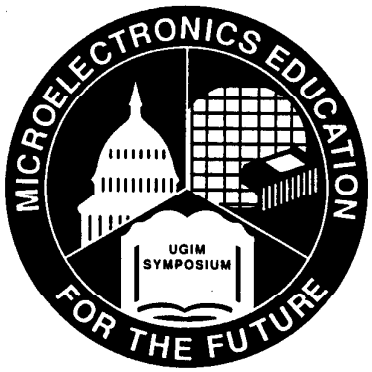
91CH3027-0

PROCEEDINGS



Ninth Biennial
University/Government/Industry
Microelectronics Symposium

June 12-14, 1991
Melbourne, Florida

SEMATECH
INNOVATION FOR AMERICA'S FUTURE

IEEE

F.I.T. Florida Institute of Technology

CAPE - VLSI Implementation of a Systolic Processor Array: Architecture, Design and Testing

Nariankadu D. Hemkumar
Kishore Kota
Joseph R. Cavallaro
Dept. of Electrical & Computer Engineering
Rice Univ., Houston, TX 77251
Phone (713)-527-4020

Abstract

The SVD is an important matrix decomposition in many real-time signal processing, image processing and robotics applications. Special-purpose processor arrays can achieve significant speed-up over conventional architectures through the use of efficient parallel algorithms. The Cordic Array Processor Element (CAPE) is a single chip VLSI implementation of a processor element for the Brent-Luk-VanLoan systolic array which computes the SVD of a real matrix. The array utilizes CORDIC (CO-ordinate Rotation DIgital Computer) arithmetic to perform the vector rotations and inverse tangent calculations in hardware. A six-chip prototype of the processor has been implemented as TinyChips using the MOSIS fabrication service. Experience gained from designing the prototype helped in the design of integrated single chip version. The chip has been implemented on a $5600 \times 6900\mu$ die in a 2μ n-well scalable CMOS process.

1 Introduction

Many real-time signal processing applications require fast computation of matrix decompositions. Special-purpose systolic arrays [10] present an attractive architecture for implementing such factorization algorithms in hardware. The special-purpose nature of these arrays allows the optimization of system performance by exploiting the parallelism inherent in the problems. Systolic arrays, by definition, allow only near-neighbor communication. The reduced path lengths and fixed fan-out allow design of large arrays with very little degradation in the clock speed.

The SVD is an important matrix factorization used in a variety of applications. The SVD exhibits better numerical stability due to the insensitivity to ill-conditioning or rank deficiency of matrices. However, the SVD is computationally intensive, requiring $O(p^3)$, for a $p \times p$ matrix. For example, the SVD of an 8×8 matrix on a SUN 3/60 requires about 1 second of CPU time using the LINPACK routine DSVDC. As a representative real-time signal processing application, the inverse kinematics engine of a robot being constructed at Rice, requires the computation of the SVD of an 8×8 matrix in $400\mu s$. With a square array of processors it is possible to reduce the time complexity for the SVD to $O(p \log p)$.

The SVD [5] of a $p \times p$ matrix M is

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

where U and V are $p \times p$ orthonormal matrices and Σ is a diagonal matrix of singular values. The columns of U and V are called the left and right singular vectors, respectively. The Jacobi method for computing the SVD is amenable to parallel and systolic computation. A sequence of orthonormal transformations are used to diagonalize a matrix M and the SVD is obtained by performing several sweeps of computation. A sweep consists of a sequence of 2-sided Jacobi rotations which annihilate every pair of off-diagonal elements at least once. It has been shown that $O(\log p)$ sweeps are required for convergence. If the sequence of rotations chosen is the parallel ordering [1], then it is possible to implement a sweep consisting of $p(p-1)/2$ 2-sided rotations in $O(p)$ time on a square array of $p/2 \times p/2$ processors.

The Brent, Luk and Van Loan systolic array [1] uses a square array of $p/2 \times p/2$ array of processors for SVD of a $p \times p$ matrix, where each processor has the ability to compute the SVD of a 2×2 matrix. A 2×2 SVD can be described as

$$R(\theta_l)^T \begin{bmatrix} a & b \\ c & d \end{bmatrix} R(\theta_r) = \begin{bmatrix} \psi_1 & 0 \\ 0 & \psi_2 \end{bmatrix}, \tag{2}$$

where θ_l and θ_r are the left and right rotation angles, respectively. The rotation matrix is

$$R(\theta) = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}, \tag{3}$$

and the input matrix is

$$M = \begin{bmatrix} a & b \\ c & d \end{bmatrix}.$$

The angles θ_l and θ_r are given by the relations

$$\begin{aligned} \theta_r + \theta_l &= \tan^{-1}\left[\frac{c+b}{d-a}\right], \\ \theta_r - \theta_l &= \tan^{-1}\left[\frac{c-b}{d+a}\right]. \end{aligned} \tag{4}$$

For a $p \times p$ matrix, the Jacobi rotation is given by

$$\mathcal{J} = \begin{bmatrix} R(\theta) & 0 \\ 0 & I \end{bmatrix} \tag{5}$$

where $R(\theta)$ is a 2×2 rotation and I is a $(p-2) \times (p-2)$ identity matrix. A 2-sided rotation for a $p \times p$ matrix consists of a left and a right Jacobi rotation. Any pair of off-diagonal elements can be annihilated through a 2-sided rotation where the rows and columns of the $\mathcal{J}$s are appropriately permuted. In a square systolic array (Figure 1), the transformations to annihilate the off-diagonal elements of the $p/2$ diagonal processors are independent of the sequence in which these elements are annihilated, and hence can be evaluated concurrently. The angles corresponding to these transformations are computed by the diagonal processors using (4). The rest of the matrix is then updated by systolically propagating the left angles horizontally and right angles vertically to transform all the 2×2 submatrices residing in the various processors. The data is then permuted to present a new set of off-diagonal elements at the diagonal. In the array, application of the previous set of transformations and computation of a new set of rotations is pipelined. This leads to waves of activity originating at

0-7803-0109-9/91/0000-0064$01.00 © 1991 IEEE

the main diagonal and systolically propagating towards the northeast and southwest corners of the array. The data exchange algorithm, called the parallel ordering, has the property that the next set of data items to be annihilated is always available in the diagonal neighbors of the main diagonal. Thus all data exchanges are near-neighbor.

2 CORDIC Arithmetic

In special-purpose VLSI processors, an efficient design with optimal area and time complexity can be obtained by using special arithmetic techniques which map the desired computations to hardware. The principal computations involved in the SVD algorithm are vector rotations and inverse tangent calculations which can be implemented efficiently using the CORDIC technique. Other operations required are addition, subtraction and divide by two which can utilize the same functional units as CORDIC.

The Coordinate Rotation Digital Computer (CORDIC) technique was initially developed by Volder [14] as an algorithm to solve the trigonometric relationships that arise in navigation problems. Involving only a fixed sequence of additions or subtractions, and binary shifts (Figure 2), this scheme was used to quickly and systematically approximate the value of a trigonometric function or its inverse. This algorithm was later unified for elementary functions by Walther [15].

The basic iteration of the circular mode of operation in CORDIC implements a 2×2 Jacobi rotation (3) of a vector and can be expressed as

$$\begin{aligned} x_{i+1} &= x_i + \delta_i y_i 2^{-i} \\ y_{i+1} &= y_i - \delta_i x_i 2^{-i} \\ z_{i+1} &= z_i + \delta_i \alpha_i \end{aligned}$$

where,

$$\begin{aligned} \alpha_i &= \tan^{-1}(2^{-i}) \\ \delta_i &\in \{-1, +1\} \end{aligned}$$

The parameter δ_i is chosen at each iteration to either force z_0 to 0 (*Rotation or Z-reduction*) or y_0 to 0 (*Vectoring or Y-reduction*). Each iteration corresponds to a vector rotation of $[x_i, y_i]^T$ through an angle α_i with the direction of rotation determined by δ_i, followed by a scaling by a factor $1/\cos\alpha_i$. The variable z_i is either used to store the initial angle or accumulate the resulting angle. Z-reduction corresponds to an anti-clockwise rotation of the vector $[x_0, y_0]^T$ through an angle z_0. A post-processing step is required to eliminate the constant scale factor accumulated at each iteration. Y-reduction is used to compute $\tan^{-1}(y_0/x_0)$.

The CORDIC technique needs modification in order to be used in the SVD processor. In y-reduction, traditional CORDIC utilizes the sign of y_i at each iteration to determine the δ_i for the next iteration. In the modified CORDIC algorithm [16], the exclusive-or of the sign bits of x_i and y_i is used to determine this sign. Otherwise, in order to allow CORDIC to converge for vectors with a negative x component, a pre-processing step is necessary to invert the signs of both x_0 and y_0 if x_0 is negative.

In the computation of $\tan^{-1}(y_0/x_0)$, the actual magnitude of x_0 and y_0 should not affect the accuracy of the computed angle. However, in traditional CORDIC small values for x_0 and y_0 result in a loss of precision greater than that for larger magnitudes. Thus normalization of the values for inverse tangent calculations can produce better results. A straightforward way to implement this would be to use a pre-processing step in which x_0 and y_0 are shifted left by the maximum amount which will not cause an overflow in subsequent CORDIC iterations. A single clock cycle implementation of this scheme would require a barrel shifter with $O(n^2)$ area complexity and a leading-zero encoder with $O(2^n)$ complexity. In this paper a novel scheme [9] for fixed point CORDIC which allows the normalization to be made part of the CORDIC iterations without any time penalty, is presented. The scheme modifies the CORDIC iterations with a small left shift of the variables x_i and y_i, if their magnitude is too small. Within a few iterations the variables are normalized and do not need any more shifts. This scheme achieves results identical to the scheme which uses a pre-processing step. The above technique reduces the hardware complexity for normalization from $O(2^n)$ to $O(n \log n)$ where n is the bit precision. This reduction was achieved by using a shifter which implements less than $\log n$ shifts and a leading-zero encoder for $\log n$ bits.

Since the basic computations required for a 2×2 SVD map directly to the primitives provided by CORDIC, a 2×2 SVD processor reduces to the following algorithm [4]:

Algorithm CORDIC SVD( )

begin
Use CORDIC Y-Reduction to find rotation angles;
Use CORDIC Z-Reduction to transform the 2×2 matrix;
end

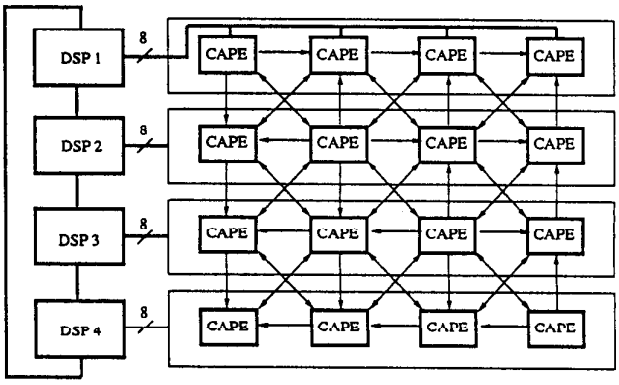


Figure 1: A linear array of DSP chips interfaced to a square array of special-purpose CAPE chips for an 8-degree of freedom robot

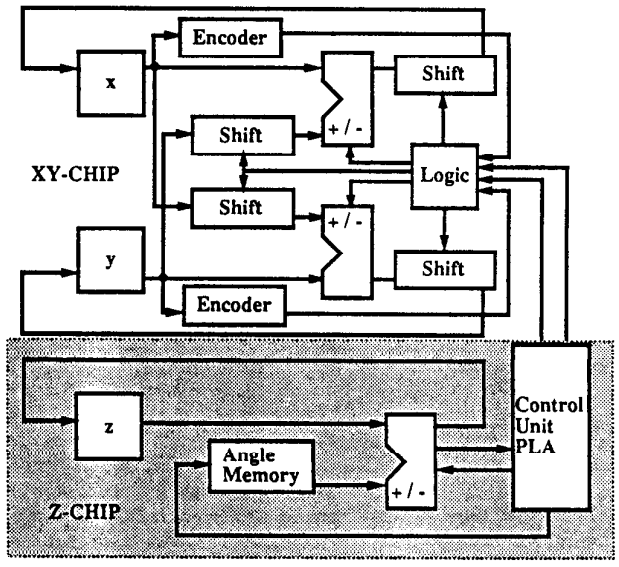


Figure 2: Block Diagram of CORDIC

3 Design of the SVD array

Figure 1 shows the inter-connection of processors to implement the systolic array. The square array of special-purpose SVD processors is connected to a linear array of general purpose DSP chips which are used for general purpose computations. The array serves as a co-processor for matrix factorizations, in particular, the SVD. The CAPE chip has been designed to allow the array to be used for matrix factorizations other than the SVD. In particular, it would be possible to compute the QR decomposition and Eigenvalue decomposition of a symmetric matrix on the array. These are some of the computations which are encountered in many engineering problems.

The CORDIC systolic processor array (CSPA) is a mesh-connected array of processors with additional diagonal links to facilitate data exchanges. Horizontal and vertical links are used to systolically propagate the left and right rotation angles, respectively.

CAPE is a single chip implementation of the CORDIC SVD processor [4]. The SVD algorithm discussed in § 1 and the CORDIC algorithm described in § 2 form the basis for the design of the CORDIC-SVD processor. The processor was designed in two phases. Initially, a six chip prototype of the processor was designed. Each component chip was fabricated as a "TinyChip" from the MOSIS fabrication facility. Implementation of the processor as a set of chips provided controllability and observability which was essential at that stage of design. The prototype served as a means of exhaustively testing every aspect of the design which is not possible with simulation. Digital logic external to the chips could be used to correct inconsistencies in the interaction between the sub-units. The overall design was then verified in spite of minor design errors.

The second phase involved designing a single chip version of the processor. This chip utilized many of the basic blocks from the six-chip prototype in an enhanced architecture. The single chip version utilized better layout techniques using higher level design tools. Some of the lower level VLSI layout issues are discussed in § 6.

4 Architecture of the Prototype

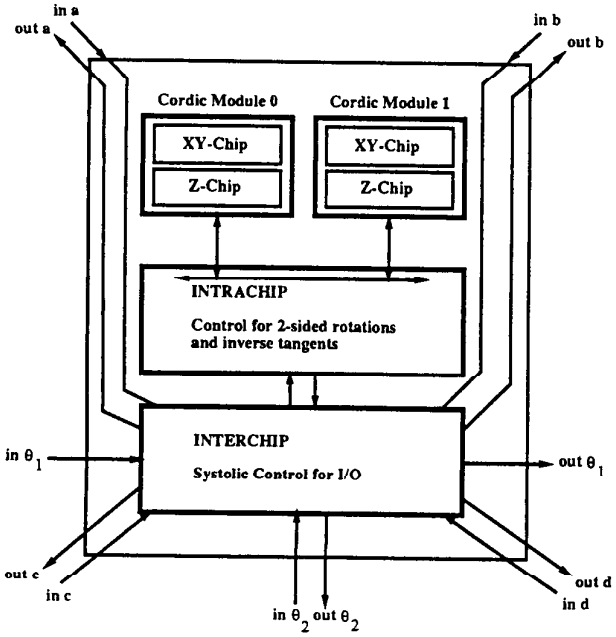


Figure 3: **Block Diagram of the Six-chip Prototype**

The basic structure of the CORDIC SVD processor was discussed by Cavallaro [2]. The architecture, shown in Figure 3, consists of four distinct modules which have been implemented as four different TinyChips. Two of the chips, the *xy*-chip and the *z*-chip, together form one CORDIC module. Maximum parallelism is achieved in the computation of the 2×2 SVD by including two CORDIC modules in the processor. The control and inter-processor communication module for the SVD processor was implemented in two separate chips. The *intracontroller* chip is used to provide all the internal control necessary to implement the computation of the 2×2 SVD. The systolic array control is provided by the *intercontroller* chip. A hierarchical control mechanism provides a way to effectively shield the low level control details from the higher level controllers. At each level the controllers provide a set of macros to the controller next in the hierarchy. In addition, this separation of control allows a degree of concurrency not possible with a single controller. The next few sections describe the individual chips in greater detail.

4.1 Design of the XY-Chip

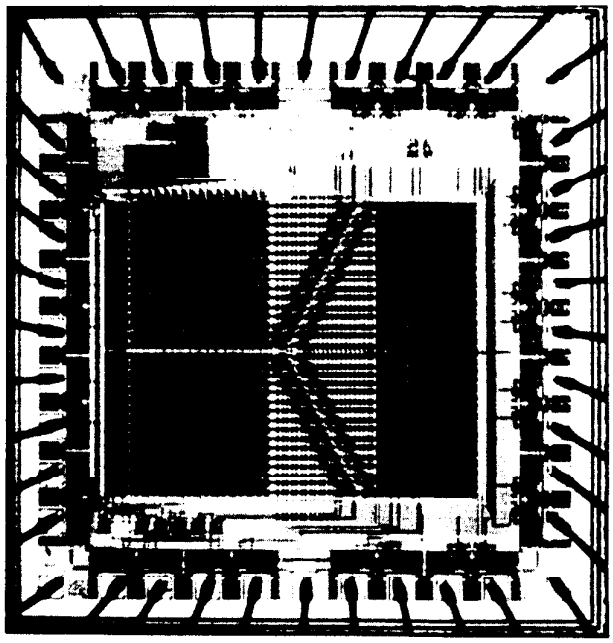


Figure 4: **Die Photograph of the XY-Chip**

Figure 2 shows an implementation of a 16-bit fixed point CORDIC processor. The xy-chip implements the x and y data paths of a single CORDIC module. A previous implementation of the xy-chip was reported by Cavallaro [3]. Each path consists of a 16-bit register, a 16-bit barrel shifter implementing all right shifts upto 15, a 16-bit ripple-carry adder with associated temporary latches at the inputs, and a data switch to implement the cross-coupling implied by the x and y CORDIC iterations. Both CORDIC iterations and scale factor correction iterations are possible on the data stored in the registers. The absence of any on-chip control allows an external controller complete flexibility of the number of iterations as well as the nature of each iteration. Read and write access to the x and y registers is possible independent of the computation. Figure 4 shows a die photograph of the xy-chip.

4.2 Design of the Z-Chip

The z-chip complements the xy-chip to form a complete 16-bit fixed point implementation of the CORDIC module. It implements the z data path which consists of a 16-bit z-register, a carry-lookahead adder and a ROM table of 16 angles. It also implements a controller to provide the control signals for the xy datapaths in the xy-chip and the z datapath in the z-chip. The controller allows use of the CORDIC module in five modes: z-reduction, y-reduction, scale factor

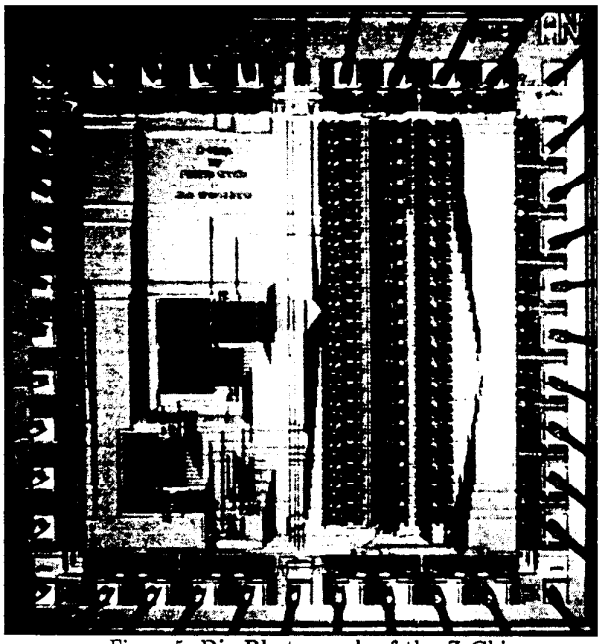


Figure 5: **Die Photograph of the Z-Chip**

correction, single add and subtract, and divide-by-two. These are the only operations required in the computation of the SVD. Figure 5 shows a die photograph of the z-chip.

4.3 Design of the Intra-chip

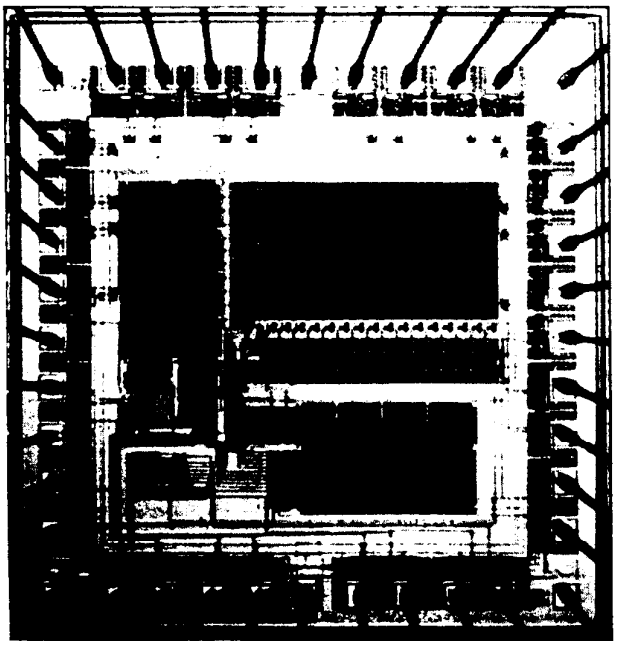


Figure 6: **Die Photograph of the Intra-Chip**

The intra-chip controls data movement between a register bank and the two CORDIC modules and sequences the primitives provided by the z-chip to compute the SVD of the 2×2 matrix stored on the processor. It allows operation in three modes corresponding to processor behavior in three different positions in the array. If the processor is on the main diagonal of the array, it computes the left and right angles and then uses these angles to diagonalize a 2×2 submatrix. Any off-diagonal processor only transforms a 2×2 submatrix with the angles received from the previous processors. However, if a processor is diagonally a multiple of three away from the main diagonal, in addition to the transformation, a delay equal to the time it takes to compute the angles is included. This delay is necessary to maintain synchronization between all the processors due to the wavefront nature of the array. These three modes allow the same processor to be used in any position of the array in spite of the required heterogenity. The aforementioned storage is a register bank consisting of ten registers implemented in the intra-chip. A 16-bit bus interconnects these registers and the two CORDIC modules. Figure 6 shows a die photograph of the intra-chip.

4.4 Design of the Inter-chip

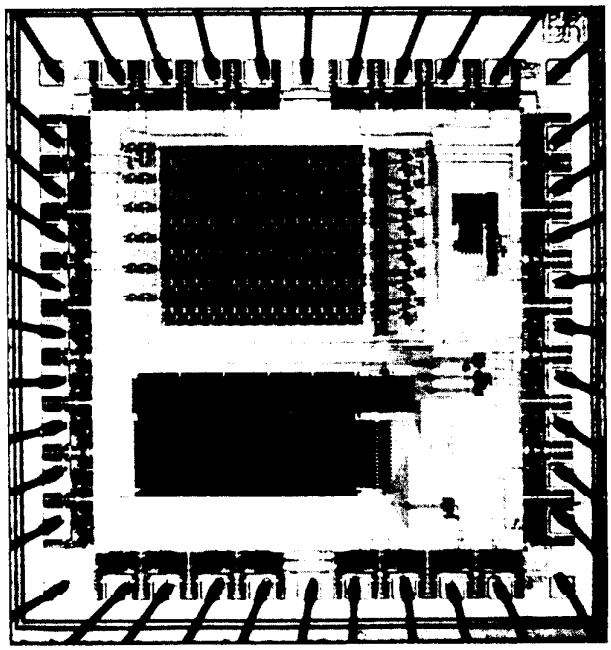


Figure 7: **Die Photograph of the Inter-Chip**

The inter-chip contains the controller which exercises the highest level of control in the processor. As a communication agent, it implements the systolic array control for the SVD processor. The parallel ordering data exchange [1] which provides a new pair of data elements at the main diagonal to be annihilated, is implemented in this chip. A set of six shift registers is used as communication buffers to store a copy of the data to be exchanged with the other processors while the intra-chip operates concurrently on its own private copy of the data.

Interprocessor communication is through a set of eight serial links. Four of these links are hardwired to exchange data with the diagonal neighbors. Four other links serve to systolically propagate the left angles horizontally and and right angles vertically. Since the SVD algorithm is compute bound, the bandwidth provided by the serial links is sufficient for the data interchange. In addition, pin limitations prohibit implementation of parallel ports to communicate with neighboring processors. Data is exchanged synchronously between processors. Data is shifted out to the neighboring processor and simultaneously shifted in from it. Synchronization in the array is of utmost importance since no other form of synchronization is provided. This is in accordance with the definition of systolic arrays [10] which emphasizes exploiting the special-purpose nature of the array to eliminate the overhead of synchronization. In the prototype, pin limitations forced a serial link for communication of data with the intra-chip. The inter-chip controller also allows initialization of the array by loading the matrix data. Figure 7 shows a die photograph of the inter-chip.

5 Architecture of CAPE

The CORDIC array processor element (CAPE) is a single chip implementation of the CORDIC SVD processor. This chip incorporates elegant solutions to the problems encountered in the six-chip prototype. CAPE also includes additional details necessary to construct a working systolic array. CAPE essentially consists of the same basic cells as the prototype. Many of the architectural enhancements were made to reduce the communication required within the chip. A completely new loading scheme was added to facilitate easy interface with a variety of hosts. The next few paragraphs describe the details of the improvements.

A study of the prototype CORDIC units indicated that better numerical accuracy could be achieved with some modifications to the modules as described in § 2. To improve the numerical accuracy of the inverse tangent calculations, a normalization shifter was included in each of the x and y data paths. These shifters are activated during y-reduction by two leading-zero encoders which monitor the magnitude of the x and y data. Simulations indicate that this scheme provides several extra bits of accuracy in many cases.

The CORDIC modules in the prototype converge only for vectors in the first and fourth quadrants while operating in the y-reduction mode. The modified scheme uses the exclusive-or of the sign bits of the x and y registers to control the next iteration. This effectively increases the range of convergence for the CORDIC modules. In addition, the above scheme adds symmetry to the x and y data paths since they cannot be distinguished externally. This allows the same CORDIC module to rotate a vector in either clockwise or anticlockwise direction. Some data movement is rendered unnecessary by the above method, for example, if the data to be used in the next computation already resides in the x and y registers in the wrong order, since the definition of the x and y registers can be interchanged, no data movement is required.

The z-chip was modified to allow mapping the x, y and z registers as part of the register bank. This allows reuse of the data left in the registers by a previous computation which is important when attempting to cut down the communication overhead.

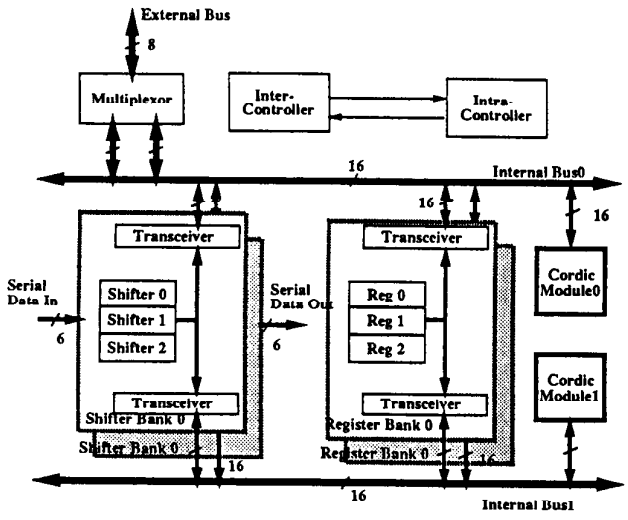


Figure 8: Internal Architecture of CAPE

A study of the prototype SVD processor showed that there was a large overhead incurred in moving data to the appropriate registers between computations. A double bus architecture, as shown in Figure 8, was used to minimize this communication. A separate CORDIC module is hardwired to communicate on each bus. The set of of six registers is split into two banks which can communicate with either bus allowing simultaneous access to two different registers. The number of registers required in the architecture was reduced from ten to six due to mapping of the CORDIC registers as general purpose registers. The double bus allows concurrent data movement to the two CORDIC modules which reduces communication cycles by half. Using the double bus architecture did not cause an increase in the area of the chip. This was a result of the reduction in the size of the intra-chip controller to half its original size. This corresponds to a reduction in the number states due to greater concurrency provided by the architecture.

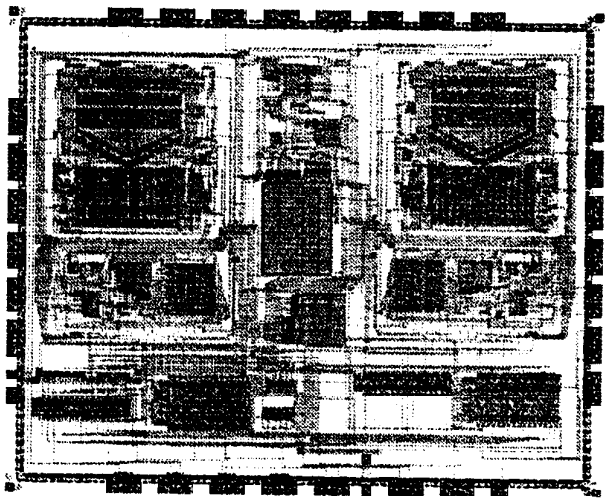


Figure 9: Layout of CAPE

Special cases that occur in the SVD computation require extra hardware. Two zero comparators were included to detect the case $\tan^{-1}(0/0)$. This is an invalid computation on the CORDIC modules which return a code equivalent of the floating-point *NaN*. The SVD algorithm allows such computed angles to be replaced by zero. The zero comparators compare the data on the bus with zero and return a control signal to the intra-chip controller to handle this special case. A literal constant zero can be driven onto either bus and hence into any register. This allows forcing a register to store a zero to handle special cases.

The inter-chip has been modified to handle an improved array loading scheme. The prototype did not make a special provision to efficiently load the array. Since CAPE is to be used in a systolic array in a real system, the interface with the host has been refined. A parallel port has been included along with the requisite handshake signals. The controller implements an asynchronous loading scheme which allows interfacing with a host running at an independent clock rate.

The prototype inter-chip was pin limited and hence forced to communicate with the intra-chip through a serial link. This restriction is no longer valid in the single chip implementation. Consequently, this serial link was replaced with an internal parallel bus which reduces the time to exchange data items between the intra-chip and the inter-chip. The rest of the inter-chip was redesigned with only minor changes.

6 VLSI Implementation

The six-chip prototype was implemented by dividing the system into four logical sub-units. This approach inherently allows division of labor. The different design groups involved could work in parallel with the minimum of interaction once the component sub-units had been identified and their interactions defined. Different versions of the sub-units were implemented to explore design alternatives. Combining the verified sub-units to obtain a working prototype was then a simple task. A structured VLSI design approach was used

following the Mead and Conway methodology [11]. Static complementary (CMOS) logic was used to simplify design. In addition, a formal two-phase clocking strategy was imposed [7, 8]. The individual units were developed using the *Magic* [12] layout editor and other VLSI tools from the University of California, Berkeley. Layout was done in a bit-slice manner. Programmable Logic Arrays (PLAs) were generated automatically from a high level description. Thus, a semi-custom design style was adopted. Switch-level simulation was performed using *esim* and *irsim*. Longest path delays were identified with *crystal*. Detailed simulation at the circuit level was performed for critical cells using *spice*. Chips were designed using the scalable CMOS technology from the MOSIS fabrication service. The single chip utilized the better routing facilities available with OCTTOOLS, also from Berkeley [13]. The design times for future versions of CAPE can be considerably sped up by using a symbolic layout style using these tools. The TinyChips were fabricated on $2200 \times 2200\mu$ die using a 2μ n-well process. The transistor count for the different chips varied between 4500 and 6000 transistors. The single chip version CAPE was designed on a $6900 \times 5600\mu$ die and contained about 26000 transistors. A plot of this chip is shown in Figure 9.

PC-based high speed test equipment, Omnilab from Orion Instruments [6], was used in verifying operation of the chips. This tester allows testing upto a frequency of 17 MHz which corresponds to a clock frequency of 4.25 MHz for the chip, due to the two-phase clocking strategy. All the components operate at the maximum test frequency. Exhaustive functional testing of some of the components was done using the parallel ports provided on an interface card on an IBM PC-AT. A C-language library of routines was created to use the CORDIC module as a co-processor for the IBM PC. This allowed an exhaustive test of its numerical stability.

7 Current Work and Summary

The CAPE project, although research oriented, provided sufficiently diverse and challenging VLSI projects that could be implemented over the period of an academic semester and within the confines of a course in VLSI and CAD. The six chip prototype has been built and tested exhaustively. A single chip version of the processor has been sent for fabrication. The principal funding for fabrication of the sub-units came from the VLSI teaching grants provided by NSF. The integration of the sub-units into a single chip was funded through a research initiation grant from NSF.

Future versions of CAPE can implement floating point CORDIC units. Although only the SVD of a real matrix has been implemented, an architecture for the SVD of a matrix with complex data has also been developed. This array can be developed starting form the current design. An architecture has been developed to compute subsidiary matrices like U and V matrices in (1). Additional goals include implementation of a fault-tolerant array. These versions can benefit from a shorter design cycle due to better place and route VLSI tools like OCTTOOLS.

Acknowledgments

This work was supported in part by the National Science Foundation under Research Initiation Award MIP-8909498.

References

[1] R. P. Brent, F. T. Luk, and C. F. Van Loan. Computation of the Singular Value Decomposition Using Mesh-Connected Processors. *Journal of VLSI and Computer Systems*, 1(3):242–270, 1985.

[2] J. R. Cavallaro. *VLSI CORDIC Processor Architectures for the Singular Value Decomposition.* PhD thesis, School of Electrical Engineering, Cornell Univ., Ithaca, NY, August 1988.

[3] J. R. Cavallaro, M. P. Keleher, R. H. Price, and G. S. Thomas. VLSI Implementation of a CORDIC SVD Processor. *Proc. 8th Biennial University/Government/Industry Microelectronics Symposium*, pages 256–260, June 1989.

[4] J. R. Cavallaro and F. T. Luk. CORDIC Arithmetic for an SVD Processor. *Journal of Parallel and Distributed Computing*, 5(3):271–290, June 1988.

[5] G. H. Golub and C. F. Van Loan. *Matrix Computations, Second Edition.* Johns Hopkins Univ. Press, Baltimore, MD, 1989.

[6] Orion Instruments. *The* **OmniLab**^{TM} *User's Manual.* Menlo Park, CA, June 1990.

[7] K. Karplus. Exclusion Constraints, A New Application of Graph Algorithms To VLSI Design. *Advanced Research in VLSI, Proc. 4th MIT Conference*, pages 123–139, April 1986.

[8] K. Karplus. A Formal Model for MOS Clocking Disciplines. Technical Report TR 84-632, Dept. of Computer Science, Cornell Univ., Ithaca, NY, August 1984.

[9] K. Kota. Architectural, Numerical and Implementation Issues in the VLSI Design of an Integrated CORDIC SVD Processor. Master's thesis, Rice University, Department of Electrical and Computer Engineering, May 1991.

[10] H. T. Kung. Why Systolic Architectures? *IEEE Computer*, 15(1):37–46, January 1982.

[11] C. A. Mead and L. A. Conway. *Introduction to VLSI Systems.* Addison-Wesley, Reading, MA, 1980.

[12] J. K. Ousterhout, G. T. Hamachi, R. N. Mayo, W. S. Scott, and G. S. Taylor. Magic, a VLSI Layout System. *ACM/IEEE 21st Design Automation Conference*, pages 152–159, June 1984.

[13] Rick Spickelmier. *Octtools Distribution, Release 4.0*, UC. Berkeley, August 1990.

[14] J. Volder. The CORDIC Trigonometric Computing Technique. *IRE Trans. Electronic Computers*, EC-8(3):330–334, Sept. 1959.

[15] J. S. Walther. A Unified Algorithm for Elementary Functions. *AFIPS Spring Joint Computer Conf.*, pages 379–385, 1971.

[16] B. Yang and J. F. Böhme. Reducing the computations of the SVD array given by Brent and Luk. *SPIE Advanced Algorithms and Architectures for Signal Processing*, 1152:92–102, 1989.