1992.

[10] T. Srikanthan and B. Gisuthan, "A novel technique for eliminating iterative based computation of polarity of micro-rotations in CORDIC based sine–cosine generators," *Microprocessors and Microsystems*, vol. 26, pp. 243-252, 2002.

[11] B. Gisuthan and T. Srikanthan, "Pipelining flat CORDIC based trigonometric function generators," *Microelectronics Journal*, vol. 33, pp. 77-89, 2002.

[12] H. S. Kebbati, J. P. Blonde, and F. Braun, "A new semi-flat architecture for high speed and reduced area CORDIC chip," *Microelectronics Journal*, vol. 37, pp. 181-187, 2006.

[13] B. Lakshmi and A. S. Dhar, "CORDIC Architectures: A Survey," *VLSI Design*, 2010.

[14] T.-B. Juang, S.-F. Hsiao, and M.-Y. Tsai, "Para-CORDIC: parallel CORDIC rotation algorithm," *IEEE Trans. on Circuits and Systems I: Regular Papers*, vol. 51, pp. 1515-1524, 2004.

[15] S. Wang, V. Piuri, and E. E. Swartzlander, Jr., "Hybrid CORDIC algorithms," *IEEE Trans. on Computers*, vol. 46, pp. 1202-1207, 1997.

[16] L. Chen, J. Han, W. Liu, and F. Lombardi, "On the Design of Approximate Restoring Dividers for Error-Tolerant Applications," *Computers, IEEE Transactions on*, vol. PP, pp. 1-1, 2015.

[17] J. E. Stine, I. Castellanos, M. Wood, J. Henson, F. Love, W. R. Davis, *et al.*, "FreePDK: An Open-Source Variation-Aware Design Kit," presented at the Proceedings of the 2007 IEEE International Conference on Microelectronic Systems Education, 2007.

[18] A. Avizienis, "Signed-Digit Number Representations for Fast Parallel Arithmetic," *IRE Transactions on Electronic Computers*, vol. EC-10, pp. 389-400, 1961.

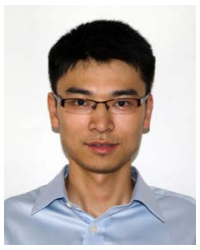
[19] S. Sathyanarayana, R. K. Satzoda, and S. Thambipillai, "Unified Cordic Based Processor for Image Processing," in *Proc. 15th International Conference on Digital Signal Processing*, 2007, pp. 343-346.

[20] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete Cosine Transform," *IEEE Trans. on Computers*, vol. C-23, pp. 90-93, 1974.

[21] M.-W. Lee, J.-H. Yoon, and J. Park, "Reconfigurable CORDIC-Based Low-Power DCT Architecture Based on Data Priority," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 22, pp. 1060-1068, 2014.

[22] T.-Y. Sung, "Memory-efficient and high-performance 2-D DCT and IDCT processors based on CORDIC rotation," in *Proc. 7th WSEAS International Conference on Multimedia Systems & Signal Processing*, Hangzhou, China, 2007, pp. 7-12.

[23] J. Park, J. Choi, and K. Roy, "Dynamic Bit-Width Adaptation in DCT: An Approach to Trade Off Image Quality and Computation Energy," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 18, pp. 787-793, 2010.



Linbin Chen received the B.Sc. degree in information engineering from Beijing Institute of Technology, China, in 2009, and the M.S. degree from Northeastern University, Boston, USA, in 2012, where he is currently working toward the Ph.D. degree in the Department of Electrical and Computer Engineering. His research interests include low power and high performance VLSI design, emerging logic and memory devices and circuits, inexact and fault tolerant computing.

Weiqiang Liu (S'10-M'12-SM'15) received the B.Sc. degree in Information Engineering from Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing, China and the Ph.D. degree in Electronic Engineering from the Queen's University Belfast, Belfast, UK, in 2006 and 2012, respectively. He is currently an associate professor in the College of Electronic and Information Engineering at Nanjing University of Aeronautics and Astronautics, Nanjing, China. His research interests include approximate computing, computer arithmetic and cryptographic hardware.

Jie Han (S'02-M'05-SM'16) received the B.Sc. degree in electronic engineering from Tsinghua University, Beijing, China, in 1999 and the Ph.D. degree from Delft University of Technology, The Netherlands, in 2004.
He is currently an associate professor in the Department of Electrical and Computer Engineering at the University of Alberta, Edmonton, AB, Canada. His research interests include approximate computing, stochastic computation, reliability and fault tolerance, nanoelectronic circuits and systems, and novel computational models for nanoscale and biological applications.

Fabrizio Lombardi (M'81-SM'02-F'09) graduated in 1977 from the University of Essex (UK) with a B.Sc. (Hons.) in Electronic Engineering. In 1977 he joined the Microwave Research Unit at University College London, where he received the Master in Microwaves and Modern Optics (1978), the Diploma in Microwave Engineering (1978) and the Ph.D. from the University of London (1982).
He is currently the holder of the International Test Conference (ITC) Endowed Chair Professorship at Northeastern University, Boston. His research interests are bio-inspired and nano manufacturing/computing, VLSI design, testing, and fault/defect tolerance of digital systems. He has extensively published in these areas and coauthored/edited seven books.