On the Hardware Reduction of z-Datapath of Vectoring CORDIC

R. Stapenhurst*, K. Maharatna**, J. Mathew*, J.L.Nunez-Yanez* and D. K. Pradhan*
*University of Bristol, Bristol, UK
**University of Southampton, Southampton, UK
km3@ecs.soton.ac.uk

Abstract— **In this article we present a novel design of a hardware optimal vectoring CORDIC processor. We present a mathematical theory to show that using bipolar binary notation it is possible to eliminate all the arithmetic computations required along the z-datapath. Using this technique it is possible to achieve three and 1.5 times reduction in the number of registers and adder respectively compared to classical CORDIC. Following this, a 16-bit vectoring CORDIC is designed for the application in Synchronizer for IEEE 802.11a standard. The total area and dynamic power consumption of the processor is 0.14 mm^2 and 700μW respectively when synthesized in 0.18μm CMOS library which shows its effectiveness as a low-area low-power processor.**

I. INTRODUCTION

The vectoring mode of CoOrdinate Rotation DIgital Computer (CORDIC) algorithm is an effective means for computation of the magnitude and phase angle of a vector [1]. In this mode of operation, the y component of the input vector is forced to zero using iterative vector rotation in a *to and fro* manner through a set of elementary rotation angles. At the end, the magnitude value and the accumulated angle (the phase angle) are available as the x and z component of the output. In terms of hardware, this vector rotation is nothing but a series of simple shift-and-add operations and the resulting structure is very hardware economical. Owing to its attractiveness it has been incorporated in several DSP and communication systems and a large volume of research work has been dedicated to improve its speed and hardware requirement [2 – 4].

This particular work concerns about realization of a hardware optimal vectoring CORDIC processor for IEEE 802.11a standard which requires evaluation of magnitude and phase angle of a vector for the Synchronizer [5]. Typical requirements of such a system are small silicon area and low-power. We have earlier proposed a low-power hardware optimal architecture for vectoring CORDIC for such system [6] where using the technique of scaling-free CORDIC formulation in conjunction with Domain folding [7, 8] and one sided vector rotation we were able to eliminate all the arithmetic operations along the angle accumulation or z-datapath and also showed that a convergence range of $[0, \pi/8]$ is sufficient for a vectoring (or forward rotational) CORDIC to cover the entire coordinate space. However, two problems were associated with that formulation, *viz*., it is not possible to apply the same formulation to reduce the hardware for the conventional *two-sided* vector rotation and for a wordlength larger than 18-bits the hardware requirement of it becomes more than the classical CORDIC.

In this particular work we propose a formulation to eliminate all the arithmetic operations along the z-datapath for conventional *two-sided* vector rotation and thereby reducing the hardware while increasing the accuracy. Also the resulting architecture shows significant hardware saving as the wordlength increases. Although we stick to the 2's complement number system, without loss of generality, this formulation can be adopted easily for redundant arithmetic and higher radix formulation. A 16-bit processor developed following this formulation requires 0.14 mm^2 area and consumes 700 μW dynamic power when synthesized in 0.18μm CMOS library. The rest of the paper is structured as follows: Section II proposes the novel formulation for eliminating all the arithmetic along the z-datapath, in Section III we describe the 16-bit vectoring CORDIC architecture following this formulation and in Section IV we evaluate the performance of the architecture. Conclusions are drawn in Section V.

II. ELIMINATION OF Z-DATAPATH FOR TWO-SIDED ROTATION

In a scaling-free CORDIC the i^{th} elementary rotational angle is described as $\alpha_i = 2^{-i}$. In this formulation, considering one-sided vector rotation where direction of each elementary rotation $\sigma_i \in \{0, 1\}$, there exists a one-to-one correspondence between the elementary CORDIC section undergoing a rotation and position of a '1' in the binary representation of the final accumulated angle. Thus if an elementary stage undergoing rotation is designated by a '1' and otherwise '0' then after the data flows through all the pipelined sections the bit pattern emerged from these pipelined stages actually represent the accumulated angle and thus there is no need for any sort of real computation along the z-datapath apart from keeping some registers to hold the intermediate bits emerging from each of the stages. However, for two-sided rotation there exists no such direct correspondence since $\sigma_i \in \{-1, 1\}$ and thus the method stated above cannot be exploited directly to eliminate the conventional requirement of ROM and adders along the z-datapath. The next section describes the method to find the correspondence of the direction of rotation and the final accumulated angle.

Lemma: If the +ve and −ve rotations are denoted by '1' and '0' respectively then a cyclic right shift of the accumulated bit pattern represents the actually accumulated angle.

Proof: Let us consider the angle to be accumulated is given by

$$\phi = 2^{-j} + 2^{-k} + 2^{-l} + 2^{-m} \quad (1)$$

where j, k, l, $m \in \{0, \ldots, b-1\}$, b being the wordlength and j being the starting point of iteration index. In binary notation φ can be expressed as having '1' at the positions j, k, l and m while other positions will contain '0'. Now, if we consider that a '1' and a '0' corresponding to an iteration index denotes a +ve and −ve rotation respectively then we can write

$$\phi' = 2^{-j} - \sum_{i=j+1}^{k-1} 2^{-i} + 2^{-k} - \sum_{i=k+1}^{l-1} 2^{-i} + 2^{-l} - \sum_{i=l+1}^{m-1} 2^{-i} + 2^{-m} - \sum_{i=m+1}^{b-1} 2^{-i} \quad (2)$$

where φ' is the angle actually computed by this method. Applying algebraic modification, equation (2) can be written as

$$\phi' = 2^{-(k-1)} + 2^{-(l-1)} + 2^{-(m-1)} + 2^{-(b-1)} \quad (3)$$

From equation (3) it can be seen that a right shift of φ' followed by substitution of positional value of b^{th} bit to j^{th} bit position yields equation (1) and hence the actual angle to be accumulated. This operation is nothing but a cyclic right shift of the accumulated bit pattern. □

It is pretty straightforward to verify that the same argument is true if the j^{th} pipeline stage starts with a –ve rotation. Thus, using this formulation, once again, it is possible to find out the required phase angle by tracking only direction of rotation exhibited by each of the elementary rotational stages and thus there will not be any requirement of arithmetic computation in the z-datapath.

III. ARCHITECTURE

The biggest problem of the scaling-free CORDIC architecture in [6] is that in order to maintain the accuracy in the definition of the elementary rotational angle $\alpha_i = 2^{-i}$ the lowest value of i has to be $i_{min} = p = \lfloor (b - 2.585) / 3 \rfloor$. Thus to ensure the convergence over the convergence range of $[0, \pi/8]$ (which was proved to be sufficient to cover the entire coordinate space in [6 – 8]) one needs to repeat $i=p$ elementary rotational stage by $N = \lfloor (\pi/8)/2^{-p} \rfloor$ times. As b increases the value of p increases and accordingly N increases rapidly resulting in no more advantage compared to the classical CORDIC anymore. In order to overcome this problem we propose a hybrid scheme by integrating appropriate number of classical CORDIC elementary rotational stages with the scaling-free elementary rotational stages. It is to be noted that the neumerical value of $\pi/8$ is 0.392699. Thus the highest value of i required to cover the convergence range will be $i = 2$ since $2^{-2} = 0.25$ whereas $2^{-1} = 0.5$ which is beyond the convergence range. Thus to cover the convergence range it is sufficient to integrate $K = (p-2)$ classical elementary rotational sections (with $\alpha_i = \tan^{-1} 2^{-i}$) with $(b-p)$ scaling-free sections. However, incorporating these classical elementary rotational sections raises two particular issues: 1) a scaling circuitry needs to be incorporated at the end of the CORDIC pipeline and 2) the formulation described in Section II does not hold because of the definition of α_i. The first problem can be tackled using the same approach as that of the classical CORDIC. However, since the number of classical stages integrated are far less than the number of classical stages used in the conventional CORDIC, for a given wordlength it can be said intuitively that the scaling circuitry required here will consume less hardware compared to the classical one. The second problem can be tackled by using a small ROM where the combinationations of the angles corresponding to the classical stages could be stored and finally be added to the angle accumulated by the scaling-free stages (which computes the actual angle by using the theory developed in Section II). However the ROM size increases as 2^K. But using the symmetric property of the combination of angles it is possible to reduce its size to 2^{K-1}. In that case the output adder needs to be changed to an adder/subtractor. To illustrate the whole design, here we describe the design of a 16-bit processor. Without any loss of generality, the method described here can be extended for other wordlengths as well.

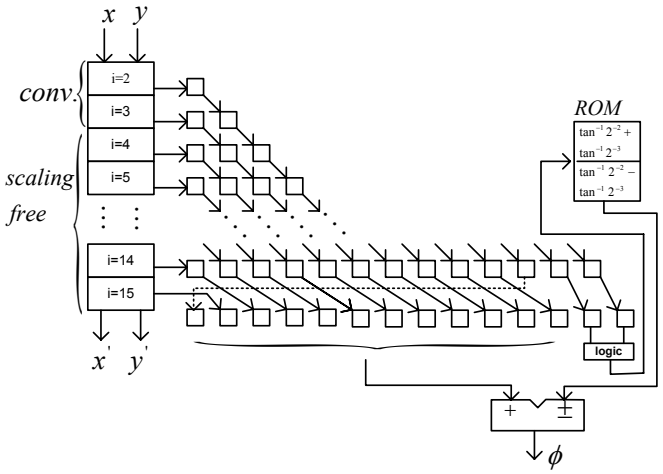

Figure 1: Proposed basic CORDIC pipeline

The complete processor consists of three separate units *viz.*, domain, basic pipeline and output unit.

Domain: This unit is reponsible for carrying out the domain folding operation (described in [6 – 8]) by which the vector lying in the range of $\theta \in (\pi/8, \pi/2]$ is mapped to $\varphi \in [0, \pi/8]$. Theoretically the whole operation can be viewed as a pre-rotation of the input vector by $\pi/4$ (when the vector lies in the range $(\pi/8, 3\pi/8]$), this is equivalent to the stage i=0 for classical CORDIC or swapping the x and y values (when the vector lies in the range $(3\pi/8, \pi/2]$). Thus the hardware requirement of this unit consists of a couple of comparetors and adders and a scaling unit by $\sqrt{2}$. However, since we need to use a scaling unit for scale factor compensation of the conventional rotational stages at the output of the processor, the scaling by $\sqrt{2}$ is merged with it. Thus the hardware rquirement of the complete *Domain* circuit is two

comparators and two adder/subtractors. The unit also generates two 2-bit signals *quad* and *domain* which indicates the quadrant and the domain in which the initial vector lays and pass these signals to the basic pipeline along with the modified values of the input vector.

Basic pipeline: According to our number convention the decimal 1 is defined as 0100000000000000. Since the total wordlength is 16-bit the value of p is 4. Thus we need i=2 and 3 conventional stages to be integrated with the scaling-free stages of i=4, …, 15. Each of the scaling-free stages require four adder/subtractors [6]. However, for $i \geq b/2$, the hardware requirement of scaling free stages becomes same as that of the classical CORDIC i.e., two adder/subtractors. The overall basic pipeline is shown in Figure 1. The entire basic pipeline is 14 stages long. But in order to balance the pipeline completely it is possible to concatenate each of the stages having two adder/subtractors together and thereby reducing the number of pipelined stages. However, we have not adopted this here since our main aim is to find out the total hardware requirement following one-to-one theoretical mapping to architecture.

Since we have used i=2 and 3 conventional stages, we need to store four possible combination of the angles corresponding to these stages in the ROM. They are: $(\tan^{-1} 2^{-2} + \tan^{-1} 2^{-3})$, $(\tan^{-1} 2^{-2} - \tan^{-1} 2^{-3})$, $-(\tan^{-1} 2^{-2} + \tan^{-1} 2^{-3})$ and $-(\tan^{-1} 2^{-2} - \tan^{-1} 2^{-3})$. But the third and fourth terms are neumerically same as the first two terms (symmetry property) and it is sufficient to store only the first two terms in the ROM. We keep an one-bit signal associated to each of these conventional stages which flows through the pipeline along with the data. '0' or '1' on these signals represents +ve or −ve rotations respectively. These signals are nothing but the inverse of the MSB of y_{i-1} since a '1' at the MSB implies that the next stage (i^{th} stage) needs to execute a +ve rotation and vice versa. Similar arrangement is kept with the scaling-free stages. At each cycle the bits emerging from different sections of the pipeline are stored in the triangular array of registers (to keep the timing right) and they flow with the respective data as shown in Figure 1. However, the signals emerging from the scaling-free stages are treated separately according to the theory developed in Section II. The cyclic right shift is carried out at the last stage of the pipeline and the bit pattern is dumped in an accumulation register as shown in the Figure 1. The two MSB of the accumulation register, which carries the information about the direction of rotation of the conventional stages are fed into a simple address decoder logic to pick out the correct value from the ROM. It also generates a single bit signal that configures the adder/subtractor at the output in either addition or subtraction mode.

On top of the signals generated by each of the elementary rotational stages, the signals *quad* and *domain* also flows through the pipeline along with the data. Thus each data has a *token* attributed to it which tells the output unit about their initial states.

The total hardware requirement of the basic pipeline is 36 16-bit adders and 80 one-bit registers.

Output unit: The main hardware of the output unit consists of two adder/subtractors. One adder/subtractor is responsible for computing the actual accumulated angle by adding/subtracting the data from the ROM to/from the 12 LSB of the accumulation register whereas the other adder/subtratcor is used for carrying out the final step for computing the actual phase angle by adding/subtratcting $\pi/4$ to/from the accumulated angle.

Scaling unit: As has been mentioned earlier, incorporation of the classical CORDIC stage in the scaling-free formulation requires scale factor comparison. In this particular case the scale factor to be multiplied is 1.040201018. This factor is coupled with the scaling of $\sqrt{2}$ adaptively when the *quad* and *domain* signals indicate that the initial poistion of the vector was in the range $(\pi/8, 3\pi/8]$. The complete constant is realized using a shift-and-add technique with a couple of multiplexers for bypassing some of the stages.

IV. Implementation Results and Performance Evaluation

The 16-bit vectoring CORDIC processor is synthesized using 0.18μm CMOS library. The maximum achievable clcok frequency is 250 MHz at 1.8V supply. However for our target system is sufficient to run the processor at 20 MHz clock frequency. The overall synthesized area of the processor is 0.14 mm^2 of which the domain, basic pipeline and the output unit requires 0.027 mm^2, 0.126 mm^2 and 0.03 mm^2 area respectively.

Power dissipation has been analyzed using Synopsys' Prime power. At 20 MHz the processor consumes 700 μW power of which the basic pipeline consume 564 μW, and the domain and output unit consumes 58 μW and 85 μW respectively.

Although the 16-bit implementation of the proposed processor shows excellent area and power performance it is more interesting to evaluate its performance compared to the classical CORDIC processor for different wordlength. Figure 2 shows the comparison of hardware requirement of the proposed one in terms of adders compared to the classical CORDIC structure for different wordlength. In this comparison we have assumed that an n-bit comparator needs area of n/2-bit adder. Hardware involving the scaling circuitry is not considered here with the assumption that it is common to both the proposed one and the classical CORDIC. However, intuitively it can be said that the scaling circuit required for the present one is less than the classical one since it used less number of conventional elementary rotational stages. It can be seen from Figure 2(a) that the proposed design requires about 75% of the n-bit adders required for the classical CORDIC where n is the wordlength of the adder. Although in terms of n-bit adder the difference of adder requirement between the present one and classical one is more or less uniform but in reality considering an n-bit adder needs (n−1) full adder the difference becomes very significant with the increase of wordlength. Figure 2(b) shows the same comparison for required registers. Once again the saving here is about 3.2 times which becomes very significant with increasing wordlength. Figure 2(c) shows the comparison of size of n-bit ROM required. In this case the amount of ROM required for the present design is either less or comparable to that of the classical CORDIC processor up to 28-bit wordlength. However, beyond that the size of ROM for the

present design becomes significantly higher compared to that of the classical CORDIC.

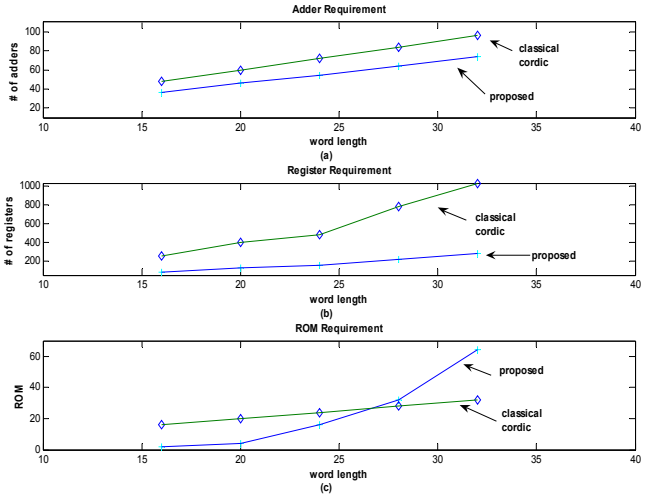

Figure 2: Hardware requirement comparison: (a) Adders; (b) Registers; (c) ROM

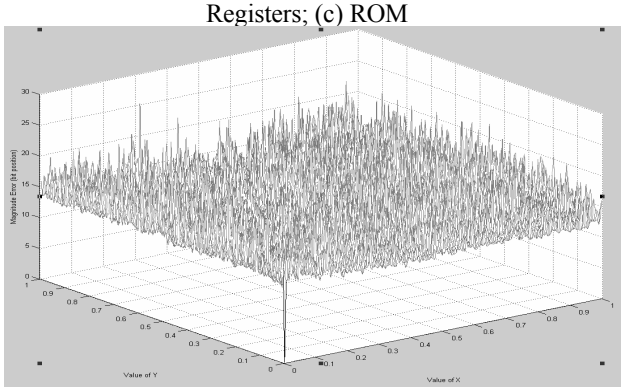

Fig 3(a) The magnitude error

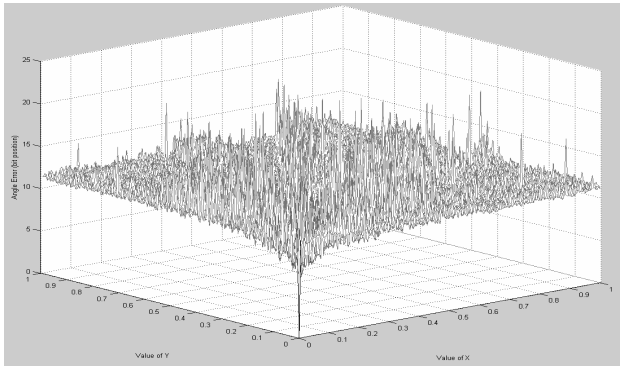

Figure 3(b) The angle error

The computational accuracy of the processor is shown in Figure 3 (a) and 3(b) for magnitude and angle respectively for the 16-bit processor. No additional attempt was made to minimize the error by using wider wordlength or by employing any normalization scheme since we are mainly interested to the performance of the system as it is. According to the theory developed in [9] for a classical vectoring CORDIC with 14 fractional digits (as used in our system) the worst case angle accuracy will be about eight bits and it will also depend on the initial values of the components of the vector. If the values of x or y component of the vector is close to 0 or 1 then the algorithm inherently exhibit high errors. Our simulation has been done using 40000 data where x and y are varied from 0 to 1. The nature of the error plot shows a comparable performance to the error characteristics of the classical CORDIC. It is to be noted that the error floor for magnitude computation is little bit higher than the expected. This is attributed to the fact that during the scaling operation the truncation error dominates where no attempt was made to minimize it by using wider wordlength.

V. Conclusions

In this article we propose a novel design of vectoring CORDIC processor. Its hardware cost is less than that of the conventional CORDIC. The complete elimination of the arithmetic processing for the z datapath makes its hardware cost less than that of the classical CORDIC and the hardware saving becomes significantly high as the wordlength increases. The algorithm proposed here shows similar error characteristic to that of the conventional CORDIC. The synthesis results for a 16-bit processor show that the proposed design occupies a very small area and consumes very low power.

References

[1] J. S. Walther, "A Unified Algorithm for Elementary Functions", Proc. Joint Spring Comput. Conf., vol. 38, pp. 379 – 385, Jul. 1971.

[2] H. Dawid and H. Meyr, "The Differential CORDIC Algorithms: Constant Scale Factor Redundant Implementation without Correcting Iterations", IEEE Trans. Comput., vol. 45, no. 3, pp. 307 – 318, 1996.

[3] D. Timmermann and S. Dolling, "Unfolded Redundant CORDIC VLSI Architecture with Reduced Area and Power Consumption", http://www-md.e-technik.uni-rostock.de/ma/dtim/vlsi97.pdf

[4] E. Antello, J. Villalba, J. D. Bruguera, E. L. Zapata, "High performance rotation architectures based on the radix-4 CORDIC algorithm", IEEE Trans. Comput., vol. 46, issue 8, pp. 855 – 870, Aug. 1997.

[5] M. Krstic, A. Troya, K. Maharatna and E. Grass, "Optimized Low-power Synchronizer Design for the IEEE 802.11a Standard", Proc.ICASSP'03, pp. II 333 – 336.

[6] K. Maharatna, A. Troya, M. Krstic, E. Grass and U. Jagdhold, "A CORDIC like processor for computation of arctangent and absolute magnitude of a vector", Proc. ISCAS 2004, Vol .2, pp. 713-16.

[7] K.Maharatna, A. Troya, S. Banerjee, E. Grass, "Virtually scaling-free adaptive CORDIC rotator", IEE Proc. : Computers and Digital Techniques, vol. 151(6), pp. 448 – 456. Nov. 2004.

[8] K. Maharatna, S.Banerjee, E. Grass, M. Krstic, A. Troya, "Modified virtually scaling-free adaptive CORDIC rotator algorithm and architecture". IEEE Trans. : Circuits and Systems for Video Technology, vol. 15(11) , pp.1463 – 1474. Nov. 2005.

[9] [9] K. Kota, J. R. Cavallaro, "Numerical accuracy and hardware tradeoffs for CORDIC arithmetic for special-purpose processors", IEEE Trans. Comput. Vol. 42(7), pp:769 - 779 , July 1993.