A NOVEL, OPTIMIZED CORDIC CORE FOR PHASE CORRELATION MOTION ESTIMATION

Andrea Molino, Fabrizio Vacca

CERCOM – Dipartimento di Elettronica
Politecnico di Torino – Corso Duca degli Abruzzi 24 – 10129, Torino (ITALY)
e-mail: `andrea.molino(fabrizio.vacca)polito.it`

ABSTRACT

This paper describes a CORDIC–based architecture to efficiently compute the phase difference between two complex numbers. The problem of fast phase difference computation is central in many signal processing algorithms. Our main focus has been posed on the Phase Correlation technique applied to Motion Estimation. A reduced complexity solution is proposed and specifically tailored to suit the application needs. The presented algorithm has been completely implemented in 0.25 μm standard–cell CMOS technology. As far as the performance are concerned the designed core outperforms a recently designed solution by more than 50% under area and energy standpoints.

1. INTRODUCTION AND PROBLEM STATEMENT

During the past fifteen years many researches have been directed towards the topic of accurate motion detection and estimation in digital video sequences [1]. Among all the possible motion estimation (ME) techniques, the so–called Block–Based Matching (BBM) method has gained an impressive diffusion. In conventional BBM the ME is carried out on fixed–size rectangular blocks (conventionally called macroblocks). The motion is described through a first–order, translational model. This means that each macroblock $b_{t_0}(x,y)$ in the reference frame $\mathscr{F}_{t_0}$ is associated with a two–dimensional motion vector $\mathbf{MV}_{i,j}$ able to describe the macroblock's motion with respect to the current frame $\mathscr{F}_{t_1}$.

Phase Plane Correlation (PPC) technique has been used for many years mainly in image registration field [2]. However, its use has been successfully reported also for motion estimation, mainly for its ability to measure large spatial displacements without sacrificing sub–pixel accuracy, its immunity to global illumination changes and its limited sensitivity to noise [3], [4]. PPC tends to produce fairly smooth and regular motion fields, leading to good quality motion compensated images [5]. From a theoretical standpoint, PPC directly exploits the shift property of the Fourier transform. Given two macroblocks $b_{t_0}(x,y)$ and $b_{t_1}(x,y)$, which differ over a moving area $\mathscr{A}$ only for translational displacement (d_x,d_y), their frequency representations are

This work was partially supported by the Italian Ministry of Research under the FIRB Grant.

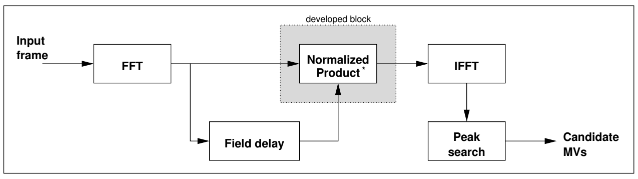

Fig. 1. Block scheme of PPC motion estimation.

related as:

$$B_{t_1}(\omega_x,\omega_y) = B_{t_0}(\omega_x,\omega_y)e^{j(\omega_x d_x,\omega_y d_y)} \tag{1}$$

where B_{t_1} and B_{t_0} are the Fourier–transformed (DFT) versions of the current and reference macroblocks respectively, while ω_x and ω_y are the frequency variables. If we define $\phi_{t_1}(\omega_x,\omega_y)$ and $\phi_{t_0}(\omega_x,\omega_y)$ as the phase information associated with the two DFTs, then their phase difference $\Delta\phi(\omega_x,\omega_y)$ can be expressed as:

$$e^{j(\Delta\phi(\omega_x,\omega_y))} = \frac{B_{t_1}(\omega_x,\omega_y)B^{\star}_{t_0}(\omega_x,\omega_y)}{|B_{t_1}(\omega_x,\omega_y)B^{\star}_{t_0}(\omega_x,\omega_y)|} \tag{2}$$

with $B^{\star}_{t_0}(\omega_x,\omega_y)$ being the complex–conjugate of B_{t_0}. It is so possible to directly determine the amount of translational motion occurred between $b_{t_0}(x,y)$ and $b_{t_1}(x,y)$: all it is needed is to take the DFT of the two macroblocks then evaluate equation (2) and, finally, take the inverse DFT finding the impulse position. Figure 1 depicts the logical sequence of the operations needed to compute the PPC–ME. The idea is to find a set of peaks for the current block: these peaks will be used to "drive" a second stage (not shown in the picture) which is essentially a BBM–ME [6].

In Table 1 we present system's requirement analysis when PPC–ME is considered. For each frame size, a macroblock size of 16×16 is considered. In the third column the total number of two–dimensional FFTs per second is reported, while in the rightmost one the required throughput for the phase difference part is shown. This number is crucial for our investigation: the direct evaluation of equation (2) requires 6 multiplications (4 for the complex multiplication and 2 for the square operation), 3 additions, 2 divisions (for the real and imaginary part respectively) and a square

Table 1. Required throughput for PPC–based motion estimation (25 fps case).

	MB/Frame	2D–FFT/s	$\Delta\phi(\omega_x,\omega_y)/\mathrm{s}$
QCIF	99	2475	$2.6\cdot 10^6$
CIF	396	9900	$10.2\cdot 10^6$
PAL	1620	40500	$41.5\cdot 10^6$

root operation. Unfortunately, multiplications, modulus and divisions are expensive and relatively inefficient operations for hardware implementations, especially when power dissipation and energy requirements are considered [7].

An handy method useful to estimate the computational complexity for a given algorithm is to evaluate the required number of elementary operations per second (EOPS), i.e. the number of n bit additions per second. Using this metric, an $n\times n$ multiplication costs n additions as well as a division operation. The square root (SQRT) operation is more difficult to evaluate. One can hypothesize to completely carry out the SQRT operation using a proper lookup table (LUT). Even if this would not be the case, we can try to neglect the SQRT computational complexity focusing only on the other parts of the algorithm. Under these assumptions the straightforward implementation of equation (2) requires $(8n+3)$ n bit additions per sample. If the FFT data are represented on 12 bits, this figure leads to more than 4.1 GEOPS (Giga EOPS) in case of PAL video. Even resorting to a high performance, 4–issue 1 GHz processor and assuming of being able to always fill the four execution units, a throughput of 4 GEOPS would be merely reachable. This means that even in a nearly–ideal case, such the one outlined above, the PPC–ME can be hardly sustained in real time by off–the–shelf embedded processors.

The main objective of this work is to try to mitigate the amount of operations needed to carry on the phase correlation computation. This is even more significant if one thinks that in the previous estimation the SQRT impact has been completely neglected. Starting from this preliminary overview of system requirements, we designed an efficient CORDIC–based solution, characterizing it from an hardware perspective. Finally the results from ASIC implementation are proposed and conclusions will be drawn.

2. TRADITIONAL CORDIC IMPLEMENTATION

It is well known that CORDIC algorithm performs a generic vector rotation through a finite number of simple rotations, whose amplitude decreases in time [8]. These CORDIC rotations are derived from the Cartesian rotation equations, that represent a rotation of a given vector $V = X_V + j\ Y_V$ through a given angle θ :

$$\begin{aligned} X_V^{new} &= \cos(\theta)[X_V^{old} - Y_V^{old}\ \tan(\theta)] \\ Y_V^{new} &= \cos(\theta)[Y_V^{old} - X_V^{old}\ \tan(\theta)] \end{aligned}$$

If the rotation angle is restricted so that $\tan(\theta) = \pm 2^{-i}$, the multiplication by $\tan(\theta)$ is reduced to a simple shift operation, so the CORDIC basic rotation stage can be derived:

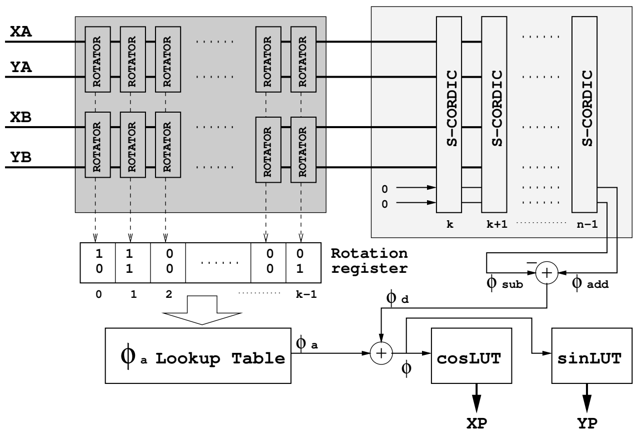

Fig. 2. LUT–CORDIC block scheme.

$$X_V^{new} = K_i(X_V^{old} - Y_V^{old}\ d\ 2^{-i}) \tag{3}$$
$$Y_V^{new} = K_i(Y_V^{old} + X_V^{old}\ d\ 2^{-i}) \tag{4}$$
$$z^{new} = z - d\ \tan^{-1}(2^{-i}) \tag{5}$$

where z is the angle accumulator that accumulates the rotation angles at each iteration, and d is the so called decision parameter that determine the rotation direction. Implementing the phase difference and the vector normalization in a traditional way requires three CORDIC operations. The first two are in vectoring mode which estimate the phase for each of the input complex terms A and B. After that, a third CORDIC (in rotation mode) rotates a unitary–modulus vector P by an angle which is the phase difference obtained from the first two CORDIC operations.

This first solution can be thought as a sort of "quick and dirty" architectural solution. Divisions and square roots have been avoided through the selection of CORDIC algorithm to compute the phase information. However significant area occupation and power dissipation can arise from the implementation of a fully pipelined solution based on this simple scheme.

3. PROPOSED CORDIC–BASED SOLUTION

The basic idea is to independently rotate the two terms A and B in vectoring mode, to align them with the x axis. If such rotations occur in the same direction, it means that no further information about their phase difference has been found, hence ϕ approximation does not need to be updated. Conversely, in case of opposite rotation it is important to record their phase difference. This difference is proportional to the current CORDIC step, and it is equal to twice the current rotation angle applied to A and B vectors.

In order to obtain a reduced complexity solution, the traditional CORDIC architecture is adapted to this scenario. A first idea would be to replace the phase accumulator in each step with a *rotation register*, wherein the CORDIC "decisions" are kept. In a second stage, it will be possible to uniquely obtain X_P and Y_P simply through a properly formatted LUT. Unfortunately the LUT size will exponentially grow when a significant number of CORDIC steps will be computed. To make the situation even worsen, two bits are actually needed for each step to represent both the rotation direction and the angle sign. This fact doubles the number of bits to be stored in the rotation register, squaring the final LUT size.

To overcome this problem, we propose a dual–step approach. A first block provides a rough evaluation of phase difference using the aforementioned method, performing k steps. Then a second stage refines the estimation mainly exploiting a more traditional CORDIC approach (with phase accumulator). As shown in the literature [9] after few steps (i.e. 5 or 6) equation 5 can be approximated as:

$$z^{new} = z - d\ (2^{-i}). \tag{6}$$

Starting from this consideration, we can note how the actual angle needs to be accumulated only during the very first steps. For this purpose we removed the accumulator using the rotation register.

Observing the second stage, we note how each CORDIC step contributes to the final phase value in one of the following way:

$$\phi_d(i+1) = \begin{cases} \phi_d(i) & \text{if vectors rotate accordingly} \\ \phi_d(i) + 2^{-i} & \text{if approx. is increased} \\ \phi_d(i) - 2^{-i} & \text{if approx. is decreased.} \end{cases} \tag{7}$$

Since $\phi_d(k) = 0$, after $n-k$ steps the value of ϕ_d will be:

$$\phi_d = \sum_{i=k}^{i=n-1} \alpha_i 2^{-i} - \sum_{i=k}^{i=n-1} \beta_i 2^{-i} \tag{8}$$

where α_i and β_i are boolean values expressing equation 7. It is so possible to “remember” independently α_i and β_i using two registers, combining them only at the end of the computation. The logic required to update α and β registers is much simpler than an accumulator. Finally ϕ_a and ϕ_d are combined together to obtain the estimated value for the phase difference ϕ.

Figure 2 shows the whole architecture for this method. As it can be seen, the architecture can be divided in three different parts: a dark region representing the coarse approximation pipeline for the target angle ϕ, a lighter region on the right devoted to the fine approximation and a final stage where sine and cosine values are finally gathered.

4. VLSI IMPLEMENTATION RESULTS

In this section we propose the hardware implementation of the presented method, which has been completely implemented resorting to a VHDL description. During the implementation phase, particular emphasis has been posed in order to maintain the core as much scalable as possible. The results from the design space explorations are reported in Figures 3 and 4 respectively. For the sake of clarity, the data reported in these figures are relative to an input width

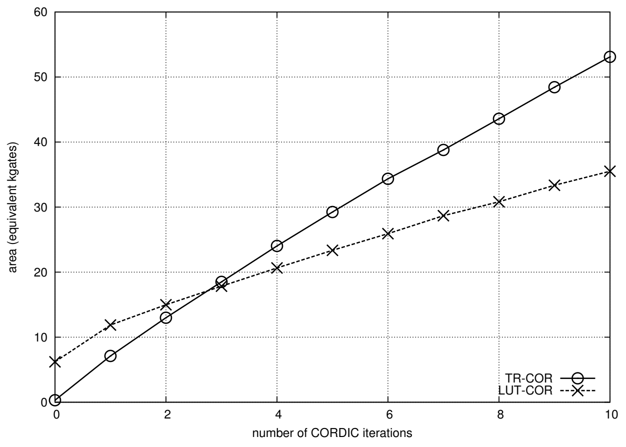

Fig. 3. Area occupation data for the proposed blocks.

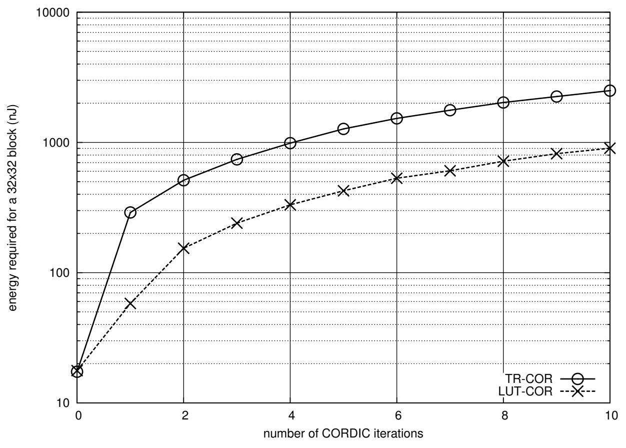

Fig. 4. Implementation results: energy required to process a single MB.

of 12 bits. Looking at Figure 3 it is possible to observe how the traditional solution (TR–COR) requires a remarkable amount of area when compared with the proposed one. In particular it has been possible to achieve an area gain of 39% with respect to TR–COR.

As previously stated, the logical synthesis has been carried out using a 0.25μm CMOS technology from ST Microelectronics. Figure 4 reports the energy required to completely process a 32×32 block. As for the area case, also when power dissipation is taken into account LUT–based solution compares favorably against TR–COR one. As an example, when 10 iterations are considered, the employment of a LUT–COR block corresponds with a saving of more than 48% in terms of energy dissipation.

Lastly it is interesting to compare these solutions with existent works in the literature. In particular, D. D. Hwang et al. [10] recently reported the design and the implementation of a enhanced CORDIC–like core suitable for fast rectangular to polar coordinates conversion. In Table 2 we report a comparative summary for the studied solutions and for the one presented in [10]. In order to make the comparison process as fair as possible, we exploit the flexibility of our methodology extracting all the data for the 12 bit, 12 iterations case. On the first two rows we present the traditional CORDIC and the proposed LUT–COR while on third one the solution from Hwang er al. is dealt with. It must be noted that the core presented in [10] covers only a single CORDIC unit. This means that we need two instances of the same core to perform the phase extraction, followed by a traditional CORDIC for the polar–to–rectangular conversion. For this reason, in the last row we propose an estimation for a solution based on [10]. As a comment for this comparison it is possible to observe how LUT–COR compares favorably also against this high–performance solution. This figure is particularly significant under the power standpoint where our solution outperforms the competitor's one by more than a factor of two.

Table 2. Performance summary: ST 0.25μm@2.5V, 12 bits, 12 iterations.

	Area (kgates)	**Throughput (MSampl/s)**	**Power (mW/MHz)**
TR-COR	60	400	3.9
LUT-COR	43	500	1.5
[10] (1 COR)	25	407	1.15
[10] (estim.)	70	400	3.7

5. CONCLUSIONS

In this paper proposes a novel technique to compute the phase difference between two complex numbers. CORDIC algorithm has been optimized and tailored to obtain a very low-complexity solution, suitable for high–end motion estimation applications. From the experimental results it has been possible to assess how the LUT–COR core represents a significant improvement with respect to existent architectures and accelerators.

As far as future developments are concerned, a complete prototype of a PPC–ME chip is currently under development.

6. REFERENCES

[1] A. Murat Tekalp, *Digital video processing*, Prentice-Hall, Inc., 1995.

[2] C. D. Kuglin and D. C. Hines, "The phase correlation image alignment method," in *Proc. IEEE International Conference on Cybernetics and Society*, San Francisco, 1975, pp. 163–165.

[3] G. A. Thomas, "Television motion measurement for DATV and other applications," Tech. Rep., BBC RD, 1987.

[4] T. Vlachos and G. Thomas, "Motion estimation for the correction of twin-lens telecine flicker," in *Proc. IEEE International Conference on Image Processing*, 1996, pp. 109–112.

[5] M. Biswas and T. Q. Nguyen, "A novel de-interlacing technique based on phase plane correlation motion estimation," in *Proc. IEEE International Symposium on Circuits and Systems*, May 2003, pp. (II)604–607.

[6] B. Girod, "Motion-compensating prediction with fractional-pel accuracy," *IEEE Trans. Commun.*, vol. 41, pp. 604–612, Apr. 1993.

[7] M. D. Ercegovac, T. Lang, J. M. Muller, and A. Tisserand, "Reciprocation, square root, inverse square root, and some elementary functions using small multipliers," *IEEE Trans. Comput.*, vol. 49, no. 7, 2000.

[8] J. Volder, "The cordic trigonometric computing technique," *IRE Trans. Electron. Comput.*, vol. 3, pp. 330–334, 1959.

[9] S. Wang, V. Piuri, and E. E. Swartzlander, "Hybrid cordic algorithms," *IEEE Trans. Comput.*, vol. 46, no. 11, pp. 1202–1207, 1997.

[10] D. D. Hwang, Fu Dengwei, and Jr. A. N. Willson, "A 400-MHz processor for the conversion of rectangular to polar coordinates in $0.25\mu m$ cmos," *IEEE J. Solid-State Circuits*, vol. 38, pp. 1771–1775, Oct. 2003.