High-Throughput Image Rotation using Sign-Prediction based Redundant CORDIC Algorithm

S. Suchitra *S.K. Lam* *T. Srikanthan*

Centre for High Performance Embedded Systems
Nanyang Technological University, Singapore- 639798

ABSTRACT

Real-time image rotation forms a core operation in many applications such as medical image processing and computer vision. High-throughput computations for image rotation are a common requirement in real-time image processing. A VLSI design for image rotation that employs CORDIC was discussed in [2]. In this paper, we propose novel techniques to increase the throughput of the CORDIC computations, thereby notably improving the overall performance of the rotation unit with acceptable increase in VLSI area. An efficient sign-prediction method called BTSP (**B**inary-**T**ree based **S**ign **P**rediction) is proposed in our redundant CORDIC system to eliminate the sign detection process. Our investigations show that the BTSP based CORDIC algorithm for real-time rotation of a 512×512-pixel image, has a significant speed-up over existing redundant CORDIC methods with reduced hardware area.

I. INTRODUCTION

Real-time image rotation forms a core operation in many applications such as medical image processing and computer vision. Rotation of a gray level image involves performing trigonometric computations on each pixel. For example the x′ and y′ values of a pixel, which has been rotated by an angle of θ is shown in (1) and (2), where (x_0,y_0) is the original position of the pixel, and δ = +1/–1 when θ is +ve /-ve ("+ve" refers to positive and "-ve" to negative).

$$x' = \cos\theta\,(x_0 - \delta y_0 \tan\theta) \quad (1)$$

$$y' = \cos\theta\,(y_0 + \delta x_0 \tan\theta) \quad (2)$$

CORDIC, developed by Volder [1] is a popular hardware-efficient algorithm that can be employed to compute the complex trigonometric functions in (1) and (2). It performs a series of micro-rotations on a vector lying on the X-Y plane over a desired input angle using simple add-shift operations. The CORDIC algorithm is based on the principle that any angle can be approximated as a summation of *n* micro angles of the form arctan (2^{-i}) (i.e. $\theta \approx (\sum_{i=1}^{n} \pm \arctan(2^{-i}))$). Equations (1) and (2) are rewritten as (3) and (4) by replacing the multiplication with simple shift operations by *i* positions.

$$x_{i+1} = k(x_i - \delta_i y_i 2^{-i}) \quad (3)$$

$$y_{i+1} = k(y_i + \delta_i x_i 2^{-i}) \quad (4)$$

The cosθ terms in (1) and (2) accumulates to a constant *k* that converges towards 1.6468. This is known as the scaling factor and can be incorporated either at the start or end of the CORDIC iterations. To evaluate δ at each step, a third variable z is introduced to keep track of the angle pending for rotation as shown in (5). z is initialized with the input angle and the sign of z at any point indicates the direction of the next rotation.

$$z_{i+1} = z_i - \delta_i \arctan 2^{-i} \text{ where } \delta_i = \begin{cases} +1, & z_i \geq 0 \\ -1, & z_i < 0 \end{cases} \quad (5)$$

CORDIC is an approximation algorithm and the smaller the value of $|z_i|$ is in the final iteration, the higher the precision of computation.

Ghosh and Majumdar [2] proposed a parallel architecture for Image Rotation using CORDIC where a 512 × 512 image is first divided into an 8 × 8 window grid. As explained in [2], the rotated positions of the

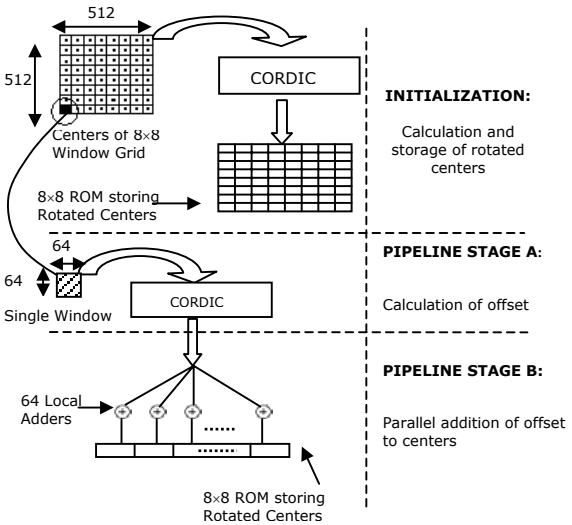

Fig 1: Illustration of Rotation Method in [2]

centers of all windows are first pre-calculated and stored in the initialization stage (Fig 1). In Pipeline Stage A, the CORDIC engine is used to compute the offset of each pixel within a window. In the Pipeline Stage B, this offset is simultaneously added to the 64 centers using local adders to obtain the positions of the 64 corresponding points of all the windows. The CORDIC engine employed for a 512 × 512 image in [2] performs 12 iterations on 20-bit operands followed by a 10 bit truncation. The number of CORDIC iterations is limited by the size of the image and does not overly increase for larger images. In this paper, we propose a novel CORDIC architecture for such a rotation engine to realize high throughputs.

II. BTSP CORDIC - THE PROPOSED ARCHITECTURE

The problem with conventional CORDIC units that employ serial adders for x, y, z computation is that the carry propagation delay manifests as a major bottleneck to the performance. Hence, we propose to use BSD (Binary Signed Digit), the radix-2 Redundant Number System to achieve speedup for CORDIC algorithms by exploiting the limited carry propagation property of such number systems [5]. However, redundant CORDIC carries with it the problem of sign-detection [3][4]. It will be demonstrated in this paper that the complexity of sign detection can nullify the benefits of using Redundant Number Systems, especially for CORDIC with low number of iterations. So, in order to justify the use of BSD, we need to hide the latency of BSD sign detection. In this paper, we propose an elegant sign prediction method called BTSP (Binary Tree based Sign Prediction) that eliminates both the sign detection circuitry and the z datapath. Also, we employ a simple pre-scaling method for scaling factor compensation.

2.1 BTSP Sign Prediction Scheme

From (5), it can be seen that for each of the iterations, the direction of rotation, given by δ_i, can be ± 1. So, for n micro-rotations, there are 2^n possible rotation combinations. Hence, an n-bit CORDIC engine would approximate any given input angle to one of the following 2^n 'target-angles': $\pm\tan^{-1}2^{-1} \pm\tan^{-1}2^{-2} \pm\tan^{-1}2^{-3}\pm \ldots \pm\tan^{-1}2^{-n}$ This can be visualized using a binary tree structure as shown in Fig 2, where the nodes describe the angles that are obtained by following the rotation directions specified by the branches. The leaf nodes correspond to the target-angles.

As the angle rotated by CORDIC approaches the input angle, higher accuracy is achieved. This leads us to an interesting corollary: the knowledge of the index of the closest 'target angle' is sufficient to deduce the CORDIC rotation directions that need to be taken. For example, if

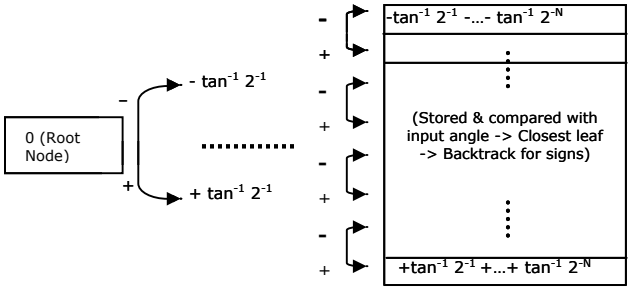

Fig 2: Binary Tree Representation of 'Target-Angles'

the input is 0.000…01 and the closest-leaf is Leaf #0, then the sequence to be taken is "000…0", where 0 signifies a –ve rotation and 1 signifies a +ve rotation.

The basic steps of the BTSP are summarized below:

1. Pre-compute and store all these leaves
2. Match the input angle to the closest leaf
3. Derive the index of the closest leaf and obtain the sequence of δ_i based on the binary tree topology.

For a 512×512 image, about $2^{12} = 4096$ values need to be stored and matched against a single input angle to determine the closest-leaf. Hence, for the BTSP algorithm to work efficiently, the problems of the large storage and complex angle matching must be tackled.

2.2 Tackling the Storage Problem

To reduce the storage of the leave angles, we can store only the +ve leaves. For negative operands, we deduce the directions for the corresponding +ve operand and then modify the operation sign of CORDIC based on the fact that $\sin(-\theta) = -\sin\theta$ and $\cos(-\theta) = \cos\theta$. To further solve the problem of storage, we make use of Theorem 1 as stated below [7]:

Theorem 1: *In N bit CORDIC, the first ($\lceil N/3 \rceil$ - 1) signed digits cannot be precomputed*

It follows from Theorem 1 that for $i > \lceil N/3 \rceil - 1$, the rotation directions can be directly inferred from the bit weights since $\tan^{-1}2^{-i} \approx 2^{-i}$. The angle pending for rotation after $\lceil N/3 \rceil - 1$ iterations (referred to as 'Remaining-Angle') in 2's complement format can be converted to a polar form with digit-set {1, -1} to infer the remaining rotation directions. The recoding mechanism uses a simple logic, which is independent of the operand length, shown in [4]. +1 and –1 would imply rotation directions +ve and –ve respectively. Hence, sign-detection becomes an issue only for the first ($\lceil N/3 \rceil - 1$) rotation directions. This means that for an N-iteration CORDIC, we need to store only $2^{\lceil N/3 \rceil - 1}$ values. Since we are storing only +ve leaves ($\delta_1 = 1$), only 4 leaves needs to be stored for a 512×512 image. A snapshot of the leaves is shown in Fig 3. The

Remaining-Angle can be obtained by performing a subtraction between the input and the closest-leaf.

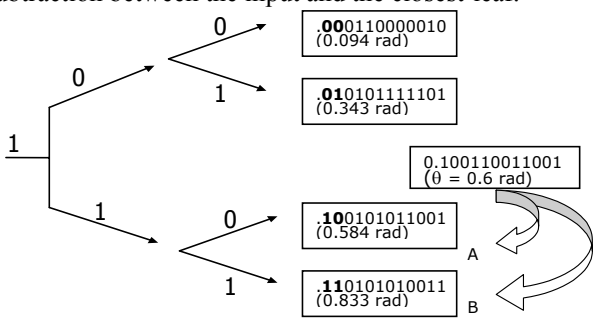

Fig 3: Illustration for Closest Leaf Matching

2.3 Tackling the Matching Problem

The complexity to match the input angle to the closest leaf is now greatly reduced since we need to check the input angle against only 4 values. The matching process can be further simplified by the following analysis. It can be seen in Fig 3 that the first 2 bits display an incremental pattern. Hence checking just 2 bits of the input angle would eliminate all but 2 leaves as candidates for the 'Closest-Leaf'. As an illustration, consider Input Angle = 0.**10**1110100111. The first 2 bits signify that the input angle lies between leaves A and B. Validating whether the input angle is closer to A or B would normally require 2 subtractions and 1 comparison. Alternatively, a less costly approach is used to pre-compute and store all the midpoints of two consecutive leaves. So, a single comparison with the relevant midpoint would lead us to the closest-leaf. The overall-architecture for a 12-iteration BTSP CORDIC engine is shown in Fig 4. Note that the computation of the Remaining-Angle is done in parallel to the first 3 iterations. Due to the parallel inference of δ from the recoded Remaining-Angle, an extra rotation has to be performed for maintaining the accuracy [4].

The following describes an example of the 12-iteration CORDIC using BTSP. The midpoints of the leaves shown in Fig 3 are:

Mid Pt # 0	0.001101111111	0.219
Mid Pt # 1	0.011101101011	0.464
Mid Pt # 2	0.101101010110	0.709

Given an input angle (θ): 0.100110011001(0.6 rad),

1. Fix the first direction to be +ve; $\delta_1 = +1$.
2. Examine the first 2 bits to extract Mid Pt # 2
3. Perform comparison between Mid Pt # 2 and θ
4. Since θ is lesser than Mid Pt # 2, Leaf A (Leaf # 2) is the Closest-Leaf. This implies $\delta_{2\text{-}3} = +1,-1$
5. While Iterations 1-3 are being computed, subtract Closest-Leaf from θ to obtain the Remaining-Angle (0.000001000000)
6. Perform 2'Complement to Polar Conversion of bits 4 – 12 to obtain $\delta_{4\text{-}13} = +1,-1,-1,+1,-1,-1,-1,-1,-1,-1$

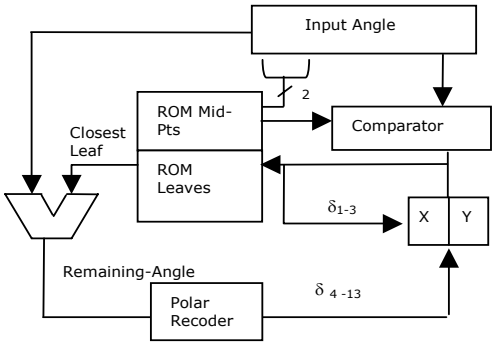

Fig 4: BTSP-based CORDIC Architecture

2.4. Scaling Factor Compensation

To compensate for k in (3) and (4), a serial pre-scaling model described in [6] is employed. During the initialization process of the rotation engine, the X and Y input registers of the CORDIC engine are first loaded with the pre-scaled coordinates of the first center C_0/k. The subsequent centers are computed by continuously incrementing the coordinates by a constant value. So, for a 512 × 512 image, X or Y is incremented by $64/k$ to obtain the next pre-scaled center. During the calculation of the offsets for a single window in Pipeline Stage A (Fig 1), an incremental of each pixel coordinate by $1/k$ is performed. These incremented values are passed to the X, Y input registers of the CORDIC engine.

III. RESULTS & COMPARISONS

The proposed BTSP CORDIC engine is compared with Conventional 2's complement CORDIC, and those that use BSD sign-detection methods proposed by Takagi et al [3] viz. Double Rotation and Correction Rotation. Table 1 and Table 2 summarize the latency and overall-area estimates in Full Adder Latency (T_{FA}) and Full-Adder Area (FA) respectively. The estimates are standardized according to the Passport 0.35 Micron Library [8]. As specified in [2], a 12-iteration, 20-bit CORDIC is used for a 512×512 image. So, the X and Y datapaths are 20 digits long. The Z datapath is assumed to be 12 digits long. The BTSP Method performs an extra 13^{th} rotation as mentioned earlier. The overall BTSP CORDIC architecture comprises of a pair of 20-digit redundant adders, registers and shifters, a 12-bit CLA for computing the "Remaining Angle", a CLA-based 12-bit comparator for deducing the closet-leaf, ROM storing seven 12-bit values and an angle recoder. Takagi's CORDIC require an additional sign detection circuit. For all the BSD methods, redundant adders are used, and a 1-Bit Redundant Adder (RA) is estimated to be 2FA. Latency of an RA is independent of operand length and is

estimated to be $2T_{FA}$. From (8), a register is 1.13 times the size of an adder.

LATENCY (in T_{FA})				
	Conv	Double	Corr (m=4)	BTSP
X, Y (or Z)	$\log_2 20 \times 12$	4×12	$[(0.25 \times 4) + (0.75 \times 2)] \times 12$	2×13
Shifting	$12 \times 4 \times 0.64$	$12 \times 4 \times 0.64$	$12 \times 4 \times 0.64$	$13 \times 4 \times 0.64$
Sign-Detect	-	$\log_2 3 \times 12$	$\log_2 6 \times 12$	-
Comparison	-	-	-	$\log_2 12 = 3.58$
Rem-Angle	-	-	-	3.58(hidden)
Recoding	-	-	-	0.17(hidden)
Total	**82.58**	**97.68**	**91.68**	**62.86**

Table 1: Comparison of Latency Estimates

AREA (in FA)				
	Conv	Double	Corr(m=4)	BTSP
X,Y	$2 \times 20 \times 2.13$	$2 \times 2 \times 20 \times 2.13$	$2 \times 2 \times 20 \times 2.13$	$2 \times 2 \times 20 \times 2.13$
Z	12×2.13	$2 \times 12 \times 2.13$	$2 \times 12 \times 2.13$	-
Shifter	$20 \times 4 \times 0.47$	$40 \times 4 \times 0.47$	$40 \times 4 \times 0.47$	$40 \times 4 \times 0.47$
ROM	$12 \times 12 \times 1.13$	$12 \times 12 \times 1.13$	$12 \times 12 \times 1.13$	$7 \times 12 \times 1.13$
Sign	-	6	12	-
Comp	-	-	-	2×12
Recoder	-	-	-	12×0.13
Adder	-	-	-	2×12
Total	**273.48**	**459.44**	**465.44**	**390.08**

Table 2: Comparison of Area Estimates

Shifting is carried out using a barrel shifter, implemented as a tree of multiplexers. In this case, the maximum shift of 12-bits warrants 4 levels of MUX. So the latency attributed to the shift operation is $4 \times 0.64 = 2.57\ T_{FA}$. Sign Detection in case of Takagi is carried out using a CLA-based circuit. The area estimate for a CLA is about $2n$ FA and latency is $\log_2 n T_{FA}$, where n = length of operand. In case of Double Rotation, 3 digits are checked and 2 redundant additions are performed at every stage. In the case of Correction Rotation (for m=4), 3, 4, 5 and 6 digits need to be checked in consecutive cycles and every fourth rotation is a double-rotation. Hence a 6-digit sign-detection circuit is employed. A similar CLA-based circuit is employed for the comparison circuit in BTSP. The latency for a 12-bit comparison is about T_{FA} The 'Remaining-Angle' is computed using a CLA. The angle recoding can be realized using a simple circuit consisting of inverters. The combined latency of CLA addition and recoding is about $3.58 + 0.17 = 3.75\ T_{FA}$, which can be hidden completely during the 1st 3 iterations. As for ROM requirements (estimated as DFFs), it is assumed that 12 arctan values (of length 12-bits each) are stored for both the sign-detection methods. For the BTSP Method, 4 leaves and 3 midpoints are stored. ROM access time is ignored in the calculations.

It can be seen from Table 1 that the latency of BSD sign detection-based CORDIC is worse than that of the conventional 2's complement CORDIC. The proposed CORDIC using BTSP has 31%, 55%, 46% latency gains when compared to Conventional, Double Rotation and Correction Rotation methods respectively. The proposed method has more area than conventional CORDIC, due to the usage of redundant numbers, but has lesser area than the other BSD-based sign detection methods.

IV. CONCLUSIONS

In this paper, we have proposed novel techniques to accelerate BSD CORDIC computation, which forms the most compute intensive core of an image rotation unit. The technique to eliminate sign detection and z data path employs only a simple lookup table. Our investigations reveal that the gain in terms of computation time with respect to conventional 2's complement and redundant number correction rotation method [3] is about 31% and 46% respectively. Moreover, the incorporation of a simple scaling-factor compensation method have contributed to the realization of a high-performance CORDIC unit that is suitable for realizing real-time image rotations.

V. REFERENCES

[1] Volder, J.E., "*The CORDIC Trigonometric Computing Technique*", IRE Trans. Elecron. Comput., Vol. EC-8, pp 330-334, 1959

[2] Ghosh, I., and Majumdar, B., *Design of an application specific VLSI chip for image rotation,* Seventh International Conference on VLSI Design, pp 275 – 278, 1994

[3] Takagi, N., Asada, T., and Yajima, S., *Redundant CORDIC Methods with a Constant Scaling Factor for Sine and Cosine Computation,* IEEE Transactions on Computers, Vol. 40, pp 989-995, 1991

[4] Timmermann, D., Hann, H., and Hosticka, B.J., *Low Latency Time CORDIC Algorithms*, IEEE Transactions on Computers, Vol. 41, pp 1010-1015, 1992

[5] Parhami, B., *Carry-Free Addition of Recoded Binary Signed-Digit Numbers*, IEEE Transactions on Computers, Vol 37, pp 1470-1476, 1988

[6] Feng, Z., and Peter K., A *High Speed Hough Transform Using CORDIC,* Proc. Int. Conf. On Digital Signal Processing, Cyprus, 1995.

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

[8] Avant! Passport 0.35 micron, 3.3 Volt SC Library CB35OS142, March 2000.