# CORDIC and SVD implementation on FPGA

Ha Van Phu, Dinh Trong Huy, Tran Thi Thanh Huyen,
*Hanoi University of Science and Technology*

***Abstract*—Singular value decomposition (SVD) allows the factorization of real or complex matrices providing quantitative information with fewer dimensions along which data points exhibit more variation. These days SVD computation is being used in numerous applications, and because of its importance, different approaches for SVD hardware computation have been proposed; however, their application is limited by the inherent SVD calculation complexity making it possible to analyze up to $8 \times 4$ matrices until now. This paper presents a hardware architecture for SVD computation utilizing CORDIC. Fixed point arithmetic was considered. The concepts were implemented in Verilog, verified and synthesized with Xilinx tools. Selected approach was physically implemented and tested.**

***Index Terms*—CORDIC, SVD, digital, hardware, Verilog, FPGA.**

## I. Introduction

Processing of matrices, especially inversion remains a key challenge for contemporary computing machines. Very smart algorithms were proposed many years ago, by the scientists who expected rapid development of digital hardware in the future. Many of those solutions were presumed to work on futuristic parallel devices. Eventually recent years have brought the long expected rapid development of digital hardware and growth of programmable logic devices complexity. There is growing interest in construction of dedicated digital hardware. This paper describes a study of hardware implementation of Singular Value Decomposition of matrix based on CORDIC modules. The authors focus on comparison of architecture variants in the context of resource allocation, speed and accuracy. An implementation of an SVD hardware for $8 \times 4$ matrices over complex fixed-point signed fraction data, based on the CORDIC algorithm It uses parallel-architecture-supported CORDIC cores that are suitable for streamed data processing. A prototype was implemented on FPGA development boards (Xilinx $XC6SLX45$).

The rest of the paper is organized as follows. A description of the SVD algorithm and CORDIC algorithm are given in section 2. SVD computation algorithm is presented in section 3. The implementation details and analysis are given in section 4 and 5. Conclusion and future work are given in section 6.

## II. Theoretical Background

### A. SVD algorithm

The singular value decomposition (SVD) of an $m \times n$ matrix A is defined by

$$A = U\Sigma V^{\mathrm{T}} \tag{1}$$

where $U$ and $V$ are orthogonal matrices of $m \times m$ and $n \times n$, respectively, i.e. $UU^T = I_m$, and $VV^T = I_n$ where $I$ is identity matrix, and $\sigma$ is a diagonal matrix such that $\sigma = diag(\delta_1, \delta_2, \ldots, \delta_n)$, where $\delta_1$, $\delta_2$,..., $\delta_n$ are the singular values of $A$. For $m > n$, at least $m - n$ singular values will be zero. If $r = rank(A)$ and $r < n$, $r$ of the singular values will be non-zero. Matrix $U$ contains $m$ left singular vector, and matrix $V$ contains $n$ singular vector. Most algorithms for the SVD are based on diagonalizing rotations since they are the simplest form of orthogonal transformations that preserve angles and lengths.

### B. CORDIC algorithm

The CORDIC algorithm (Coordinate Rotation Digital Computer) provides an iterative method of performing vector rotations by arbitrary angles using only shifts and adds. The algorithm is derived from the Givens Rotation transform:

$$\begin{cases} x' = xc_\theta - ys_\theta \\ y' = yc_\theta + xs_\theta \end{cases} \tag{2}$$

$$\Leftrightarrow \begin{cases} x' = cos\theta.(x - ytan\theta) \\ y' = cos\theta.(y + xtan\theta) \end{cases} \tag{3}$$

If the rotation angle is restricted so that $tan(\pi) = \pm 2^{-i}$, the multiplication by the tangent term is reduced to simple shift operation. Arbitrary angles of rotation are obtainable by performing a series of successively smaller elementary rotations. If the decision at each iteration, i, is which direction to rotate rather than whether or not to rotate, then the $cos(\delta_i)$ term becomes a constant (because $cos(\delta_i) = cos(-\delta_i)$). The iterative rotation can now be expressed as:

$$\begin{cases} x_{i+1} = K_i(x_i - s_i.y_i.2^{-i}) \\ y_{i+1} = K_i(y_i + s_i.x_i.2^{-i}) \end{cases} \tag{4}$$

Where:

$$K_i = cos(tan^{-1}(2^{-i})) = \frac{1}{\sqrt{1 + 2^{-2i}}} \tag{5}$$

$$s_i = \pm 1 \tag{6}$$

Removing the scale constant from the iterative equations yields a shift-add algorithm for vector rotation. The product of the Ki's can be applied elsewhere in the system or treated as part of a system processing gain. That product approaches 0.6073 as the number of iterations goes to infinity:

$$K = \prod_{k=0}^{\infty} \frac{1}{\sqrt{1 + 2^{-2i}}} = 0.60725 \tag{7}$$

Then, the CORDIC equations are:

$$\begin{cases} x_{i+1} = x_i - s_i.y_i.2^{-i} \\ y_{i+1} = y_i + s_i.x_i.2^{-i} \end{cases} \tag{8}$$

## III. SVD Computation Algorithm

Algorithm for calculate SVD base on 2 steps: Bidiagonalization and Diagonalization. Therein, Bidiagonlization converts the original matrix to 2-diagonal form, and Diagonalization makes the 2-diagonal form to single diagonal form. Elements on the diagonal is Singular Value. Given Rotation is used in this 2 steps

### A. Bidiagonlization

The original matrix $A \in R^{m\times n}$ is decomposed to 3 matrices:

$$A = LBR \tag{9}$$

where

$B \in R^{m\times n}$ is bidiagonal matrix
$L \in R^{m\times m}$ and $R \in R^{n\times n}$ are orthogonal matrices
L is column-rotate matrix and R is row-rotate matrix

This algorithm is expressed by following diagram 1

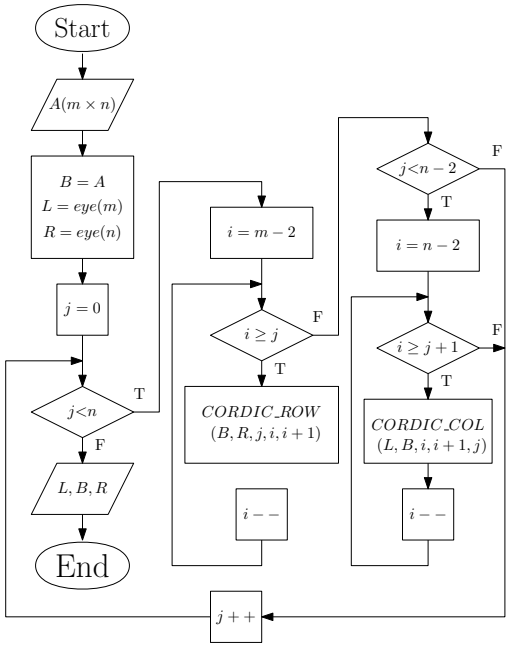


Fig. 1. Bidiagonalization algorithm

Where:

$eye(m)$ and $eye(n)$ are unitary matrices with size $m \times m$ and $n \times n$
Input is $mxn$ matrix A ($m > n$, for suite with ultra wide band receiver)
Outputs are bidiagonal matrix B($m \times n$), orthogonal matrix $L$, $R$, they satisfy $A = LBR$

At the $j-th$ loop, this algorithm destructs elements on $j-th$ column and $j-th$ row. The elements are processed with order: top up (column) and right to left (row).

*1) Column rotation:* At $j-th$ loop, the column rotation is processed with 2 consecutive rows with reference for rotating at $j-th$ column. The maximum local loop inside one column rotation is $(m-2)$. The next turn this number is subtracted one.

*2) Row rotation:* At $j-th$ loop, the column rotation is processed with 2 consecutive rows with reference for rotating at $j-th$ column. The maximum local loop inside one column rotation is $(m-2)$. The next turn this number is subtracted one.

### B. Diagonalization

Process which convert the matrix from Bidiagonal form to Diagonal form is realized by Given rotation. The Given Rotation is apply on the diagonal of input matrix. This process is repeated many time until the output reaches expected requirement.

$$B = P\Sigma Q \tag{10}$$

Where:

$\Sigma \in R^{m\times n}$ is diagonal matrix
$P \in R^{m\times n}$ and $Q \in R^{n\times n}$ are orthogonal matrices

This algorithm is expressed by following diagrams 2

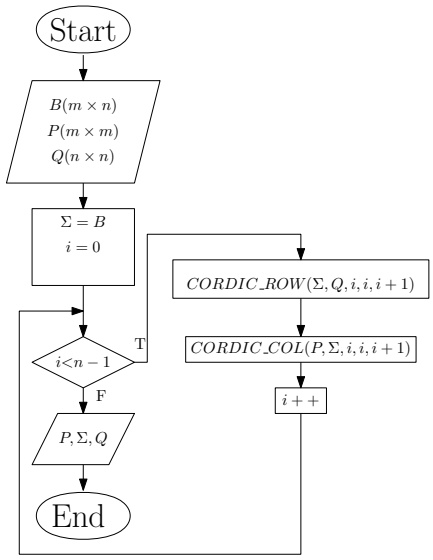


Fig. 2. Diagonalization algorithm

Where

$eye(m)$ is unitary matrix with size $m \times m$
Input is bidiagonal $m \times n$ matrix
Outputs are diagonal $m \times n$ matrix $\Sigma$, 2 orthogonal matrix $P$,$Q$, they satisfy $B = P\Sigma Q$

At the $i-th$ loop, the algorithm destruct elements on the right and below of $B(i,i)$ by process row rotating one time and column rotating one time. But this element is only approaches to zero but not equal to zero. So that, this process is repeated many time to get good result as expected. After 16 times shown in Fig.3, value around diagonal approaches zero, and value on diagonal approaches Singular Value.

## IV. SVD Hardware Implementation

From the described in previous sections, it is clear that the complexity of SVD computation rests on the matrix rows management and the computation of functions sine and cosine. Fig.4 shows a block diagram with the main

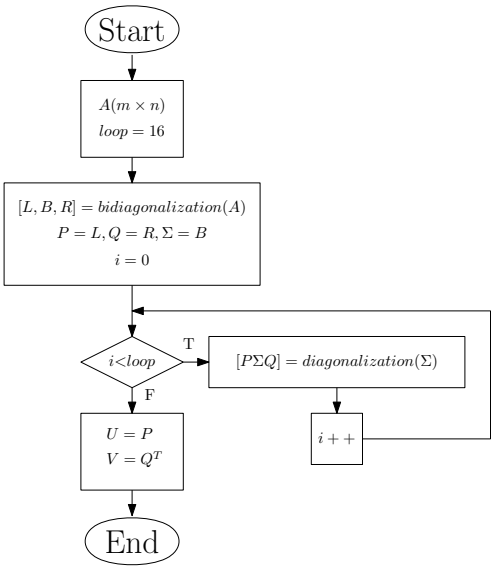


Fig. 3. SVD algorithm

components for the hardware implementation of the SVD computation algorithm in Fig.1,Fig.2, and Fig.3. A Control Unit provides synchronization and control signals to carry out all the different tasks on each component of Datapath. Therein, Control block puts out signals such as $CE0\_A, CE1\_A, SEL\_ROT, CE0\_UV, CE1\_UV$ in order to control datapath block.

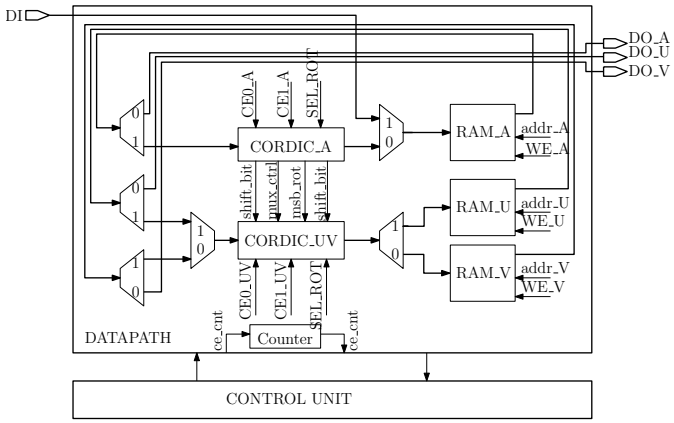


Fig. 4. Block diagram for the hardware implementation of the SVD computation algorithm.

### A. RAM

In order to manage the information stored in two different rows of the analyzed matrix $A$, it is necessary utilize a RAM with the following characteristics:

- 8 24-bit input data ports and 8 24-bit output data ports
- 6-bit address(1-bit: row/column, 3-bit: row address, 2-bit: column address)
- control signals for enabling data transferences to the memory during writing operations (WE)

Fig.(5) shows a block diagram of this one-port RAM, which main feature is its capability of reading and writing in two directions(row and column).

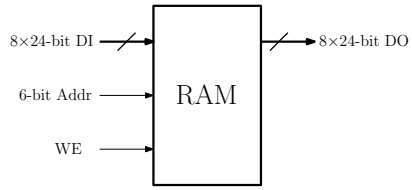


Fig. 5. RAM for rows and columns management.

### B. CORDIC unit

The internal structure of the CORDIC unit is showned in Fig.6 Control block is separated to use again for $N$ vectors $(x, y)$ (N CORDIC units). Value of Counter defines number of shift bit in each loop. Most significant bit (MSB) of $y_k$ control ADD/SUB blocks of $N$ CORDIC units. Multiplier $K$ $(MULK)$ multiples input with $K$. Because $K$ is defined as a constant, it is implemented by SHIFT and ADDER. Beside, ROTATION block makes the acurance increased.

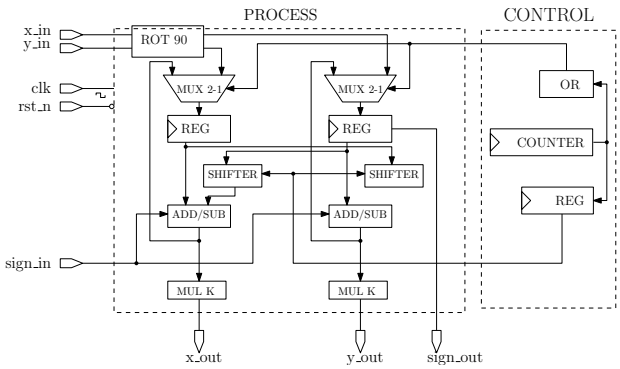


Fig. 6. CORDIC for two elements

### C. 2x4 CORDIC

Inputs of 2x4 CORDIC block are 4 vectors(x_i,y_i). Fig.7 shows block diagram of 2x4 CORDIC block. This block is implemented from four CORDIC unit. Control block in 2x4 CORDIC generates i/MUX ctrl to control MUX blocks and ordinal loop number in CORDIC unit for each loop. MUX block selects one of MSB signals to control ADD/SUB block in CORDIC unit depend on index. Index defines vector $(x\_k, y\_k)$ which is realized CORDIC. This vector also defines rotation angle of each loop.

### D. 2x8 CORDIC

Instead of having a 2x4 CORDIC and a 2x8 CORDIC, we implemented 2x8 CORDIC from two 2x4 CORDIC blocks in order to save resources. Fig.8 shows block diagram of 2x8 CORDIC.

## V. Results

This section shows the experimentation carried out to test the performance of the proposed hardware implementation for SVD computation of an $8 \times 4$ matrix $A$, the resource utilization on an FPGA device, as well as the obtained results.

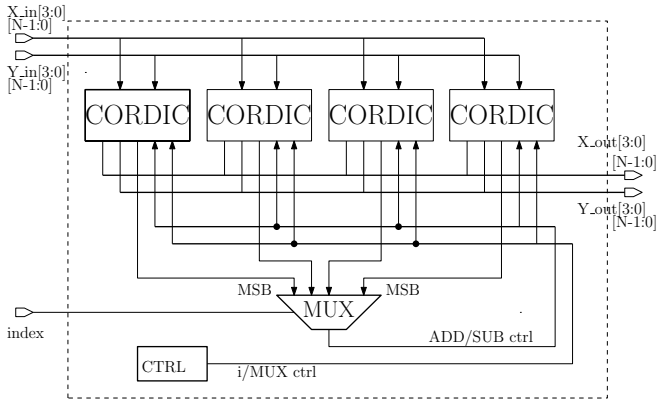


Fig. 7. CORDIC system diagram

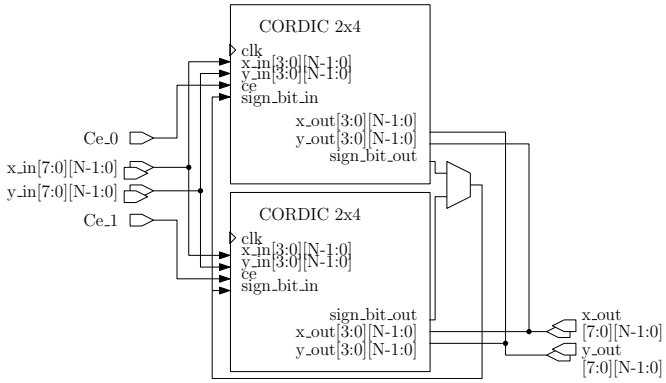


Fig. 8. Block diagram of 2x8 CORDIC

### A. Hardware Implementation Results

The proposed SVD hardware computation unit was implemented in a low-cost FPGA device from Xilinx $XC6SLX45$. Table I summarizes the hardware implementation results for $8 \times 4$ matrix of the proposed SVD hardware computation unit as a percentage of the available resources, as well as the maximum operation frequency.

| Resource Utilization | Xilinx $XC6SLX45$ |
|---|---|
| Number of slice register | 8% |
| Number of slice LUTs | 70% |
| Multipliers | 0 |
| Max. Op. Freq. | 47.690MHz |

TABLE I
RESOURCE UTILIZATION OF THE PROPOSED FPGA-BASED SVD COMPUTATION UNIT

### B. SVD Computation Results

Table II shows the corresponding singular values with the minimum and maximum estimation errors for the case of a $8 \times 4$ matrix. This table also shows the elapsed time for the software and hardware implementations

| Singular Value | Matlab | FPGA | % error |
|---|---|---|---|
| $\sigma\_1$ | 2555696 | 2555356 | 0.013 |
| $\sigma\_2$ | 1940169 | 1939863 | 0.016 |
| $\sigma\_3$ | 1210372 | 1210014 | 0.029 |
| $\sigma\_4$ | 960308 | 960143 | 0.017 |

TABLE II
SVD COMPUTATION OF A $8 \times 4$ MATRIX

## VI. Conclusion

Singular value decomposition is an important signal processing technique in many applications. So it was implemented in many different ways. Unfortunately, this method is limited by the complexity and resource utilization of SVD computation. In this paper, the parallel architecture of SVD computation is presented. It has been shown that the method, parallel CORDIC computation, parallel read and write in one row or column to reduce a lot of computing and storing cycles. It also saves hardware resource: number of multiplier (using CORDIC replace multiplier), number of CORDIC blocks (sequence computing).....

## Acknowledgment

There are many people without whose support this project would not have been possible. The most important person who we would like to thank is our instructor: Dr. Ngo Vu Duc. He helped us in ways beyond what his titles suggest, both through sharing his wisdom and experience of running this project before, and through his last-minute efforts to solve emergent problems that could or could not have been foreseen by us.

## References

[1] Kuan-Ju Huang, Jui-Chung Chang, Chih-Wei Feng, and Wai-Chi Fang, *A Parallel VLSI Architecture of Singular Value Decomposition Processor for Real-time Multi-channel EEG System*. 2013 IEEE 17th International Symposium on Consumer Electronics (ISCE)

[2] Luis M. Ledesma-Carrillo, Eduardo Cabal-Yepez, Rene de J. Romero-Troncoso, Arturo Garcia-Perez, Roque A. Osornio-Rios, Tobia D. Carozzi. *Reconfigurable FPGA-Based Unit for Singular Value Decomposition of Large $m \times n$ Matrices*. 2011 International Conference on Reconfigurable Computing and FPGAs

[3] A. Ahmedsaid, A. Amira and A.Bouridane. *Improved SVD systolic array and implementation on FPGA.*

[4] Yue Wang, *Singular Value Decomposition Based Pipeline Architecture for MIMO Communication Systems*, June 2010.

[5] Todd K. Moon and Wynn C. Stirling, *Mathematical Methods and Algorithms for Signal Processing.* Prentice Hall, 1999.