AN FPGA-BASED SINGULAR VALUE DECOMPOSITION PROCESSOR

Dissertation submitted in partial fulfillment of the requirement for the degree of

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

By

Rahul Attri (141069)

Shiven (141072)

UNDER THE GUIDANCE OF

Dr. Harsh Sohal

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY,WAKNAGHAT

CERTIFICATE

We hereby declare that the work reported in the B-Tech report entitled "**AN FPGA-BASED SINGULAR VALUE DECOMPOSITION PROCESSOR**" submitted at **Jaypee University of Information Technology, Waknaghat, India,** is an authentic record of the work carried out by Rahul Attri (141069) and Shiven (141072) under the supervision of **Dr. Harsh Sohal.** We have not submitted this work else where for any other degree or diploma**.**

Rahul Attri (141069)

Shiven (141072)

Department of Electronics And Communication

Supervisor : Dr. Harsh Sohal

Grade : Assistant Professor

Signature :

ACKNOWLEDGEMENT

It is our privilege to express our sincerest regards to our project supervisor **Dr. Harsh Sohal**, for their valuable inputs, able guidance, encouragement, whole-hearted cooperation and direction throughout the duration of our project. We deeply express our sincere thanks to our supervisor Dr. Harsh Sohal for encouraging and allowing us to present the project on the topic "AN FPGA BASED SINGULAR VALUE DECOMPOSITION PROCESSOR" at our department premises for the partial fulfillment of the requirements leading to the award of B-Tech degree.

Toward the end, we might want to express our true regards on account of every one of my companions and other people who helped me straight forwardly or in a roundabout way amid this venture work.

ABSTRACT

This features a FPGA based Singular Value Decomposition processor which utilizes the two-sided turn Jacobi SVD calculation. Two SVD processors, the Basic SVD Processor and the Extended SVD Processors. In the Basic SVD Processor, the greatest network, which can be obliged in the focused on gadget, is investigated by exploiting of the highlights of the gadget, and a few outline methods are utilized to speed the SVD calculation. The objective of the Extended SVD Processor is to register a major SVD without expanding the processor estimate by reusing the SVD exhibit of the Basic SVD Processor. Both of the SVD processors can successfully compute the SVD, and the errors from the Matlab algorithm was used to measure the quality of the image using SVD algorithm.

TABLE OF CONTENTS

List of Figures

LIST OF SYMBOLS

a_{pq}	the cell value
J(p,q,ϕ)	right sided Jacobi rotation matrix
J(p,q, Θ)	left sided Jacobi rotation matrix
J(p,q,ϕ,j)	right sided Jacobi rotation matrix for normalized SVD algorithm
SJ(p,q,Θ, j)	left sided Jacobi rotation matrix for normalized SVD algorithm
Z	the CORDIC scale factor
U	n x n orthogonal matrix
Z	n x n nonnegative diagonal matrix
A	original matrix
Φ	right rotation angle
Θ	left rotation angl

CHAPTER 1

INTRODUCTION

1.1 SVD

The Singular Value Decomposition (SVD) is a standout amongst the most essential calculations in numerical polynomial math for furnishing quantitative data about a grid with various applications in flag and picture preparing. This work was roused by the plan of an ongoing computerized discourse upgrade conspire which requires quick Singular Value Decomposition (SVD) calculation. A work associated exhibit is proposed to be utilized with the CORDIC (Coordinate Rotation Digital Computer) calculation to register the SVD. The array structure allows all the processing elements to compute in parallel thus increasing the computation speed. The image below can be represented by a matrix of grey-scale values 0-255. The picture of the mathematician Gauss is a 340 x 280 pixel image with 256 shades of grey. When the singular value decomposition of the image is calculated, three resultant matrices are produced one of which is the matrix containing the singular values on the diagonal and all other values are zero. The diagonal singular values are listed in order of greatest to least. The image where k = 2 represents the image as reproduced from 2 singular values. It can be seen that when k = 32, there is almost no difference between the original image and the one reproduced from only 32 singular values.

The applications of SVD include signal processing ,image deblurring, and digital image processing. This example provided a little context what is to be achieved and how it can be applied .The Singular Value Decomposition (SVD) of a matrix is a computationally complex linear algebra algorithm, and its applications vary from control systems, to digital speech processing and digital image processing.

Real-time applications, such as digital signal processing and image processing requires fast SVD computation involving large matrices. The high through put needed in the SVD computation can be obtained only through special architectures. A mesh-connected array is proposed to be used along with CORDIC (Coordinate Rotation Digital Computer) algorithms for the computation of SVD. All the processing elements can compute in parallel because of the improved array structure. The structure speeds the implementation of the SVD processor. A two-sided revolution Jacobi SVD calculation is utilized to process the SVD and is actualized on a two million entryway FPGA. A work associated cluster structure is proposed to process the SVD of a framework, in order to abbreviate the emphasis time and in this manner increment the execution speed. The array consists of an n/2xn/2 array of 2x2 processor elements to compute the SVD of an nxn matrix. The trigonometric functions and the vector multiplication in the algorithm uses CORDIC (Coordinate Rotation Digital Computer) algorithms for hardware-efficient solutions. Firstly, two SVD processors, the Basic SVD Processor and the Extended SVD Processor were developed. The algorithms to decompose the matrix were first evaluated in Matlab and then the processors were implemented using the Virtex-II FPGA from Xilinx as the target device for the decomposition. A two-sided revolution Jacobi SVD calculation is utilized to process the SVD and is actualized on a two million entryway FPGA. A work associated cluster structure is proposed to process the SVD of a framework, in order to abbreviate the emphasis time and in this manner increment the execution speed. These two processors were successfully implemented on the FPGA device.

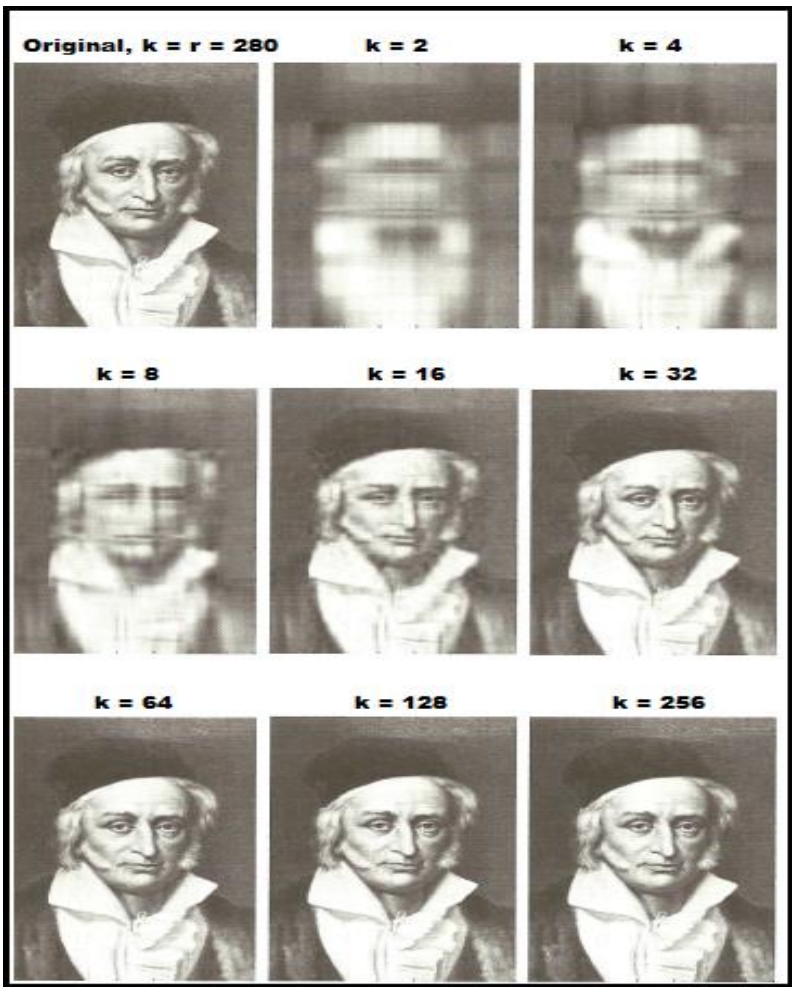


Fig 1.1

A matrix is simple a rectangular array of number considered as an entity. it is usually enclosed in either parentheses or brackets. with M rows and N columns in the array,(written as m×n).and usually denote a matrix in capital letters such as A,B,and so on.however,where a matrix has only one row, it will often we regarded as row vector and denoted by Bold lowercase letter, similarly treat as one column but in general m×n matrix is of the form

$$A=\begin{bmatrix} a11 & \cdots & a12 \ldots . a1n \\ a21 & \cdots & a22 \ldots . a2n \end{bmatrix}$$

This matrix is said to have order M×N the mn numbers that constitute called elements or entities. in particular, a_{ij} denotes the elements in the i^{th} row and j column

Types of matrix:

1. Square matrix

2. Diagonal matrix

3. Scalar matrix

4. Identity matrix

5. Null matrix

6. Transpose of matrix

Matrix operations:

We know that matrices useful for storing information. the real motivation for introducing matrices, however , is that there are useful rules for manipulating them that correspond with familiar rule of ordinary algebra.

Thus two matrix A and B are equal if they have the same dimensions and if all their corresponding entries are equal.

Application of matrix:

It is used for simple calculation tool, can be represented in a simple form and complex form. They are used for plotting graphs, statics and also to do scientific studies in almost different fields. it is used in representing the real world data like traits of people's population, habits etc.

1.2 SVD Algorithm :

The SVD of a real n x n matrix A is its factorization into the product of three matrices,

$$A = U\Sigma V^T$$

where U and V are n x n unitary matrices and Σ = diag (σ1,σ2,...,σn) is an n x n diagonal matrix. The columns of U and V are called respectively the left and right singular vectors and σ_i the i^{th} singular value of A. The singular values in Σ may be arranged in any order but they are usually arranged in decreasing order.

CHAPTER 2

Literature Review

The importance of the SVD as a matrix factorization technique is underscored by the variety of algorithms available. They range from serial algorithms to parallel Jacobi methods. On a conventional uniprocessor system, the most commonly used procedure is the Golub-Kahan-Reinsch algorithm. This SVD algorithm is a kind of serial algorithm that is implemented in both EISPACK and LINPACK . The first step in the Golub-Kahan-Reinsch algorithm is the bi-diagonalization of the matrix. This is followed by an iterative diagonalization of the bi-diagonal matrix to complete the SVD. For an m X n matrix, the time complexity o f the Golub-Kahan-Reinsch algorithm is of the order 0.

The QR-based SVD algorithm and the Jacobi SVD algorithm are the most commonly used classes of algorithms among many numerically stable algorithms for computing the SVD. In sequential computing the QR-based algorithms are usually preferred because they are faster than the Jacobi-based algorithms. However, the inherent parallelism which characterises the Jacobi method has made them an attractive solution on parallel computers. Moreover, James Demmel demonstrated that the Jacobi algorithms may be more accurate than the QR-based SVD algorithm in

The Jacobi SVD algorithms have recently attracted a lot o f attention as they have a high degree o f potential parallelism different Jacobi-based methods for SVD computation are usually considered, namely the Kogbetliznt (two-sided rotation) method and the Hestenes (one-sided rotation) method . Although the two-sided rotation method is computationally more expensive than the one-sided method, the two-sided algorithm usually is adopted because it is suitable for mapping onto a regular systolic array architecture and highly suitable for parallel computation.

Brent, Luk and Van Loan proposed an expandable systolic array of simple processors to compute the SVD. A ‘parallel ordering’ discovered by Brent and Luk is adopted in this structure instead o f the conventional cyclic-by-row ordering or the cyclic-by-column ordering . The Brent, Luk, and Van Loan (BLV) array combined the Jacobi SVD method with this ‘parallel ordering’ scheme to exploit the parallelism inherent in the SVD computation.

Algorithms for use on uniprocessor systems require many division and square root calculations to compute the sine and cosine rotation parameters necessary in Jacobi algorithms. The Coordinate Rotation Digital Computer (CORDIC) algorithm can easily compute the trigonometric functions and the vector multiplication by using a set of shift-add algorithms. All have invoked the CORDIC techniques in the BLV systolic array and performed an SVD processor in hardware .

The appearance of reconfigurable rationale PCs grants higher rates of devoted equipment arrangements at costs that are focused with the customary programming approach. A broadly useful processor based SVD processor was acknowledged by Manfredi, C. Be that as it may, because of the speed confinement of the broadly useful processor, just a low-arrange SVD is performed for their situation under the continuous usage. The Jacobi SVD strategy in light of the CORDIC calculation and associated by utilizing the parallel cluster engineering planned by is an equipment proficient arrangement and can be acknowledged in VLSI innovation .FPGAs present a powerful tool for SVD algorithm hardware realization, due to their reconfigurable features and shorter design cycle. Based on the BLV array in present an architecture which can increase the efficiency of the BLV array structure. Bobda and Steenbock designed an SVD processor targeting FPGAs by utilizing their reconfigurability aspect, and thus provided an improved device for the hardware solution o f the SVD.

2.1The Basic SVD Processor

The Basic SVD (BSVD) Processor is based on the BLV square mesh-connected array using a PE for each 2 x 2 sub matrix, interchanging matrix elements and rotation angles and parallel ordering. The BSVD Processor algorithm transfers the rotation data to all off-diagonal PEs at the same time rather than propagating them through the PE array. The BSVD Processor architecture consists of two function blocks, the BSVD PE array (the diagonal PE sand the off-diagonal PEs) and the array controller.

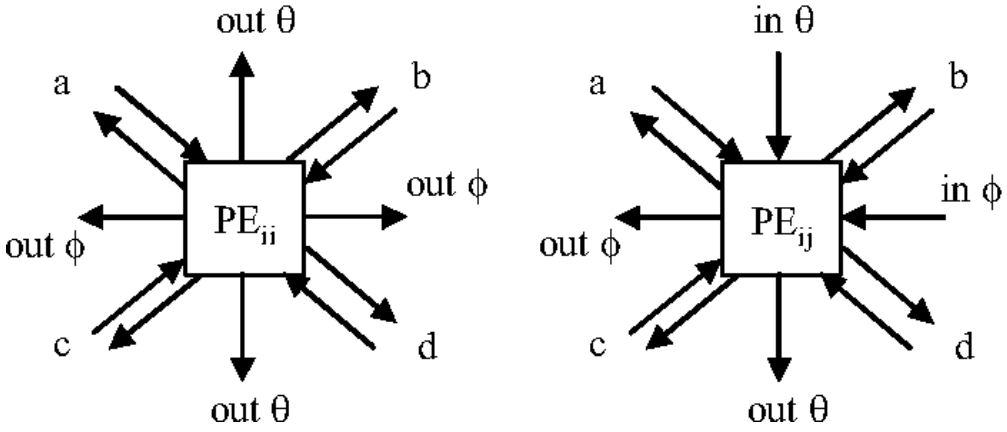


Fig 2.1

In the BSVD array, the 2x2 SVD algorithm for the sub matrices is split into two portions, one to compute the left and right rotation angles called the Angle Solve, and the other to perform the two-sided Jacobi rotation equation called Vector Rotation. To compute a 4 x 4 BSVD requires 2 diagonal PE sand 2 off-diagonal PEs. The diagonal PEs compute the Angle Solve function and transfer the new rotation parameters to all PEs simultaneously. Then all PEs concurrently execute the vector rotation and exchange the matrix elements with their neigh bours. This reduces the number of steps required to accomplish one iteration from n/2steps required by BLV to 2 steps required by the BSVD array. The data exchange is implemented by the array controller and takes less time than one iteration of computations.

The array controller has three tasks.

- It controls the data in and out of the array
- It schedules the propagation of the rotation angles
- Using the BLV parallel ordering, it exchanges all the matrix elements after all PEs have completed the Vector Rotation function.

The array controller includes one module for each of the sub matrices. So for a 4 x 4 BSVD processor, four modules are required. Each module uses the same architecture and state machine algorithms. Within each module there are four blocks as shown. Each iteration, the 4input matrix elements accepted by each sub matrix are stored. temporarily in the array controller module in serial memory. Parallel to serial conversion is required to convert the data toserial form. Since data is also output from each sub matrix it needs to be converted back from serial to parallel. The state machine block implements algorithms to control the movement and rearrangement of the data so that pairs of columns can be orthogonalized in parallel.

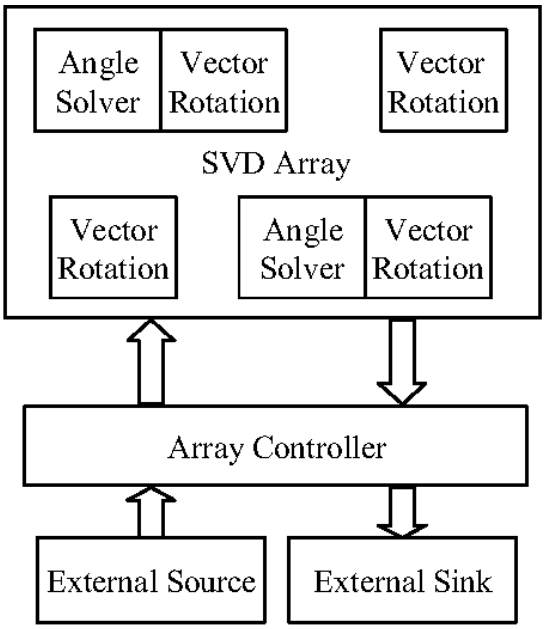
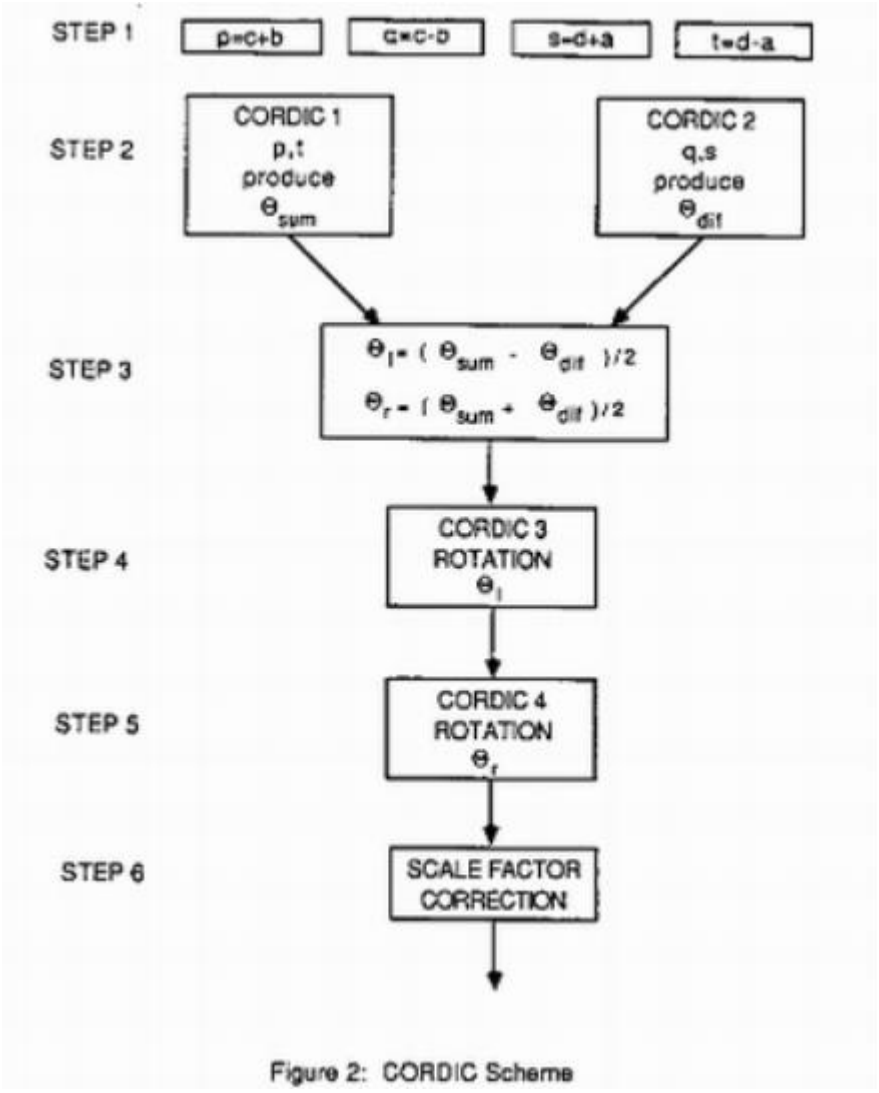


Fig 2.2



c Figure 2: CORDIC Scheme

fig 2.3

2.2 FPGAs vs. ASICs and General-Purpose Processors

FPGAs (Field Programmmable Gate Arrays), which have become one o f the most effective hardware implementation devices for digital circuit design, have been more and more adopted in industry. FPGAs have a fixed but electrically programmable architecture which consists of a two-dimensional structure of logic blocks. Thanks to their reconfigurable capability, FPGAs

can be used to implement different circuits simply by appropriate programming.

In contrast to other implementation technologies, such as ASICs (Application Specific Integrated Circuits), FPGAs do not need a new silicon chip fabricated for each design, and the hardware-efficient algorithms which run in FPGAs speed the implementation, compared with the software-efficient algorithms used in general-purpose processors. Furthermore, a design can be easily verified on a prototype development system in the lab. Therefore the whole development o f applications using FPGAs requires less time than using ASICs, and the revision of the FPGAs design costs less as well.

2.3 Comparisons between the BLV Array and the Proposed Array

Table summarizes the performance comparisons between the proposed array and the BLV Array, n is the size of the SVD matrix, and T_{as} and T_{vr} are the time for Angle Solver and the time for Vector Rotation, respectively. Ti and T2 are the broadcasting time in the BLV array and the proposed mesh-connected array respectively. They consist of the time of rotation angle propagation and cell value interchange. Since the broadcasting time is shorter than the time of Angle Solve and Vector Rotation, it can be ignored in the performance time per iteration. It should be mentioned that the 'T_{as}'s and the 'T_{vr} 's in both the BLV array and the proposed array are equal

.

SVD Array	BLV Array	Proposed Array
'Time per iteration	T_{AS}+(n/2)*T_{VR}+T,	T_{as} +T_{vr} +T_2

Table 2.1

2.4APPLICATIONS

2.4.1 Application 1 - image compression

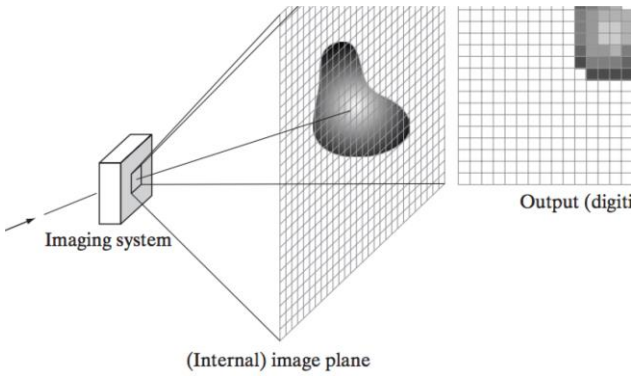


Fig 2.4

Grayscale image = Matrix, each entry represents a pixel brightness.

2.4.2Application 2 -Image deblurring

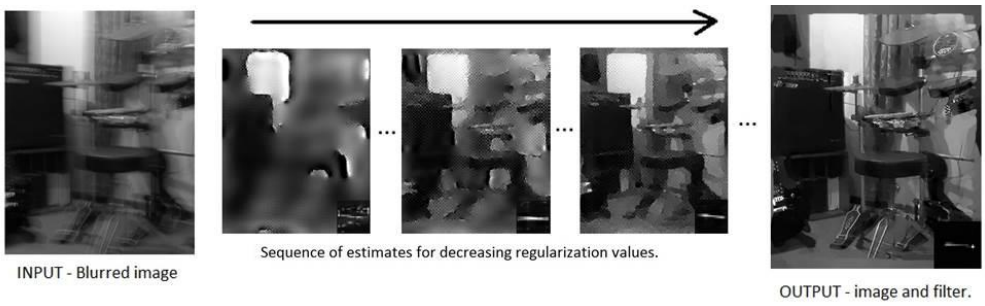


Fig 2.5

Image processing is method to perform some operation on an image, in order to get an enhanced image .it is a type of signal processing in which input is an image and output may be image or charecteristics associated with that image . These days image processing is becoming a more growing technology in word. . It is widely used in engineering and computer4 science discipline to.

Image processing uses three steps which are following

1 Canning the image via image various tools like, acquisition

2 Second analysis of image and manipulating the images

3 The output of image i.e report of the image in which add analysis of images.

Two types methods for image processing techniques are uses

1 Analog

2 Digital Image processing

Which here we are discussing

1. Analogue image processing can be used for the Hard copies like printouts and photographs image analysis used various fundamental of interpretion while using these visual technique

2. Computerized picture handling strategy controls of the advanced picture by utilizing PC the three general stage that a wide range of information need to experience while utilizing advanced method are pre-preparing improvement and show data extraction.

Resizing image

Image interpolation occur when you resize or distort your image from one pixel grid to another .it is necessary when you need to increase or decrease the total number of pixel

Interpolation works by using known data to estimate values at unknown points image interpolation works in two direction and tries to achieve a best approximation of pixel intensity based on the values of surrounding pixel

Many compact digital camera can perform both an optical and digital zoom a camera performs an optical zoom by moving the zoom lenses so that increase

exponentially. even though the photo with digital zoom contains the same numbers of pixel,

2.4.3 OTHER APPLICATION

• computer tomography (CT)

• magnetic resonance

• seismology

• crystallography

CHAPTER 3

SYSTEM DEVELOPMENT

3.1 Software and Hardware

FPGA AND XILLINX

A field-programmable gate array is an integrated circuit designed to be configured by a customer or a designer after manufacturing – hence field-programmable. The FPGA configuration is generally specified using a hardware description language(HDL), similar to that used for an application-specific integrated circuit (ASIC).

Fig 3.1

FPGAs contain a variety of programmable rationale pieces, and a chain of importance of reconfigurable interconnects that enable the squares to be wired together, in the same way as other rationale entryways that can be between wired in various designs. Rationale pieces can be arranged to perform complex combinational capacities or only basic rationale entryways like and XOR. In many FPGAs, rationale pieces likewise incorporate memory components, which might be basic flip-failures or more entire squares of memory Contemporary field-programmable door clusters (FPGAs) have vast assets of rationale entryways and RAM squares to actualize complex computerized calculations. As FPGA plans utilize quick I/Os and bidirectional information transports, it turns into a test to confirm revise timing of legitimate information inside setup time and hold time. Floor arranging empowers asset allotment inside FPGAs to meet these time

limitations. FPGAs can be utilized to execute any sensible capacity that an ASIC could perform. The capacity to refresh the usefulness in the wake of delivery, fractional re-arrangement of a segment of the plan and the low non-repeating building costs in respect to an ASIC outline (despite the for the most part higher unit cost), offer favorable circumstances for some applications.

Some FPGAs have simple highlights notwithstanding computerized capacities. The most widely recognized simple element is programmable slew rate on each yield stick, enabling the designer to set low rates on gently stacked pins that would somehow or another ring or couple unsatisfactorily, and to set higher rates on vigorously stacked sticks on fast channels that would some way or another run too gradually. Additionally basic are quartz-gem oscillators, on-chip protection capacitance oscillators, and stage bolted circles with installed voltage-controlled oscillators utilized for clock age and administration and for fast serializer-deserializer (SERDES) transmit timekeepers and recipient clock recuperation. Genuinely normal are differential comparators on input pins intended to be associated with differential flagging channels. A couple of blended flag FPGAs have incorporated fringe simple to-advanced converters (ADCs)and computerized to-simple converters (DACs) with simple flag molding squares enabling them to work as a framework on-a-chip. Such gadgets obscure the line between a FPGA, which conveys advanced zeros on its inner programmable interconnect surface, and field-programmable basic bunch (FPAA), which passes on straightforward characteristics on its internal programmable interconnect surface. Truly, FPGAs have been slower, less vitality proficient and for the most part accomplished less usefulness than their settled ASIC partners. A more established investigation had demonstrated that plans actualized on FPGAs require overall 40 fold the amount of region, draw 12

fold the amount of dynamic power, and keep running at 33% the speed of relating ASIC implementations[citation needed]. All the more as of late, FPGAs, for example, the Xilinx Virtex-7 or the Altera Stratix 5 have come to match comparing ASIC and ASSP arrangements by giving fundamentally lessened power use, expanded speed, bring down materials cost, negligible usage land, and expanded potential outcomes for re-design 'on-the-fly'. Where beforehand an outline may have included 6 to 10 ASICs, a similar plan would now be able to be accomplished utilizing just a single FPGA.

Favorable circumstances of FPGAs incorporate the capacity to re-program in the field to settle bugs, and may incorporate a shorter time to market and lower non-repeating designing expenses. Sellers can likewise take a center street by building up their equipment on normal FPGAs, yet produce their last form as an ASIC with the goal that it can never again be adjusted after the outline has been conferred.

Xilinx claims that few market and innovation progression are changing the ASIC/FPGA worldview:

• Integrated circuit advancement costs are rising forcefully

• ASIC unpredictability has protracted advancement time

• R&D assets and headcount are diminishing• Revenue losses for slow time-to-market are increasing

•Financial constraints in a poor economy are driving low-cost technologies

These patterns improve FPGAs an option than ASICs for a bigger number of higher-volume applications than they have been truly utilized for, to which the organization characteristics the developing number of FPGA configuration begins

Some FPGAs have the capacity of halfway re-design that gives one part of the gadget a chance to be re-modified while different segments keep running.

MATLAB CODE

SVD Function

```
function [U, E, V] = svd_funct(A)
% expect 2x2 matrix as input
% outputs 3 2x2 matrixes, following the A = UED singular decomposition

a = A(1,1);
b = A(1,2);
c = A(2,1);
d = A(2,2);

sum = atan((c+b)/(d-a)); %sum = angle([p+t*i]) %theta sum (tr + tl)
diff = atan((c-b)/(d+a)); %diff = angle([q+s*i]) %theta diff (tr - tl)

l = (sum - diff)/2; %radians to rotate left
r = (sum + diff)/2; %radians to rotate right

L = [cos(l), sin(l); -sin(l), cos(l)]; % left rotation matrix
R = [cos(r), sin(r); -sin(r), cos(r)]; % right rotation matrix

U = L;
E = L.'*A*R;
V = R.';
% A = U*E*V

SVD demo
A = [1,2;3,4]

a = A(1,1)
b = A(1,2)
```

```
c = A(2,1)
d = A(2,2)

p = c + b
q = c - b
s = d + a
t = d - a

sum = angle([p+t*i]) %theta sum
diff = angle([q+s*i]) %theta diff

l = (sum - diff)/2
r = (sum + diff)/2

L = [cos(l), sin(l); -sin(l), cos(l)]
R = [cos(r), sin(r); -sin(r), cos(r)]
%cos sin -sin cos
taoeu = sin(l)
aoeudhctns = sin(pi/2)

[U,E,V] = svd(A)

ED = L*A*R.'
```

>>**SVD_demo**

A = 1 2

3 4

a = 1

b = 2

c = 3

d = 4

p = 5

q = 1

s = 5

t = 3

diff = 1.3734

l = -0.4165

r = 0.9569

L = 0.9145 -0.4046

0.4046 0.9145

R = 0.5760 0.8174

-0.8174 0.5760

taoeu =-0.4046

aoeudhctns = 1

U = -0.4046 -0.9145

-0.9145 0.4046

E = 5.4650 0

0 0.3660

V = -0.5760 0.8174

-0.8174 -0.5760

ED = 0.0000 0.3660

5.4650 -0.0000

CHAPTER 4

IMPLEMENTION AND METHODOLODGY

4.1 Hardware Implementation Solutions of SVD Algorithm

The two-sided rotation Jacobi SVD algorithm was chosen for implementation. The 2 X 2 real Jacobi SVD algorithm is introduced in applied to the computation of an n x n real SVD in this chapter. The hardware implementation of the Jacobi SVD algorithm needs a hardware efficient algorithm to compute the trigonometric functions, such as the inverse tangent function used to compute the left and right rotation angles, and a special mechanism that realizes the parallel computation of the Jacobi SVD algorithm in the hardware device to speed the implementation. The Coordinate Rotation Digital Computer (CORDIC) algorithm which uses only shifts and adders for the trigonometric functions and vector multiplication is described. The Brent-Luk-Van Loan (BLV) mesh-connected array architecture n 2 is used to explain how to connect (—) 2 x 2 submatrices for computing the SVD of an n x n real matrix.

4.2 CORDIC Algorithm

The Digital Signal Processing (DSP) landscape has long been dominated by microprocessors due to their capabilities, such as single cycle multiply-accumulate instructions, and their low cost and extreme flexibility. However, software approaches are not fast enough for the demands o f real-time DSP, and the advent of reconfigurable logic devices permits the higher speeds o f dedicated hardware solutions. Unfortunately, the algorithms optimized for those software approaches do not map well into the hardware, so hardware-efficient solutions have been developed to realize the DSP algorithms in hardware. Among these equipment productive arrangements is an arrangement of move include calculations known as CORDIC for figuring a wide range o f trigonometric and different capacities . The

CORDIC calculation is especially suited to equipment usage since it doesn't require any increases. CORDIC spins around pivoting the period of a perplexing number, by duplicating it by a progression o f consistent qualities. In any case, the duplicates would all be able to be forces of 2, so in parallel math they should be possible utilizing just moves and includes; no real multiplier is required. The CORDIC calculation was at first created by Voider , and the main inspiration was to iteratively settle trigonometric conditions. Later, Walther extended the algorithm to solve a broader range o f equations, which include the hyperbolic and square root equations. The CORDIC algorithms implement five varieties of functions, Vector Rotation (Polar to Rectangular Translation), Vector Translation(Rectangular to Polar Translation), trigonometric functions, hyperbolic functions, and square root equations. The CORDIC algorithm has also been proposed for computing Discrete Fourier , Discrete Cosine , Discrete Hartley and Chirp-z Transforms , filtering , SVD , and solving linear systems .

4.3Implementing SVD on a 2 x 2 matrix

Brent, Luk, Van Loan proposed a mesh-connected network of processors for handling SVD on a larger scale. However, to achieve the SVD through a network of processors, the SVD of a single 2 x 2 matrix must be first developed. They proposed an iterative process, based on multiple processors concurrently providing an SVD for their corresponding 2 x 2 matrix. Forsythe and Henrici proposed the diagonalization of a n x n matrix by a sequence of 2 x 2 SVD's.

$$\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin 0 & \cos\theta \end{bmatrix}^{T} \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix} = \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix}$$

Fig 4.1

The math involved is fairly straightforward on the handy TI-83. However, how can this be implemented on an FPGA that is based on mathematical

operations through logic gates. Three trig functions needed to be developed for sin, cos, and tan.

Where Φ and Θ represent the rotation angles used in the algorithm.

$$\begin{cases} \phi + \theta = \tan^{-1} \dfrac{c+b}{d-a} \\ \phi - \theta = \tan^{-1} \dfrac{c-b}{d+a} \end{cases}$$

Fig 4.2

To further break down the problem being solved, the left and right rotation angles must be written in terms of the input values {(a, b), (c, d)} that compose the 2 x 2 matrix input

Left rotation angle Φ:

$$\Phi = \left(\frac{1}{2}\tan^{-1}\left(\frac{c+b}{d-a}\right)\right) + \left(\frac{1}{2}\tan^{-1}\left(\frac{c-b}{d+a}\right)\right)$$

Right rotation angle Θ:

$$\Theta = \left(-\frac{1}{2}\tan^{-1}\left(\frac{c-b}{d+a}\right)\right) + \left(-\frac{1}{2}\tan^{-1}\left(\frac{c+b}{d-a}\right)\right)$$

Fig 4.3

Verilog Library of Programmable Modules (LPM's) are extremely accurate but utilize a large percentage of the system resources. Which is to say that there are built in Libraries within the FPGA's development software that already facilitate trigonometric functions. However, it is noted that the single 2 x 2 matrix will take multiple trig functions to return a desired SVD.

Additionally, concurrency is the goal, which means that multiple calculations should be occurring simultaneously. A single ATAN function in the LPM took up 20% of the system's resources alone. Therefore, Coordinate Rotation Digital Computer (CORDIC) modules needed to be developed for this specific application. The CORDIC algorithm works generally by a series of successive approximations that iteratively bring you closer to the true value through vector rotations. The number of iterations and the resolution of the binary value being used determine the margin of error produced from the calculations. The image below shows the generic CORDIC algorithm.

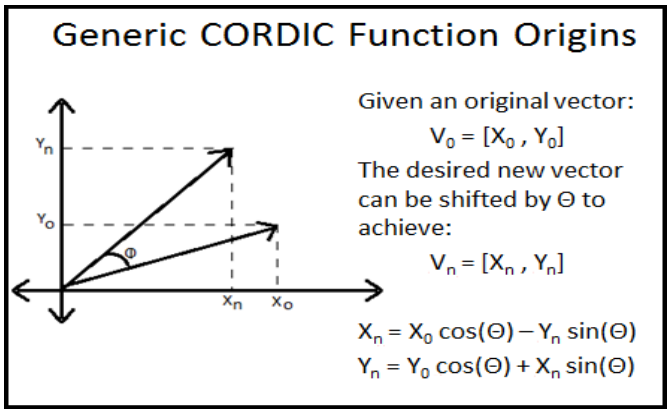


Fig 4.4

4.4Implementing SVD in a mesh-connected network of processors

With the functions in place to handle the SVD of a single 2 x 2 matrix, the next step is to begin integrating numerous instances of those processors and link them together. They are be linked in a mesh-connected network of processors. The general flow of information works iteratively again where the diagonal elements first calculate the left and right rotation angles and calculate the SVD locally. Those angles are propagated outward to

neighbouring processors. The SVD for those neighbouring elements are calculated while the rotation angles are propagated further outwards until all elements have completed one iteration of the SVD. The flow of signals is represented below for a 8 x 8 matrix of values.

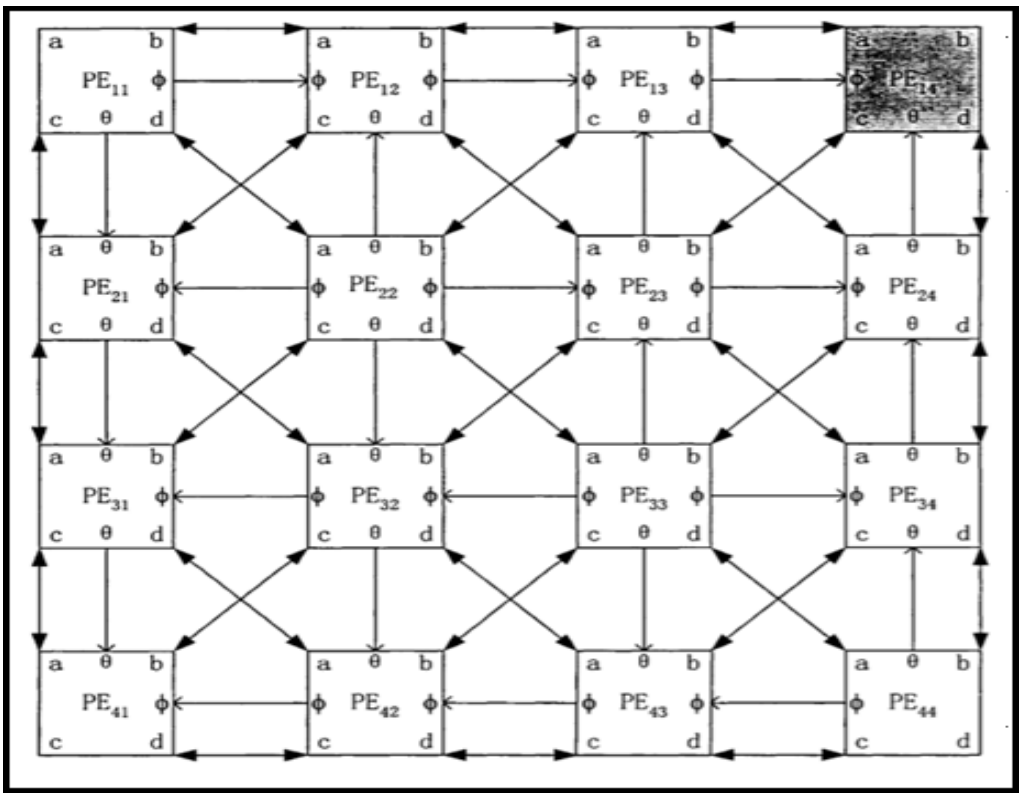


Fig 4.5

The next step is to trade the resultant data between neighbouring processors and repeat. The 16 submatrices in the array structure can be sorted into two groups based on their positions, which are the four diagonal processing elements and the twelve off-diagonal processing elements. These submatrices are interconnected by the input and output lines for transmitting the rotation angles and the cell values. Obviously, there are some differences between the input and output of these two submatrices. The diagonal processing elements propagate the two rotation angles Θ and ϕ to the other off-diagonal processing elements along the rows and the columns; in contrast, the off-diagonal processing elements just have access to the two parameters Θ and ϕ

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 CONCLUSION

The Singular Value Decomposition is an important matrix factorization method used extensively in engineering applications. It is particularly useful in the context of signal processing, image processing and robotics applications. Real-time data processing necessitates the use of hardware to sustain the computation speed required. The Jacobi-type methods, which form the basis for the systolic algorithms, are especially amenable to parallel processing. Two widely used SVD algorithms which are based on the Jacobi-type methods are the one-sided Jacobi rotation SVD algorithm (Hestenes Method) and the two-sided Jacobi rotation SVD algorithm (Kogbetliantz Method). The Kogbetliantz Method is embraced on the grounds that it is useful for mapping onto standard systolic exhibit design and is profoundly appropriate for parallel calculation. Brent, Luk and Van Loan propelled one sort of the systolic cluster with innate parallelism and synchronized multiprocessing capacities which are suited to address the difficulty of constant preparing. In the mean time, a parallel requesting proposed by Brent, Luk and Van Loan can be connected in the SVD calculation, too. It utilizes less emphasess to figure the SVD than the section cycle requesting and the column cycle requesting. As per the BLV exhibit calculation, all the preparing components function as corner to corner rushes of movement. The slanting preparing components can not begin work until the point that all the off-askew handling components complete the calculation. In this thesis, an n x n meshed-connected array based on the BLV array structure is proposed to

eliminate the idle time that the processing elements wait for the input data. Only two steps are required in the proposed array to finish one iteration computation instead of (n-1) steps in the BLV array. All processing elements can be defined as two groups of the blocks, Angle Solve blocks and Vector Rotation blocks. In the first step, Angle Solve blocks calculate the left and right rotation angles. Then Vector Rotation blocks rotate the original matrix with the left and right rotation angles, respectively. After O(n) sweeps, the original matrix is transformed into a diagonalized matrix. In order to realize the hardware implementation solution, the CORDIC arithmetic technique is used to figure out the inverse tangent functions and vector rotation functions. Two SVD processors, the Basic SVD Processor and the Extended SVD Processor, are developed in the thesis. Due to the area limitations of the device,Virtex-112000 from Xilinx, a 4 X 4 Basic SVD Processor or an 8X8 Extended SVD Processor is the maximum size processor which can be contained on the chip. The Basic SVD Processor takes advantage of the proposed mesh-connected array and some design techniques utilizing the features of the device to improve the performance.

5.2 FUTURE SCOPE

Now that the SVD algorithm has been successfully realized in the hardware implementation, some amelioration can be considered in the future. On the base of the Basic SVD Processor and the Extended SVD Processor technology, the SVD of a big matrix can be computed by using multi-FPGA devices. The analogous research where the multi-FPGAs perform the matrix algorithm has been published . The Spartan device can be considered to take the place of Virtex-II, because Spartan-3 also has DCM, block Select RAM and embedded multipliers, and it is a low cost FPGA. Due to the area limit of the device, the left and right rotation matrixes have not been computed in the design. It only needs the vector rotation blocks. There are two solutions to achieve the left and right rotation matrixes. One solution is to use more Vector Rotation blocks in the design, and the singular vector and the left and right rotation matrixes can be computed in parallel. Another one is

to reuse Vector Rotation blocks, and the idle time that Angle Solve blocks should wait for the new data is extended to three times in a 4 X 4 Basic SVD Processor

The 2X2 matrix can further help us to make nXn matrices where n>=2 through mesh connected array structure that we discussed above .

REFRENCES

[1] R. P. Brent, F. T. Luk, C. Van Loan, “Computation of the Singular value Decomposition Using Mesh-connected Processors,” Journal of VLSI and Computer Systems, vol. 1, no. 3, pp. 242-270, 1985.

[2] R. P. Brent, F. T. Luk, "The Solution of Singular Value and Symmetric Eigenvalue Problems on Multiprocessor Array”, SIAM Journal of Scientific and Statistical Computing, vol. 6, pp69-84, 1985.

[3] A. Ahmedsaid, A. Amira, A. Bouridane, "Improved SVD Systolic Array and Implementation on FPGA", Proc. IEEE Field-Programmable Technology, pp35-42, 2003.

[4] G. E. Forsythe, P. Henrici, " The Cyclic Jacobi Method for Computing the Principal Values of a Complex Matrix", Transactions of the American Mathematical Society, 94(1), pp1-23, January, 1960.

[5] R. Andraka, " A Survey of CORDIC Algorithm for FPGA Based Computers", Proc. of ACM/SIGDA International Symposium on

FPGAs, pp191-200, 1998.

[6] W. Ma, "An FPGA-Based Singular Value Decomposition

Processor", Masters Thesis, University of New Brunswick,

August 2005.

[7] R. Andraka, “A survey of CORDIC algorithm for FPGA based computers”,International Symposium on Field Programmable Gate Arrays, Proc.s of the 1998 ACM/SIGDA sixth international symposium on Field programmable gate arrays, Page(s): 191 -200, 1998.

[8] H. Andrews and C. Patterson, “Singular value decompositions and digital image processing”, IEEE Trans, on Acoustics, Speech, and Signal Processing [see also IEEE Transactions on Signal Processing], Vol.: 24, Issue: 1 , Feb 1976, Pages:26 - 53.

[9] “AN FPGA-BASED SINGULAR VALUE DECOMPOSITION PROCESSOR" by Weiwei Ma

[10] Eigen values and Singular Values (Mathworks.com)