N94-18358

3rd NASA Symposium on VLSI Design 1991

6.2.1

Direct Kinematics Solution Architectures for Industrial Robot Manipulators: Bit-Serial Versus Parallel

J. Lee
Department of Electrical Engineering
University of Houston
Houston, TX 77204-4793

K. Kim
Superconducting Super Collider Lab.
2550 Beckleymeade Avenue
Dallas, TX 75237

Abstract - **This paper investigate a VLSI architecture for robot direct kinematic computation suitable for industrial robot manipulators The Denavit-Hartenberg transformations are reviewed to exploit a proper processing element, namely an augmented CORDIC. Specifically, two distinct implementations are elaborated on, such as the bit-serial and parallel. Performance of each scheme is analyzed with respect to the time to compute one location of the end-effector of a 6-links manipulator, and the number of transistors required.**

1 CORDIC Techniques

The matrix A_j describing the jth link is proposed to be implemented via 4 CORDICs: parallel two for the w-axis operation, and another parallel two for the x-axis. Since, the rotation and translation are disjoint each other, the 4 CORDIC can be done via a 2-stages cascade [5].

Let the jth joint orientation vector denote by p_j, where $p_j = A_j p_{j-1}$. Consider an intermediate vector p_j^A, between p_j and p_{j-1}:

$$p_j = Trans(w_{j-1}, d_j) Rot(w_{j-1}, \theta_j) p_j^A : stage - 1 \quad (1)$$

$$p_j^A = Trans(x_j, a_j) Rot(x_j, \psi_j) p_{j-1} : stage - 2. \quad (2)$$

One set of transformations for each stage, i.e. $Trans(w, d)Rot(w, \theta)$, is a block-diagonal matrix and can be orthogonally implementable by two 2x2 matrix transformations. Note that is implementable through an augmented PE, rather two different PEs, observing that $Trans(w, d)$ is a trivial operation. Then,

$$p_j = \begin{bmatrix} x_j \\ y_j \\ . \\ w_j \\ 1 \end{bmatrix} = \left| \begin{array}{ccc} Rot(w_j, \theta_j) & : & O \\ . & . & . \\ O & : & Trans(w_j, d_j) \end{array} \right| p_j^A. \quad (3)$$

p_j (also, similarly for p_j^A) is decomposed into two blocks, e.g the first two elements of p_j becomes one vector X_j:

$$p_j = [X_j; w_j, 1]^t = [Rot(w_j, \theta_j); w_j + d_j, 1], \tag{4}$$

where w_j is for the w-axis component of the vector p_j, and X_j for x- and y-axis components rotated by θ_j. In a similar way, for p_j^A we can choose a rotated vector of y- and w-axis disjointly through axis shuffling. Finally, consecutive n-pairs of rotation and translation can be implemented via a 2n-stages cascade. We will name each stage as a macro-PE (or, an augmented PE), which can be 2n-pipelined to compose an n-links computation processor. Not to differentiate the two different sets of transformations, w-axis and x-axis respectively, we employ index i in unified notations for a macro-PE: for a reference axis w_i,there are rotation of θ_i and translation of d_i, $X_{i-1} = (x_{i-1}, y_{i-1})$ for an input, and $X_i = (x_i, y_i)$ for an output.

Each macro-PE including one $Trans(w_i, d_i)$ and one $Rot(w_i, \theta_i)$ can be implemented, as in Figure 1.a. One-joint processor is shown in Figure 1.b. Finally, for a 6-joints system, Figure 1.c shows a fully pipelined structure.

From this point, we will concentrate on implementation of a macro-PE. Observing that Rot and $Trans$ functions are disjoint each other, let us isolate the rotation part at first. This vector rotation for $X_i = (x_i, y_i)$ by the angle θ_i can be realized by an iteration algorithm called CORDIC [4] instead of computing trigonometric functions and applying matrix multiplication. CORDIC realizes a vector rotation by a partial sum of micro-angle rotations with a pre-fixed sequence of angles. When the rotation macro-angle is represented as a sum of decomposed micro-angles, i.e $\theta_i = \sum_{k=0}^{n} \theta_{i,k}$,

$$X_i = \prod_{k=0}^{n} k_k \begin{bmatrix} 1 & -tan\theta_{i,k} \\ tan\theta_{i,k} & 1 \end{bmatrix} X_{i-1} \tag{5}$$

where $k_k = cos\theta_{i,k}$ is a micro-scale composing a final scale factor, explained later. Such a specific form of the pre-fixed micro-angle sequence as $\tan^{-1} 2^{-i}$, is attractive for VLSI implementation since it is composed only of additions, shiftings, and a arctangent lookup table For the simplicity of notation, subscript i indexing a certain stage will be omitted, and X, Y and Z stand for abridged notations for those having subscript i.

Non-redundant : The micro-iterations of the conventional (hereafter, it will be called non-redundant) CORDIC are 3 linear recursive equations: X recurrence (X-rec.), Y-recurrence (Y-rec.) and Z-recurrence (Z-rec.) [4].

$$\begin{aligned} X[i+1] &= X[i] + \sigma_i 2^{-i} Y[i] \\ Y[i+1] &= Y[i] - \sigma_i 2^{-i} X[i] \\ Z[i+1] &= Z[i] - \sigma_i \tan^{-1} 2^{-i} \end{aligned} \tag{6}$$

With an initial value of $Z[0] = \theta_i$, CORDIC rotates initial values of $X[0]$ and $Y[0]$, to the last value $X[n]$ and $Y[n]$, while making $Z[i]$ close to zero, so that $Z[n]$ is forced to be zero. With n number of iterations, n-bit accuracy of X and Y in the output can be achieved.

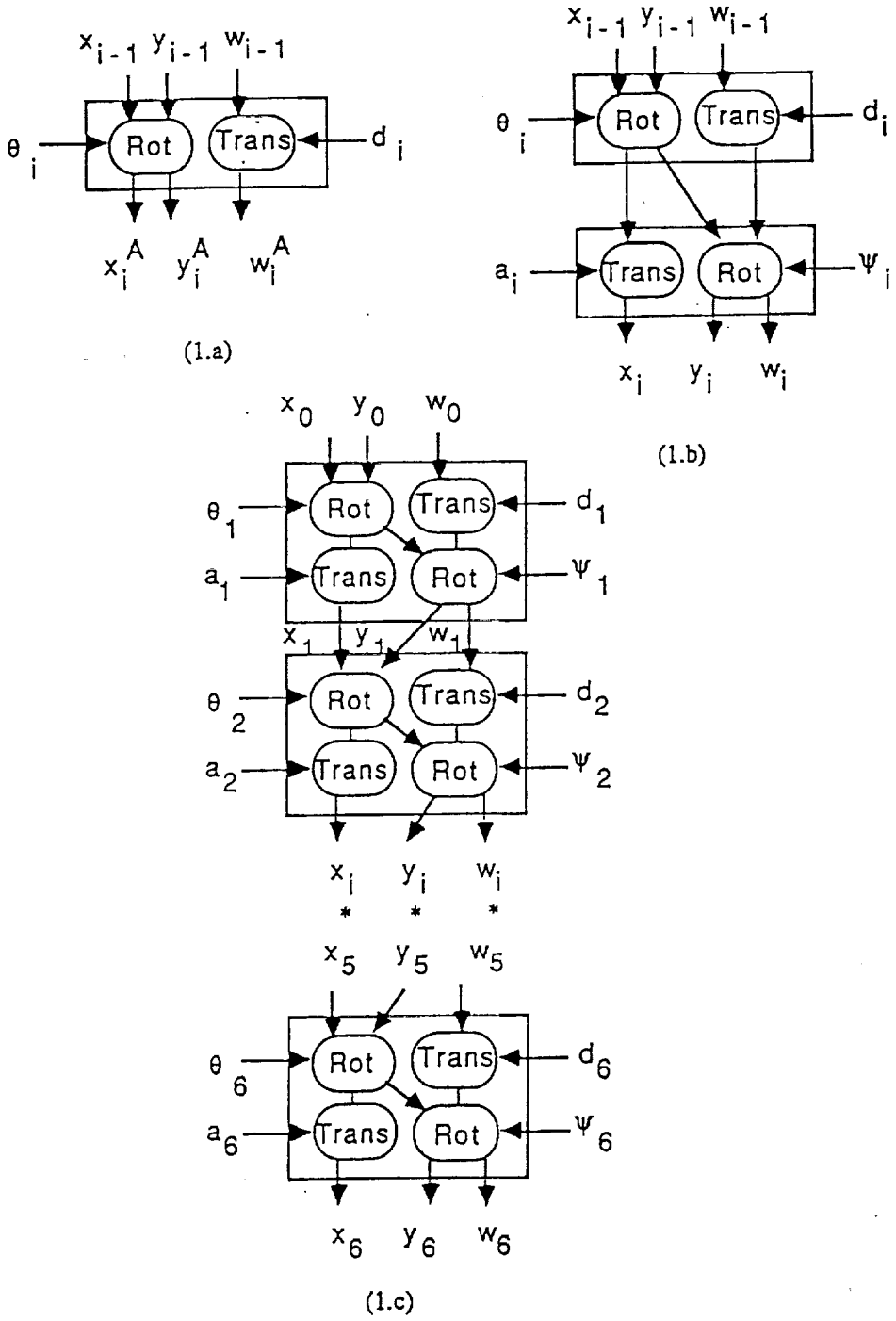


Figure 1: CORDIC-based Pipelined Architecture for Direct Kinematics Computation: a. A macro-PE, One-stage from an orientation to an intermediate, b. 2-stages cascade, An A_i transformation module for a link, c. A complete pipelined Computation Module for 6-links system.

For a known angle, the direction of the rotation, σ_i can be pre-computed or calculated one by one on-the-fly using the following selection function.

$$\sigma_i = \begin{cases} 1 & \text{if } Z[i] \geq 0 \\ -1 & \text{if } Z[i] < 0 \end{cases} \tag{7}$$

The CORDIC rotation does not preserve the input norm. To get a rotated vector having the same length as the input $(X[0], Y[0])$, $X[n](Y[n])$ needs to be compensated by a scaling factor K

$$K = \frac{\|[X[n], Y[n]]^t\|}{\|[X[0], Y[0]]^t\|} = \prod_{i=0}^{n-1} \sqrt{1 + \sigma_i^2 2^{-2i}}, \tag{8}$$

where $\|\cdot\|$ stands for the norm of the vector. Note that K is *constant* for the non-redundant scheme since σ_i is in {-1, 1}.

Redundant : Non-redundant CORDIC is slow inherently with delay of $O(n^2)$ due to its recursiveness and serial dependency, since a micro-rotation with delay $O(n)$ should be finished before processing the next micro-rotation. Delay performance of a macro-rotation (n micro-rotations) can be improved from $O(n^2)$ to $O(n)$ by using redundant arithmetic (carry-free addition such as carry save or signed-digit addition) to determine the direction of the rotation $\hat{\sigma}_i$, based on an estimate instead of an exact value [9]. The redundant arithmetic gives a delay of $O(1)$ instead of $O(n)$, and the estimation of direction is necessary not to erode the advantage of $O(1)$. This requires the modification of the recurrences and selection function. This redundant CORDIC scheme produces the output about 4 times faster than the non-redundant. However, it introduces additional cost since the scale factor K is variable depending on a macro-angle by allowing $\hat{\sigma}_i$ to be in {-1, 0, 1}.

Constant-Factor-Redundant : To reduce implementation cost of redundant CORDIC, it would be good to have a constant scale factor by forcing $\hat{\sigma}_i$ in {-1, 1}. However, since $\hat{\sigma}_i$ is determined from an estimate, there arises a convergence assurance question. There was proposed a scheme appending correcting iteration stages at proper positions [10]. Along to this idea, the number of extra correcting iterations is further reduced by dividing the micro-iterations (for $i = 0$ to $i = n - 1$) into two groups: one group where the direction of the rotation is in {-1, 1} for $i = 0$ to $i = n/2$ and the other in {-1, 0, 1} for $i = (n + 1)/2$ to $i = n - 1$ correcting iterations by 50 % since correcting iteration is not needed for the second half of the micro-iterations and we still obtain a constant scale factor K since the value of K in n-bit precision does not depend on the $\hat{\sigma}$ value for $(n + 1)/2 \leq i \leq (n-1)$. Z-recurrence also can be modified so that $\hat{\sigma}_i$ is determined quickly by looking at a few most significant bits. This new scheme is called Constant-Factor-Redundant-CORDIC(CFR-CORDIC). The modified recurrences and selection functions for the scheme are described below.

$$\begin{aligned} X[i+1] &= X[i] + \hat{\sigma}_i 2^{-i} Y[i] \\ Y[i+1] &= Y[i] - \hat{\sigma}_i 2^{-i} X[i] \\ U[i+1] &= 2(U[i] - \hat{\sigma}_i 2^i \tan^{-1} 2^{-i}) \end{aligned} \tag{9}$$

where $U[i]$ is for the implementation simplicity, which is equal to $2^i Z[i]$, and the selection function is given as follows:

$$\hat{\sigma}_i = \begin{cases} 1 & \text{if } \hat{U}[i] > 0 \\ & \text{or } \hat{U}[i] = 0 \cap i < n/2 \\ 0 & \hat{U}[i] = 0 \cap i \geq n/2 \\ -1 & \text{if } \hat{U}[i] < 0 \end{cases} \tag{10}$$

When t fractional bits are used in the estimate value, i.e., $\hat{U}[i]$ is computed using t fractional bits of redundant representation of $U[i]$, the following correcting iteration need to be included, where the interval between indexes of correcting iterations should be less than or equal to $(t-1)$ up to the last iteration index equal to $n/2$. When the correction stage is necessary at the jth step of micro-iteration,

$$U^C[j+1] = U[j+1] - 2\hat{\sigma}_j^C 2^j tan^{-j} 2^{-j} \tag{11}$$

with the direction of the rotation $\hat{\sigma}_j^C$ determined from the same selection function of eq.(10), except being decided based on $\hat{U}[j+1]$ instead of $\hat{U}[i]$.

So far, we discussed about recursive structures of several CORDIC schemes to implement the rotation part in the basic PE, as depicted in Figure 1. The PE, augmented by a translator, necessitates scaling operation at each stage, because shuffling of the output at each stage makes continuous accumulation of the scaling factor complex to be processed at the final stage. The scaling operation has been solved either by an explicit way or an implicit. The explicit way is dividing the rotated vector by a constant, which is known for the non-redundant, to be calculated while running the micro-steps of CORDIC [4,9]. The division can be processed by another CORDIC (in a linear mode) or a divider. The implicit approach reconfigures the sequence of micro-iterations of the CORDIC, eventually to have a different norm from that without scaling micro-iterations. Scaling micro-iterations target in general at making the adjusted scaling factor in a form of 2^i or 1, which can be easily set to the unit size. Each micro-iteration can be composed of i) reduction axis-scaling [11], ii) repetition of vector-scaling, iii) expansion axis-scaling or combinations thereof [12]. Relevant issues regarding solution search are to be further studied, more than the greedy method or the decomposed [13]. In summary, the explicit scaling almost doubles the system complexity, while the implicit increases 25 % for the non-redundant and about 30 % for the redundant.

2 Application to Direct Kinematics

In this section, we design an architecture for the direct kinematics computation, based on CFR-CORDIC. The data-path is the parallel. To analyze its performance, we will define a new measure, namely one-position calculation time. Via this measure, we will also analyze performance the bit serial architecture similarly implementable as in

2.1 Performance Measure

Let's define the following parameters.
b_i : the number of bits in each input x, y and w
b_f : the number of bits in each output
n_f : the number of links (=6)
f_c : the available data shift rate
Δ : the step time per micro CORDIC iteration
f_i : the input bit rate
Additionally, we define a measure parameter T_Δ,

$$T_\Delta = \text{step-time}(\Delta) * \text{number of steps},$$

to compare the performance of various schemes. For a discrete element implementation, Δ corresponds to one single external clock time $1/f_c$. Note that Δ varies depending on a particular implementation of a macro-PE. Without loss of generality, let's define the unit of Δ to be 1 for one-bit full addition time. The input processing rate can be alternatively interpreted as

$$\frac{f_i}{b_i} < \frac{1}{T_{\Delta_1}}, \tag{12}$$

which limits the maximum rate of input vector sampling to be processable through an implemented processor.

2.2 Performance Comparison

Bit Serial: A macro-PE using serial data path and arithmetic units for CORDIC is shown in Figure 2 [6]. Figure 2.a shows symmetric components of a bit-serial PE in x, y and w representation, and Figure 2.b is for the detail of each block (X-recurrence or Y-recurrence) employing bit serial arithmetic. W-recurrence is in Figure 2.c, and Z-recurrence in Figure 2.d. The x and y components of the input vector X_{i-1} are taken initially as $X[0]$ and $Y[0]$, and the initial angle $Z[0]$ is set to the corresponding joint angle. After performing n micro-iterations, CORDIC produces n-bit precision outputs leading to X_i.

In the serial scheme without macro-pipelining, denote a basic step-time as Δ_1, which is equivalent to Δ. To use one adder recursively n_f times to process an n_f links,

$$T_{\Delta_1} = \Delta_1 * n_f(b_f + b_i(b_i + log_2 b_i)),$$

where the output has b_f bits buffer.

CFR-Redundant Parallel : To increase the throughput of the previous, the bit-serial PEs can be substituted by those using parallel arithmetic. When parallel arithmetic and non-redundant CORDIC are adopted, the corresponding parameter becomes

$$T_{\Delta_2} = \Delta_2 * n_f(b_i + log_2 b_i)$$

where Δ_2 equals to the time for one micro-rotation (time for variable shifter plus time for carry-propagate addition), approximately $2\log_2 b_i$ assuming fast variable shifter and carry-propagate adder. The step time can be further shortened by adopting CFR-CORDIC,

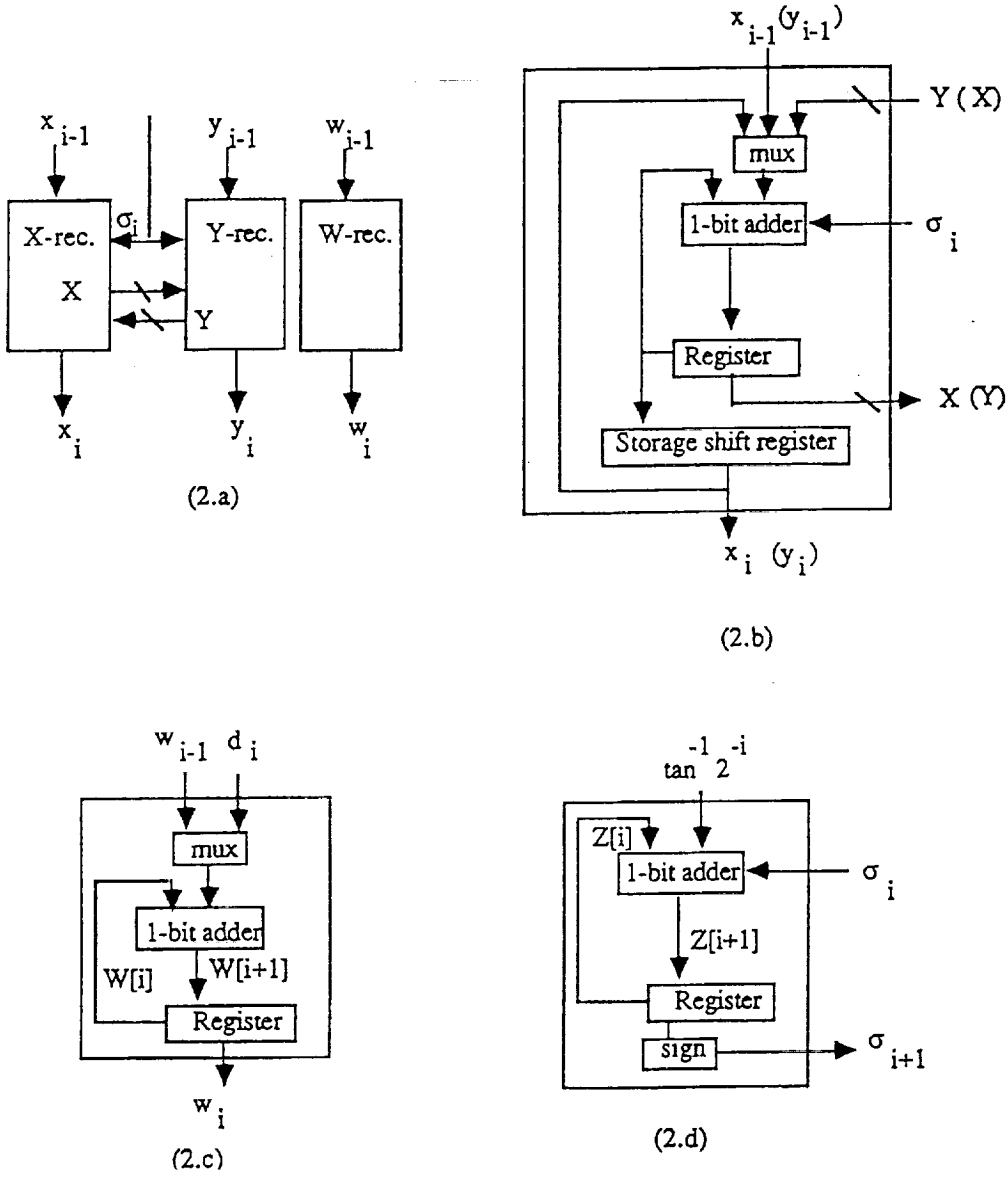
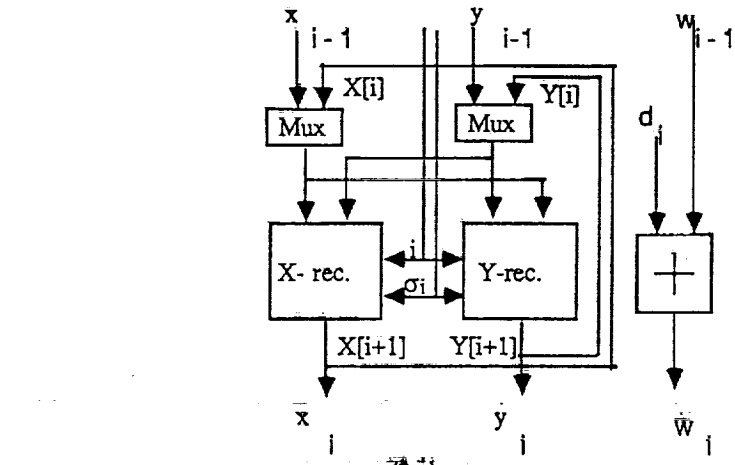


Figure 2: A bit-serial PE : a. A macro-PE with X-, Y- and W-recurrence, b. Detail of either block, c. W-recurrence, d. Z-recurrence.

where a carry-free adder (signed-digit adder) is replaced for carry-propagate adder. Figure 3.a shows a macro-PE in components, and Figure 3.b is for the detail of each block (X-recurrence or Y-recurrence) employing parallel/redundant arithmetic. Z-recurrence is in Figure 3.c.



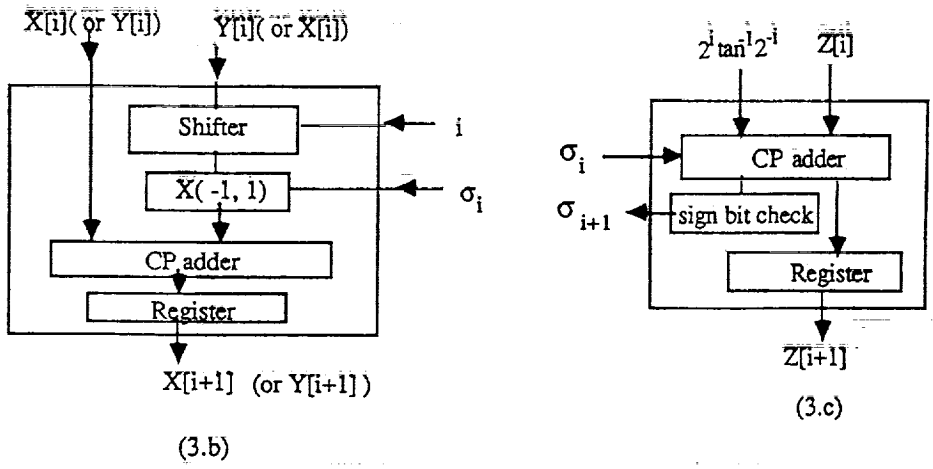


Figure 3: A parallel/redundant PE: a. A macro-PE with X- and Y-recurrence, b. Detail of either block, c. Z-recurrence.

Description	Δ_i/Δ	T_{Δ_i}	Processing rate	TRs estimate
Bit-serial (parallel)	1	1200Δ	600K 4M	2K 12K
Parallel(CFR) (parallel)	5	500Δ	2M 10M	6K 40K

Table 1: Time and complexity comparison

In this case, the sign of $Z[i]$ at the ith micro-iteration can not be detected by looking at the most significant bit since $Z[i]$ is in redundant number representation. To determine the sign of $Z[i]$ quickly by looking at a few significant bits, CFR-CORDIC uses an estimate of shifted-$Z[i]$ ($U[i]$) using t fractional bits. As discussed earlier, the number of fractional bits used for the estimate also determines the frequency rate of a correcting iteration: more fractional bits are used, less number of correcting iterations are required. Let the number of correcting iterations denoted by η. The corresponding T_{Δ_3} becomes

$$T_{\Delta_3} = \Delta_3 * n_f(b_i + log_2 b_i + \eta)$$

where Δ_3 equals to the time for carry-free addition plus the time for the maximum of a selection function and a variable shifter, approximately $(1 + log_2 b_i)$. Note that a practical number of correcting iterations is much smaller than b_i, e.g. 1 for the 16bit resolution. Hence, we can approximate T_{Δ_3} to be that for the redundant without a correcting iteration.

For a case, $b_i = 12$, $b_f = 16$, the estimated T_Δ is summarized in Table 1. To get first order estimates of available speed and area, we use a figure that one full adder (also one bit shifter) requires approximately 50 TRs and one 20nsec clock cycle [14].

3 Conclusion

We have examined various kind of CORDIC schemes as a macro-PE module for the direct kinematics processor, and showed that its micro-level regularity is suitable for VLSI implementation, depicted along with specific schematics which include the conventional non-redundant, the redundant and the Constant-Factor-Redundant schemes. The cost-effectiveness of selected architectures has been analyzed using bit-serial, parallel or pipelined structure with respect to the time and the number of modules required, to compute one location of the end-effector for a 6-links manipulator, given a set of angle measurements The comparison table exhibits the CORDIC-based robotics processor as a prospective solution in VLSI to be used for a wide range of kinematics calculation requirement, compromising the size versus speed.

References

[1] J. Denavit and R. Hartenberg, "A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices," *Journal of Applied Mechanics,* pp.215-221, 1955.

[2] P. Nanua, K. Waldron and V. Murthy, "Direct Kinematic Solution of a Stewart Platform," *IEEE Trans on Robotics and Automation,* Vol 6, No 4, pp.438-444, Aug. 1990.

[3] D. Moldovan and G. Lee, "On the Use of Parallel Architectures for Robotic Manipulators: The Kinematics Problem," *Int. J. Robotics and Automation,* Vol 1, No 2, pp.47-53, 1986.

[4] J. Walther, "A Unified Algorithm for Elementary Functions," *AFIPS Spring Joint Computer Conference*, pp.379-385, 1971.

[5] C. Lee, "CORDIC-based Architectures for Robot Direct Kinematics and Jacobian Computation," *3rd Int. Symp. Intelligent Control,* pp.609-614, 1988.

[6] R. Harber et. al.,"Bit-serial CORDIC Circuits for Use in a VLSI Silicon Compiler," *Int. Conf. Circuit and System*, pp.154-157, 1989.

[7] M. Kameyama, T. Matsumoto and H. Hideki, "Implementation of a High Performance LSI for Inverse Kinematics Computation," *IEEE Int. Conf. Robotics and Automation,* pp.757-762, 1989.

[8] H. Kung, "Let's Design Algorithms for VLSI systems," *Caltech Conf. VLSI*, pp.65-90. 1979.

[9] M. Ercegovac and T. Lang, "Redundant and On-Line CORDIC: Application to Matrix triangularization and SVD," *IEEE Trans. on Computers*, Vol. C-39, No 6, pp.725-740, June 1990.

[10] N. Takagi, T. Asada and S. Yajima, "Redundant CORDIC methods with a constant scale factor for sine and cosine computation", Submitted to IEEE Trans. on Computers, 1989.

[11] G. Haviland and A. Tuszynski, "A CORDIC Arithmetic Processor Chip," *IEEE Trans. on Computers,* Vol C-29, No 2, pp.68-79, Feb. 1980.

[12] J. Delosme, "VLSI Implementation of Rotations in Pseudo-Euclidean Spaces," *Proc. of ICASSP,* pp.927-930, 1983.

[13] J. Lee and T. Lang, "Matrix triangularization by fixed-point redundant CORDIC with a constant scale factor," *Proc. SPIE Conference on Advanced Signal Processing Algorithms, Architectures, and Implementations*, July 1990.

[14] J. Harding, T. Lang and J. Lee, "A Comparison of Redundant CORDIC Rotation Engines," *Int. Conf. Computer Design 91*, Oct. 1991.