THE DESIGN OF A FIXED-POINT DIGITAL RESOLVER

by

Michael Andrews

A Thesis Submitted to the Faculty of the

DEPARTMENT OF ELECTRICAL ENGINEERING

In Partial Fulfillment of the Requirements
For the Degree of

MASTER OF SCIENCE

In the Graduate College

THE UNIVERSITY OF ARIZONA

1 9 6 8

STATEMENT BY AUTHOR

This thesis has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED: Michael Andrews

APPROVAL BY THESIS DIRECTORS

This thesis has been approved on the date shown below:

G. A. KORN
Professor of Electrical Engineering

7/24/68
Date

GAKorn for J.V. Wait
J. V. WAIT
Associate Professor of
Electrical Engineering

7/24/68
Date

PREFACE

This thesis illumines one small area in the vast sea of knowledge. The author hopes to enlarge this area. Of course, an understanding such as this demands the gratitude toward those who patiently guided this author towards his goal. In particular, the author wishes to thank Professor Wait and Professor Korn of The University of Arizona whose continued inspiration made possible this work. And to my devoted wife and typist, a bouquet of flowers!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATIONS

ABSTRACT

This paper describes the design of a high-speed digital resolver to calculate R sin Θ and R cos Θ by the cross-addition algorithm applied to vector rotation (similar to Volder's CORDIC method).

A proposed design of a peripheral "black box" that may be called as a hard-wired subroutine is included. The results of a computer simulation of this system on the PDP-9 digital computer is described along with error tabulations. The paper briefly analyzes other methods of calculating sine and cosine, including polynomial and rational functions, digital differential analyzer and table look-up approximation methods. It is shown that digital computers operating in fixed-point mode must utilize shift instructions to retain useful accuracy. Thus, these programs essentially become floating point subroutines and are subsequently slow, unless floating point hardware is available.

Using MECL logic the digital resolver proposed in this paper calculates the sine and cosine with a 20 bit word length simultaneously in less than 15 micro-seconds with an accuracy of .0001 per cent of full scale.

CHAPTER I

THE PROBLEM - REAL-TIME COMPUTATION OF THE SINE AND COSINE FUNCTION

It has long been recognized that the general-purpose digital computer can be used for dynamic-system simulation and the solution of simultaneous differential equations, since it allows high bit accuracy coupled with relatively simple arithmetic and control circuitry.

The solution of sets of simultaneous differential equations performed by numerical approximation methods using iterative procedures requires considerable execution time on general-purpose digital computers. Equations with many trigonometric functions also burden the computing time of general-purpose digital computers because they must resort to approximating expansions in terms of rational functions, power series or by means of table look-ups.

This paper is primarily interested in computers operating in the fixed-point mode. However a typical rational function approximation utilizing

floating-point hardware is examined. A comparison is made of typical methods including polynomial, rational, digital differential analyzer (DDA) and table look-up. The major parameters guiding this analysis are speed and accuracy. The analyses in this paper focus on sine and cosine generators with 20 bit word lengths. Computer subroutines are compared to peripheral hardware "black box" approaches and it is shown that on a speed-accuracy basis, the Cross-Addition Digital Resolver mated with a high speed I/O channel of a general-purpose digital computer provides the most expeditious approach towards real-time computation of the sine and cosine.

CHAPTER II

POLYNOMIAL APPROXIMATIONS

The following polynomial approximations of the sine function are seldom used as digital program subroutines because of errors generated. Nevertheless, analysis of their behavior illustrates some considerations involved in designing sine-cosine generating subroutines. The chief problem is to properly locate the binary point throughout machine execution to insure retaining the useful accuracy of the computer.

Taylor Series Approximation

The Taylor series approximation is shown below:

$$\sin \theta = \theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \cdots \frac{(-1)^n \theta^{2n+1}}{(2n+1)!} \quad ; \quad n = 0,1,2,\ldots$$

$$\text{Truncation Error} < \left| \frac{\theta^{2n+3}}{(2n+3)!} \right|$$

Hence the accuracy of this approximation depends

largely on the number of terms used. For example, an accuracy of five decimal places requires the retention of terms up to $\frac{\theta^7}{7!}$. Moreover, the machine handling of numerous multiplications necessitates some skillful programming to utilize the full word length efficiently.

As an example, consider the following. If the binary point is interpreted one place to the right of the machine point, the binary point moves five places to the right of the machine point when the number is raised to the fifth power. The programmer is now forced to consider the new binary point position. Therefore, the reduction of leading zeroes necessitates left and right shifting of numbers during the machine calculation sequence. Machine time increases and the programmer seeks a compromise to scale the numbers for different angular arguments in order to retain the same level of accuracy over the full range of arguments.

Computer Programs (Appendix A, Parts I and II)

The polynomials used for the calculation of the sine and cosine are manipulated to reduce machine execution time by Horner's method (McCracken, 1964). A typical polynomial used was

$$\sin X = X(1 - (\tfrac{1}{6}(X(X - (\tfrac{1}{20}(X(X(X)))))))$$

The two PDP-9 computer programs for the Taylor series approximation, though rather crude, illustrate the increase in execution time due to binary point shift requirements in order to retain significant bits. These programs calculate the sine function to at best three decimal places. Yet, the computer word length is equivalent to almost six decimal places. Any further accuracy burdens execution times by requiring numerous shift routines and double precision arithmetic operations.

The execution time of this program on the PDP-9 computer is 100 usec excluding pre- and post-execution sequences. The pre-execution sequence determines the quadrant of the argument and stores the sign of the final answer. The post-execution sequence places the algebraic sign on the final answer. The combined pre- and post-execution sequences include four memory reference instructions and three operate instructions. This requires an additional 20 usec. Total execution

time for this three term approximation exceeds 120 usec.

Error Analysis

A typical computer execution using the programs in Appendix A, Parts I and II, generated the errors listed in Table 1. Note that two separate programs are required for the different ranges of the argument. The table indicates that truncation errors could be significantly reduced with a five term series.

Hastings (1955) shows the curve of relative error versus argument for a three term approximation, Fig. 1. A comparison of the relative error of Fig. 1 to the errors in Table 1 indicates that machine-introduced errors due to round-off constitute the major portion. The solution requires further reduction of loss of significant bits during the execution, an impossibility with single-precision fixed-point multiplication on the PDP-9 computer. Hence the necessity of a floating-point subroutine becomes evident.

Table 1

Error Generation - Taylor Series (Three Term) Approximation on PDP-9 Computer

θ Radians	$\sin\theta$ Actual	$\sin\theta$ Computed	Fractional Error $^*\Delta E$
0.10000	0.099833	0.099334	+ .005
0.90000	0.783327	0.787487	- .005

$$^*\Delta E = \frac{\sin\theta_{actual} - \sin\theta_{computed}}{\sin\theta_{actual}}$$

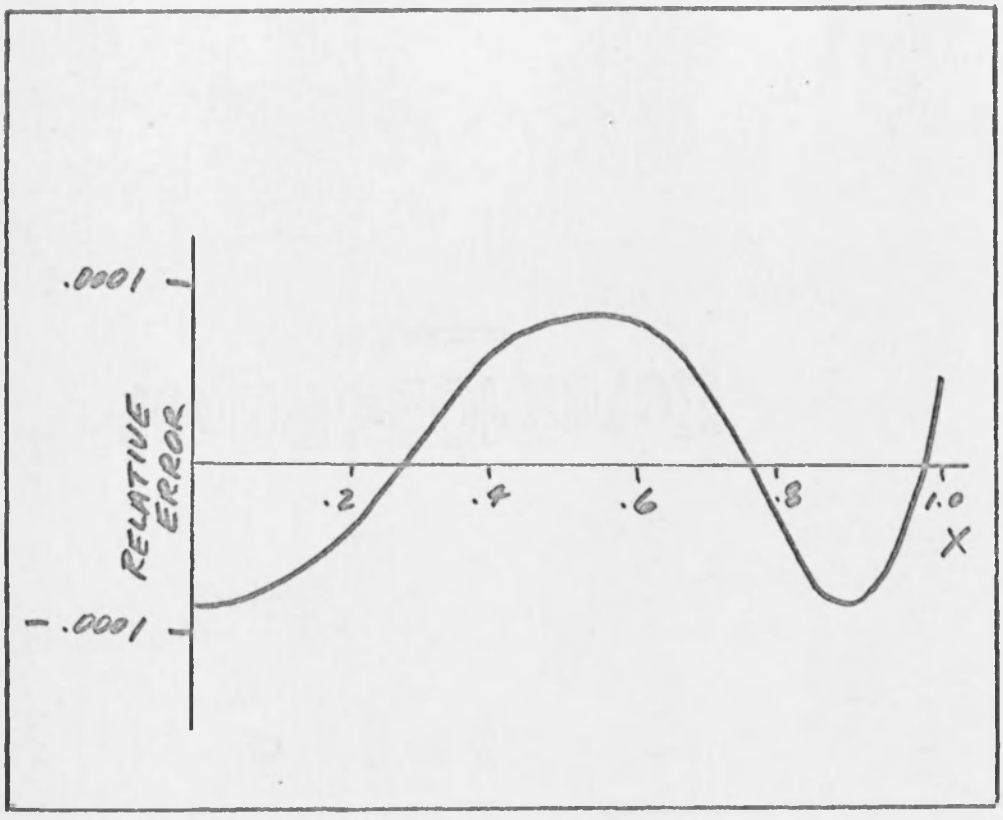


Fig. 1. Relative Error For Taylor Series (Three Term) Approximation to Sine Function

Chebyshev Series Approximation

Studies by Booth (1955) have shown that when the accuracy of work demands the full precision of the computer word length, the Chebyshev series approximation for the sine function is best since it gives function values to the required precision with reduced complexity of programming.

Using Chebyshev polynomials, we obtain, Lanczos (1952):

$$\sin \tfrac{\pi}{2}\theta = + 1.57079\ 6326\ \theta - 0.64596\ 4102\ \theta^3 + 0.07969\ 2704\ \theta^5 - 0.00468\ 1984\ \theta^7 + \ldots$$

for $0 \leq \theta \leq 1$

An analysis of this approximation is quite similar to the Taylor series analysis presented previously. The computer programs are also similar except for the multiplication constants (Appendix A). The machine-introduced error is essentially caused by round-off of the coefficient A_n, Clenshaw (1954). Since the programs are

similar, the majority of the total error is due to round-off error.

The Chebyshev series approximation is sometimes desirable since it requires a shorter computer program over the full range of the argument (Appendix A, Part III). However, its errors are slightly greater than the pair of Taylor series approximations. Execution time including overhead on the PDP-9 computer is approximately 120 usec. Table 2 lists the errors generated in a typical computer execution found in Appendix A, Part III.

Table 2

Error Generation - Chebyshev Series (Three Term) Approximation on PDP-9 Computer

θ Radians	Sin θ Actual	Sin θ Computed	Fractional Error *ΔE
0.10000	0.099833	0.099840	- .007
0.90000	0.783327	0.783323	+ .006

$$*\Delta E = \frac{\text{Sin}\,\theta_{\text{actual}} - \text{Sin}\,\theta_{\text{computed}}}{\text{Sin}\,\theta_{\text{actual}}}$$

CHAPTER III

RATIONAL FUNCTION APPROXIMATIONS

Rational function approximations are common to most library subroutines of digital computers with floating point capability (Scientific Computing Services, 1966). The following example is included to show the excessive amount of time required to execute this subroutine (DEC Science Library, 1967). Accuracy is better than 26 bits.

$$*\sin x = \sum_{i=0}^{n} C_{2i+1} Z^{2i+1}; \quad n = 4$$

$$C_0 = 1.57079$$
$$C_1 = -0.64593$$
$$C_2 = 0.079689$$
$$C_3 = -0.004673$$
$$C_4 = 0.0001514$$

*Quadrant selection determined by the program. Refer to DEC Science Library, 1967.

The overhead for this execution sequence requires 264 microseconds for the initial conversion from integer to floating-point format and 180 microseconds for conversion back to integer format. Quadrant decision sequences are included in the actual subroutine for the sine and cosine. This subroutine requires 9.4 milliseconds and 9.9 milliseconds respectively (DEC Science Library, 1967) for a combined total of 19.7 microseconds.

CHAPTER IV

PARALLEL DIGITAL DIFFERENTIAL ANALYZER (DDA)

The parallel DDA is characterized by the basic unit, an integrator. The simplest type of integrator implements

$$\theta_{n+1} = \theta_n + \Delta X \dot{\theta}_n \quad \text{(Euler Integration);}$$

it consists of essentially two serial adders and a fixed program control section that gates pulses from one register to another under prescribed conditions.

Sine and Cosine Generator

By interconnecting these integrator units in much the same way as with integrators in analog computers, one uses the DDA to solve differential equations with preset initial conditions and scaled integrator inputs. The sine and cosine functions are generated in the following manner. If $u = \sin\theta$ and $v = \cos\theta$, the inputs are:

$$du = \cos\theta\, d\theta = v\, d\theta$$

$$dv = -\sin\theta\, d\theta = -u\, d\theta$$

These equations govern the operation of the sine and cosine generator in Fig. 2. With register capacities of 2^n bits, an output from the accumulator registers will occur when $vd\theta$ and $ud\theta = 2^n$. Then

$$du = v\frac{d\theta}{2^n}$$

$$dv = -u\frac{d\theta}{2^n}$$

In the solution of sine-cosine functions in the rectangular coordinate system, the inputs to the integrand registers become

$$dx = R\cos\theta\,\frac{d\theta}{2^n}$$

$$dy = -R\sin\theta\,\frac{d\theta}{2^n}$$

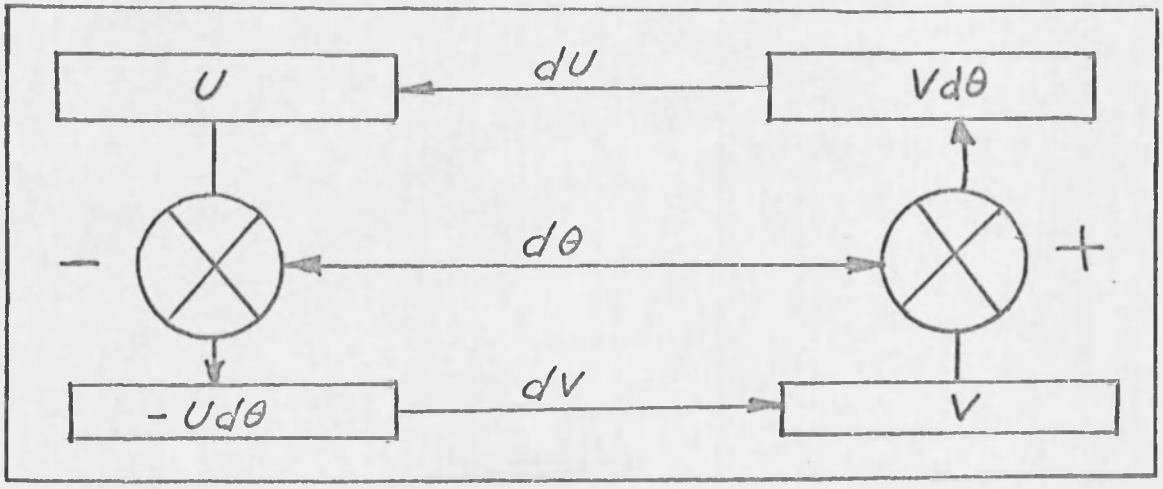


Fig. 2. DDA Sine and Cosine Generator

Limitations

Using this scheme, Truitt (1961) points out that the total available register length after prescaling depends on step size (dy) and the maximum loop gain. The DDA variable is a numeric quantity held in a register of n decimal (or binary) stages. The number of significant bits of the variable is 10^n (or 2^n). However, prescaling requirements sometimes reduce the available register length. This loss of the number of significant bits must be overcome by a corresponding decrease in step size (dy), subsequently an increase in machine speed if a level of accuracy is to be maintained.

Korn and Korn (1964) show that scaling a real-time sine wave $X(t) = A \sin 2\pi ft$ so that $X < \frac{n}{2\pi f}$ bits out of the available register length must not correspond to more than n bits per second since $\frac{dx}{dt}_{max} = 2\pi fA$. Thus with $n = 10^5$ increments per second, the DDA represents a 16 Hz sine wave with at best a 0.1 per cent accuracy. Furthermore, unless the generated errors are estimated and corrected, the simple numerical integration techniques such as the Euler or even open-trapezoidal integration commonly used seriously effect accuracy in fast calculations,

Nelson (1962) and Wait (1963). A four-step sequential operation and a two-step simultaneous operation proposed by Harris (1965) offer error minimization techniques, unfortunately with attendant loss of speed.

DDA Considerations in Combined Computation

Externally the DDA behavior is in many respects similar to a low speed analog computer with sampled outputs. The DDA can be preprogrammed with a patch panel. Initialization and scaling of integrators is similar to that required for analog computer integraters. As shown above, the accuracy and resolution of the polar to rectangular coordinate conversion depends on the step size or iteration rate of the DDA.

In a callable subroutine, a special purpose version of a DDA could provide an accurate approximation to the sine function (Wait, 1963 and Peterson, 1961). In this case, it would require "n" clock pulses to slew full scale. With 30 MHz logic, this would require 525 microseconds with a .001 per cent of full scale accuracy on 20 bit word lengths. The input and retrieval cycle times require "2n" clock pulses or 2 microseconds for 30 MHz logic.

CHAPTER V

TABLE LOOK-UP

The table look-up computational method is essentially a two step process. The computer retrieves previously stored data from memory and executes an interpolative routine to acquire a "best fit" solution.

Several methods are employed for extracting previously stored data. All require a finite number of memory locations and decision handling hardware. The fastest procedure, similar to successive approximations, is determining where in the table the function value of the independent variable is stored and iteratively halving the total until the smallest increment is left, Ledley (1960). The computer then executes an appropriate numerical interpolative routine on functional values adjacent to the given point.

Sine and Cosine Tables

The method described assumes that $0 \leq \theta \leq \frac{\pi}{2}$. The first quadrant is divided into n segments. N values of θ_n, sin θ_n and cos θ_n are stored in memory. The computer calculates the location of the lower adjacent value of θ_n to θ, namely θ_α per Fig. 3. Then the sin θ_α and cos θ_α are used in a linear interpolation routine to calculate sin θ when sin θ is not a previously stored value in memory.

One simple interpolative routine would be as follows. The computer executes the following equations:

$$\theta - \theta_\alpha = K_1$$

$$K_1 \cos \theta_\alpha = K_2$$

$$K_2 + \sin \theta_\alpha = \sin \theta$$

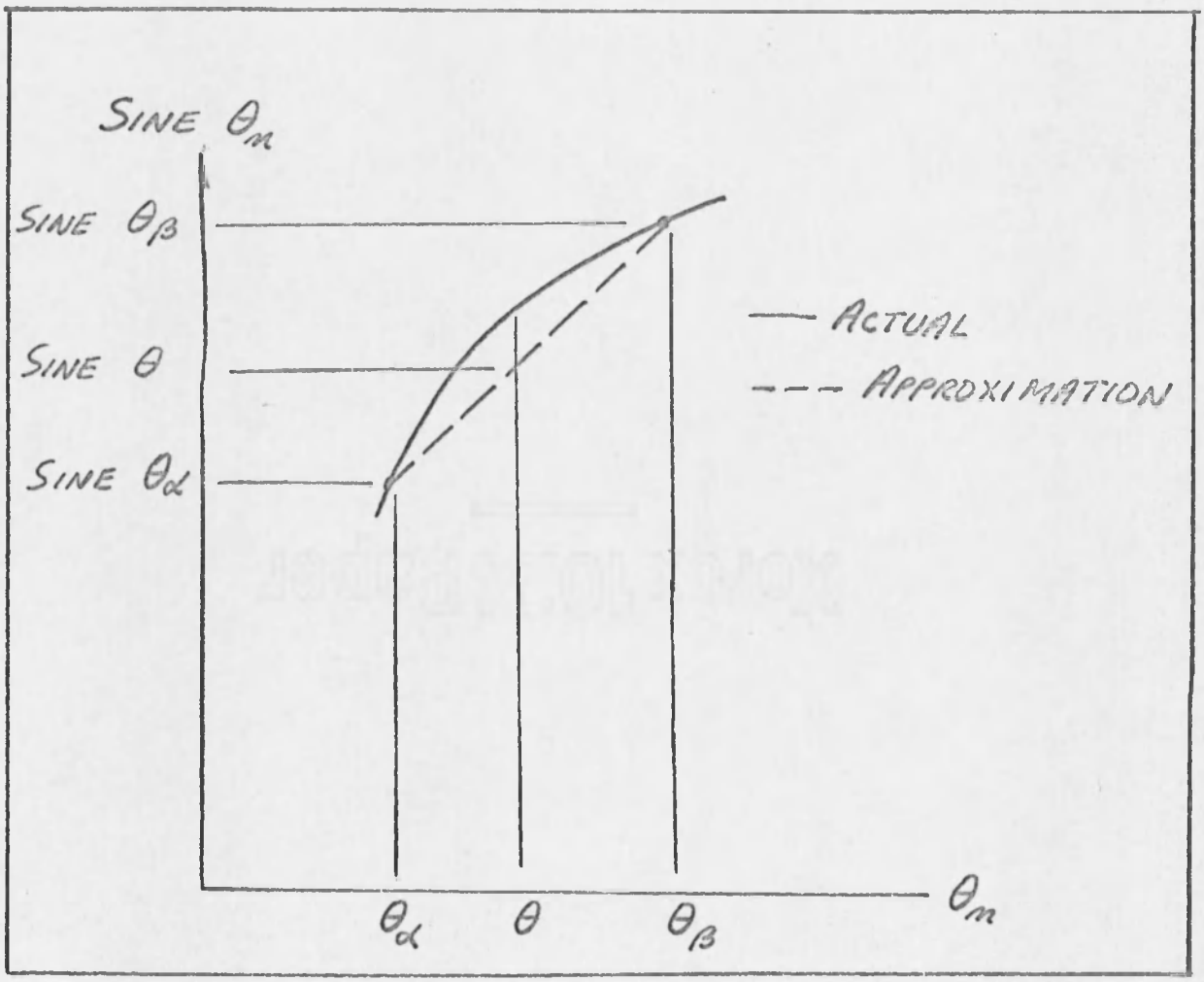


Fig. 3. Graphical Interpolation - Table Look-Up

Error Analysis

Speed and accuracy again conflict in this method. Speed is essentially dependent on time to locate the two adjacent arguments to θ and execute the interpolation routine for values of sin θ not stored in memory. Table 3 lists the results of the previous table look-up scheme for N = 16. The straight line segment breakpoints were chosen to minimize the maximum error. The maximum error occurs at $\theta \doteq 30^{\circ}$ and is .0018. The program requires eight additions, one division, one multiplication and four data retrieval cycles from memory. The PDP-9 computer equivalent execution time is 70 usec. The overhead necessary for this program is used for a quadrant decision sequence, thus an additional 20 microseconds is required.

It is worthwhile to note that utilization of indirect address and autoindex schemes available on the PDP-9 computer can provide greater precision. Private communication with Professor Korn disclosed that such a program would require 64 memory locations to attain .1 per cent accuracy. The execution time for this program would be between 40 to 60 microseconds.

Table 3

Error Generation
Table Look-Up (N=16)

Argument θ	Fractional Error *ΔE
4	.0016
11	.0018
18	.0018
25	.0018
32	.0018
39	.0012
44	.0014
50	.0014
56	.0014
62	.0013
68	.0014
73	.0006
77	.0006
81	.0006
85	.0006
88.5	.0006

$$*\Delta E = \frac{(\text{Sin}\,\theta_{\text{actual}} - \text{Sin}\,\theta_{\text{approximate}})}{\text{Sin}\,\theta_{\text{actual}}}$$

CHAPTER VI

CROSS-ADDITION ALGORITHM

The cross-addition algorithm is a computer program based on a paper by J. E. Volder (1959) with certain modifications. This algorithm makes use of the basic multiplication algorithm within the binary computer to calculate the sine and cosine of an argument simultaneously. It solves the following equation for X and Y.

$$X = K_1 R \cos \theta$$

$$Y = K_1 R \sin \theta$$

Its chief advantage is real-time digital computation with little loss of accuracy attendant with programs based on real-time execution intervals and fixed-point computations.

Theory of Operation

A discussion of the cross-addition algorithm used to compute the sine and cosine of a number is enlightened by the following analogy. Consider terms

$R \sin \theta$ and $R \cos \theta$ of a number θ_A to be rectangular components of a vector in a plane, i.e., given the vector $R \angle \theta$

$$R \sin \theta_A = Y$$

$$R \cos \theta_A = X$$

For simplicity without any loss of generality, let $R = 1$. Then

$$Y = \sin \theta_A$$

$$X = \cos \theta_A$$

Let A represent that vector whose X and Y components satisfy the required relationship for this particular value of θ_A. Of course with each new value of θ_A the X and Y components will change as will the position of vector A. The vector A will represent the particular values of $\sin \theta_A$ and $\cos \theta_A$ which are to be computed. There are no restrictions on the

position or length of A. θ_A satisfies the following relationship.

$$0 \leq |\theta_A| \leq \pi$$

However, once θ_A is given the position of vector, A and its length are fixed.

Now consider the following vector B. Its length is arbitrary also, but for simplicity let its length equal that of A. However, its initial starting position will always be at θ equal to zero.

If some means were available to rotate vector B and control its rotation so that after a certain number of rotations it eventually moves to A's position, then in a sense the change in vector B's position will represent the X and Y components of vector A. Thus, the final position of vector B will represent R sin θ_A and R cos θ_A. The controlling factor remains to be the difference of the two arguments of vector A and vector B. When the sum

$$\arg A + \arg B$$

is zero, vector B represents vector A. The

cross-addition algorithm performs the required sequence of pseudo-rotations on vector B until the (arg A + arg B) = 0.

The particular sequence of pseudo-rotation depends on the direction of the angle θ_A and the manner in which binary multiplication is performed within a binary computer. The least time consuming procedure requires the sum (arg A + arg B) to be smaller than the previous sum for each and every rotation step of vector B towards vector A. The computer performs this sequence of steps by monitoring this sum. Thus the magnitude of each angular increment of rotation decreases sequentially.

Now consider a typical rotation sequence to perform the necessary computation. Vector B is set on the abscissa. $\theta_B = 0$ where θ_B represents the argument of vector B. θ_{B_i} is that argument of vector B when B is in the ith rotated position. The computer has three permanent storage registers and three temporary storage registers. The permanent storage registers contain the initial values of vector B's components during execution and the value of θ_A the argument of vector A. The temporary storage registers

contain the X and Y components of the rotating vector B as vector B's angular position is incremented towards A and the sum (arg A + arg B).

Let X_1 and Y_1 represent the initial X and Y components of vector B

$$X_1 = 1$$

$$Y_1 = 0$$

and the first angular increment of rotation for vector B be 90 degrees.

$$X_2 = \mp Y_1 = R_1(\cos \pm 90^\circ)$$

$$Y_2 = \pm X_1 = R_1(\sin \pm 90^\circ)$$

Thus $X_2 = 0$ and $Y_2 = X_1$, if vector B rotated 90 degrees. $X_2 = 0$ and $Y_2 = - X_1$ if vector B rotated - 90 degrees. The direction of initial rotation was such as to reduce the absolute value of (arg A + arg B). Thus if θ_A were greater than zero, vector B would rotate clockwise or - 90 degrees and if θ_A were less than zero, vector B would rotate counterclockwise or

+ 90 degrees. This first rotation step is unique as will be explained shortly. The next rotation step for vector B is 45 degrees. Its direction depends on the sign of the remainder in the temporary angle register. At this point, however, the temporary X and Y registers contain 0 and X_1 respectively. The temporary angle register contains ($\theta_A \pm 90°$).

The next rotation step and all subsequent steps are characterized in Fig. 4. B_i represents the position of vector B after the ith angular increment of rotation has been performed. Vector B_i will rotate clockwise or counterclockwise to effect a decrease in the absolute value of the sum of the temporary angle register. Thus the computer examines the sum in the temporary angle register after the ith angular increment has been either added or subtracted. If the sum was positive, then vector B rotated less than required. If the sum was negative, then vector B rotated past vector A or overshot vector A. This assumes that the initial value of θ_A was greater than zero. If this were not the case, then a positive sum indicates overshoot while a negative remainder indicates undershoot.

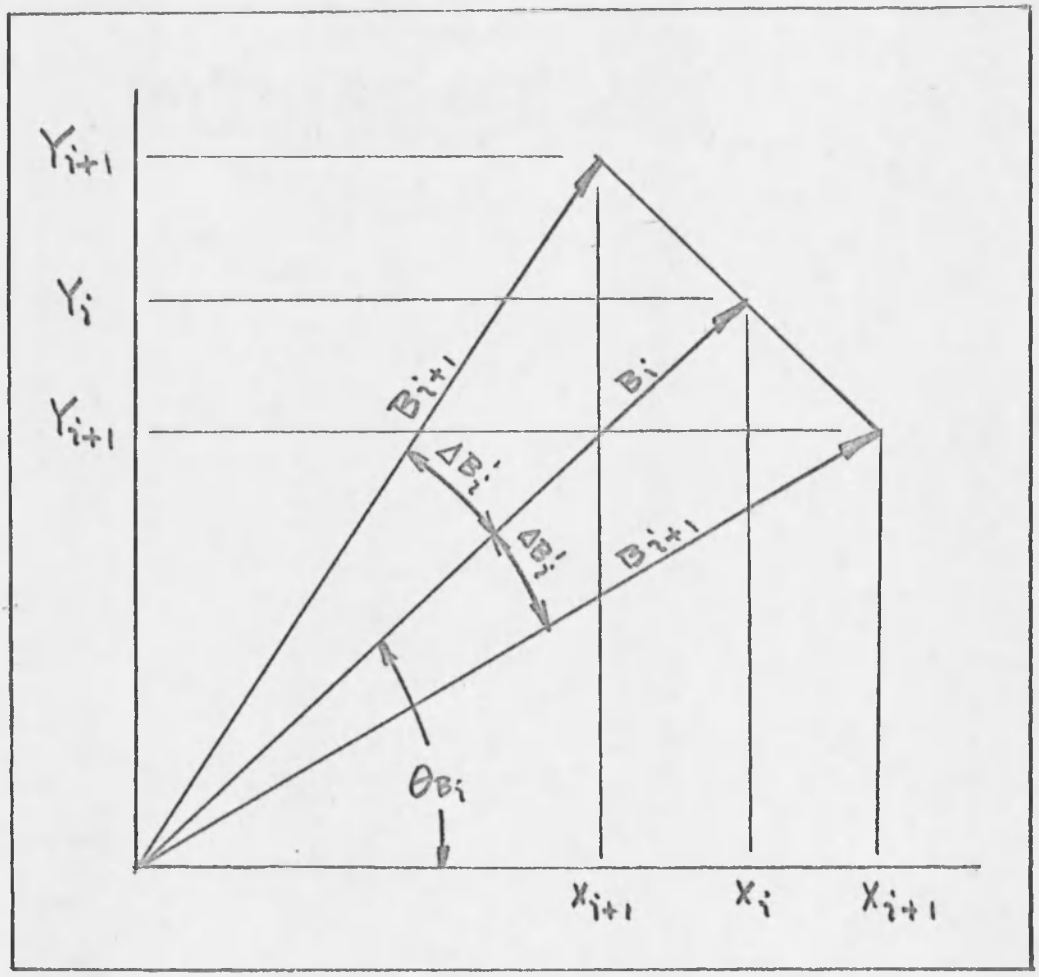


Fig. 4. Typical Angle Increment Step in the Cross-Addition Algorithm

Each value of argument B_i in Fig. 4 has a particular value in the rotation sequence governed by the following relationship.

$$\Delta B_i^1 = \tan^{-1} 2^{-(i-2)} \quad \text{for } i > 1$$

This relationship allows the next rotation step and subsequently the change in the contents of the temporary X and Y registers which represent the X_i and Y_i components of vector B at the ith rotation step to be accomplished by simply shifting and adding or subtracting the previous values of the X_i and Y_i components of vector B at the i-1 rotation step. In general

$$Y_{i+1} = \left|B_i\right| \sin(\theta_i \pm B_i^1)$$

$$= \left|B_i\right| \left[(\sin\theta_i)\cos(\pm B_i^1) \pm \sin(\pm B_i^1)(\cos\theta_i)\right]$$

$$Y_{i+1} = \left|B_i\right| \left[(\sin\theta_i)\cos(\pm\tan^{-1} 2^{-(i-2)}) \pm \sin(\pm\tan^{-1} 2^{-(i-2)})\cos(B_i^1) \right]$$

$$= \left|B_i\right| \left[(\sin\theta_i)\left(\frac{\left|B_i\right|}{\sqrt{1 + 2^{-2(i-2)}\left|B_i\right|}}\right) \pm (\cos\theta_i)\left(\frac{2^{-(i-2)}\left|B_i\right|}{\sqrt{1 + 2^{-2(i-2)}\left|B_i\right|}}\right) \right]$$

$$\text{Let } K_i = \sqrt{1 + 2^{-2(i-2)}}$$

$$\text{Since } \sin \theta_i = \frac{Y_i}{|B_i|}$$

$$\cos \theta_i = \frac{X_i}{|B_i|}$$

$$K_i Y_{i+1} = K_i \left[|B_i| \frac{Y_i}{|B_i|} \frac{|B_i|}{\sqrt{1 + 2^{-2(i-2)}}|B_i|} \pm \right.$$

$$\left. |B_i| \frac{X_i}{|B_i|} \frac{2^{-(i-2)}|B_i|}{\sqrt{1 + 2^{-2(i-2)}}|B_i|} \right]$$

$$= Y_i \pm 2^{-(i-2)} X_i$$

Similarly for X_{i+1}

$$X_{i+1} = |B_i| \cos(\theta_i \pm B_i^1)$$

$$= |B_i| \left[(\cos \theta_i) \cos(\pm B_i^1) \mp (\sin \theta_i) \sin(\pm B_i^1) \right]$$

$$= |B_i| \left[(\cos \theta_i) \cos(\pm \tan^{-1} 2^{-(i-2)}) \mp \right.$$

$$\left. (\sin \theta_i) \sin(\pm \tan^{-1} 2^{-(i-2)}) \right]$$

$$= |B_i| \left[\cos \theta_i \left(\frac{|B_i|}{\sqrt{1 + 2^{-2(i-2)} |B_i|}} \right) \mp \right.$$

$$\left. \sin \theta_i \left(\frac{|B_i|}{\sqrt{1 + 2^{-2(i-2)} |B_i|}} \right) \right]$$

$$K_i X_{i+1} = K_i |B_i| \left[\frac{X_i}{|B_i|} \frac{|B_i|}{\sqrt{1 + 2^{-2(i-2)}|B_i|}} \mp \right.$$

$$\left. \frac{Y_i}{|B_i|} \frac{2^{-(i-2)}|B_i|}{\sqrt{1 + 2^{-2(i-2)}|B_i|}} \right]$$

$$= X_i \mp 2^{-(i-2)} Y_i$$

Close examination of the X_{i+1} and Y_{i+1} components of vector B rotated to the i+1 angular increment indicates that the computer performs two simultaneous shift and add operations with the temporary X and Y storage registers. In a binary computer the relationship $2^{-(i-2)}X_i$ is implemented by merely shifting the contents of the X_i register i-2 places to the right. This is the basis for the name - cross-addition algorithm.

Associated with each rotation step is a small vector growth. However this vector growth is independent of direction of vector rotation. Thus this growth can be considered as a fixed constant for any particular sequence of pseudo-rotations required to reduce the quantity (arg A - arg B) to zero. The number of pseudo-rotation steps is n-2 where n is the number of bits in the binary number. The total vector growth is

$$\Delta K_n = (\sqrt{1 + 2^{-0}})(\sqrt{1 + 2^{-2}}) \ldots (\sqrt{1 + 2^{-2(n-2)}})$$

The complete rotation sequence of vector B towards vector A results in the sum (arg A + arg B) = 0 and the following quantities in the X and Y temporary storage registers.

$$Y_{n+1} = (\sqrt{1 + 2^{-0}})(\sqrt{1 + 2^{-2}}) \ldots (\sqrt{1 + 2^{-2(n-2)}})R_1$$

$$\sin(\pm 90^\circ \pm 45^\circ \pm 26^\circ \pm \ldots)$$

$$X_{n+1} = (\sqrt{1 + 2^{-0}})(\sqrt{1 + 2^{-2}}) \ldots (\sqrt{1 + 2^{-2(n-2)}})R_1$$

$$\cos(\pm 90^\circ \pm 45^\circ \pm 26^\circ \pm \ldots)$$

where $(\pm 90^\circ \pm 45^\circ \pm 26^\circ \pm \ldots) = \theta_1$

If the vector growth is compensated by initially setting $\frac{1}{K_n}$ in the X register instead of 1.000 then the following results occur.

$$Y_{n+1} = R_1 \sin \theta_1$$

$$X_{n+1} = R_1 \cos \theta_1$$

These are the required numbers to be calculated.

Table 4 lists ΔK_n for binary numbers with various bit lengths. The special angular increments required for the cross-addition algorithm are listed in Table 5.

Simulation Program

Appendix A, Part IV lists the PDP-9 computer simulation of the cross-addition algorithm. Nineteen permanent storage registers are required for the special angular increments. In this program, the angle increments start at address location ten. The argument for vector A is deposited in address location THETA. The scaling factor for the argument is

$$1.5707961 \text{ Radians} \longleftrightarrow 65{,}536_{10}$$

$$\longleftrightarrow 200{,}000_{8}$$

The scaling factors for the sine and cosine are

$$\sin 1.5707961 \longleftrightarrow 65{,}536_{10}$$

$$\longleftrightarrow 200{,}000_{8}$$

$$\cos 0.0000000 \Leftrightarrow 65,536_{10}$$

$$\Leftrightarrow 200,000_{8}$$

The PDP-9 computer execution time for this program is approximately 500 usec. The initial value of X is $39,797_{10}$. This number compensates for vector growth which in this case for an 18 bit word is 1.64676. If no vector growth occurred, the initial value of X would be 65,536 to the base ten (200,000 in octal).

Error Analysis

Tables 6 and 7 list the errors generated for the sine and cosine functions with the cross-addition algorithm simulation on the PDP-9 computer. No error was greater than the last two significant bits of the 17 bit word. Specker (1965) has shown that the maximum absolute error is 2^{-R} exp (2^{-R}) where R is the word length. For a 17 bit word this error is less than 5×10^{-12}. The maximum machine error introduced by the simulation program is approximately 6×10^{-5}. One immediately senses that the cause of error is simply the finite word length in the machine.

Table 4

Total Vector Growth Versus Word Length

Binary Bits Number n	Vector Growth ΔK_n
2	1.41421
3	1.58114
4	1.6298
5	1.64248
6	1.64569
7	1.64649
8	1.64669
9	1.64674
10	1.64676
11	1.64676
12	1.64676
13	1.64676
14	1.64676
15	1.64676

Table 5

Pseudo-Rotation Angular Increments

Step i	Angular Increment Magnitude B_i^1		
	Radians	Degrees (Approx.)	
1	1.57079610	90	
2	0.78539805	45	
3	0.463648	26	34'
4	0.244979	14	2'
5	0.124355	7	8'
6	0.0624188	3	34'
7	0.0312399	1	47'
8	0.0156237		53'
9	0.00781230		26'
10	0.00390621		13'
11	0.00195313		7'
12	0.000976563		3'
13	0.000488281		2'
14	0.000244141		1'
15	0.000122070		25"
16	0.0000610352		12"
17	0.0000305176		6"
18	0.0000152588		3"
19	0.0000076294		1"

Table 6

Error Generation of the Sine Function
Using the Cross-Addition Algorithm

Degrees (Approx.)	Radians (Actual)	Sine (Scaled)*		Difference
		(Computer)	(Actual)	
10	0.175	11,410	11,411	+ 1
20	0.349	22,411	22,414	+ 3
30	0.524	32,791	32,790	- 1
40	0.698	42,119	42,118	- 1
50	0.873	50,218	50,217	- 1
60	1.047	56,749	56,750	+ 1
70	1.222	61,588	61,589	+ 1
80	1.396	64,537	64,535	- 2
90	1.571	65,536	65,539	+ 3

*Scaled (65,536)(Sin 90°) = $65,536_{10}$ = $200,000_{8}$

Above numbers are in decimal base.

Table 7

Error Generation of the Cosine Function
Using the Cross-Addition Algorithm

Degrees (Approx.)	Radians (Actual)	Cosine (Scaled)*		Difference
		(Computer)	(Actual)	
10	0.175	64,535	64,531	- 4
20	0.349	61,582	61,585	+ 3
30	0.524	56,743	56,741	- 2
40	0.698	50,209	50,207	- 2
50	0.873	42,110	42,109	- 1
60	1.047	32,779	32,776	- 3
70	1.222	22,398	22,400	+ 2
80	1.396	11,397	11,395	- 2
90	1.571	- 13	- 17	- 4

*Scaled (65,536)(Cos 0°) = $65,536_{10} = 200,000_{8}$

Above numbers are in decimal base.

CHAPTER VII

A HARD-WIRED CROSS-ADDITION DIGITAL RESOLVER

The digital system described herein is a special-purpose on-line 20 bit binary computer with a fixed-program processor. It is designed to solve the fixed trigonometric relationships

$$X = K_1 R \cos(\theta + \theta_1)$$

$$Y = K_1 R \sin(\theta + \theta_1)$$

where $|\theta + \theta_1| \leq 180$ degrees. $R \cos \theta_1$ and $R \sin \theta_1$ are initially deposited in the X and Y registers respectively, while θ is deposited in the accumulator of the angle adder section. The computer generates $K_1 R \cos(\theta + \theta_1)$ and $K_1 R \sin(\theta + \theta_1)$ in the X and Y registers respectively. K_1 is a fixed constant equal to 1.6467 which appears in the final answers. The nature of the computer introduces this factor called vector growth. If θ_1 equals zero, then the initial values of X and Y to be deposited in the

computer are R and zero. Note that R cos θ_1 and R sin θ_1 must be scaled so $K_1R \cos(\theta + \theta_1)$ and $K_1R \sin(\theta + \theta_1)$ satisfy

$$-(2^{n-1} - 1) \leqslant K_1R \cos(\theta + \theta_1) \leqslant 2^{n-1} - 1$$

$$-(2^{n-1} - 1) \leqslant K_1R \sin(\theta + \theta_1) \leqslant 2^{n-1} - 1$$

where n = 20. Overflow indication is furnished for 1's complement addition when the absolute value of algebraic summed result exceeds the capacity of the accumulator ($2^{19} - 1$). The computer incorporates 1's complement fixed-point arithmetic throughout.

Execution time is dictated by the gate-and-add cycle times of the X and Y adder section. The sequence consists of 19 add cycles and 133 gate delays. An add cycle is propagation of the sum of two 20 bit words and the carry or its complement generated from the most significant bit. (All register and accumulator contents can be monitored with indicator lamps.)

Functional Description

The processor consists of logic gates and wire interconnects that control timing pulses from the event counter to other functional units. The internal clock drives an event counter which contains preset delay pulses that control the various functional units. Since the main clock controls the timing sequences, the computer is basically synchronous. The functional units are the X-Y adder, angle adder, event counter, angle increment memory and the clock (Fig. 5).

As shown in the figure the clock is initiated and terminated by load and dump XY signals. The clock serves as the prime driver for the event counter. The event counter generates all gate control pulses for the resolver. The angle adder unit operates on the given argument with stored values in the angle increment memory. Complement signals from the angle adder control internal sequential data transfer within the XY adder to generate the cross-addition algorithm.

Design Details

Real-Time Clock

The clock generates timing pulses to increment the event counter. The pulse frequency is primarily

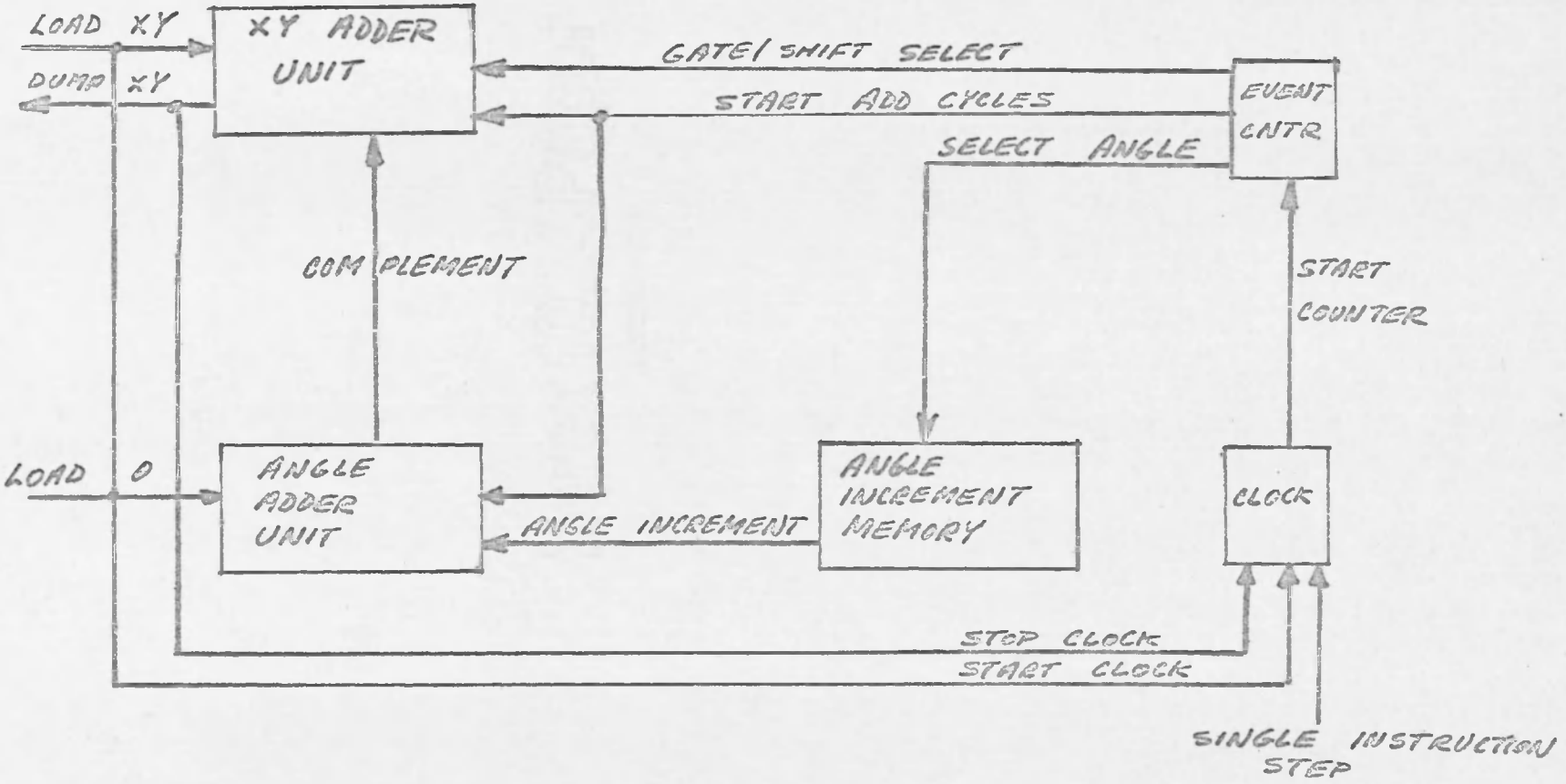
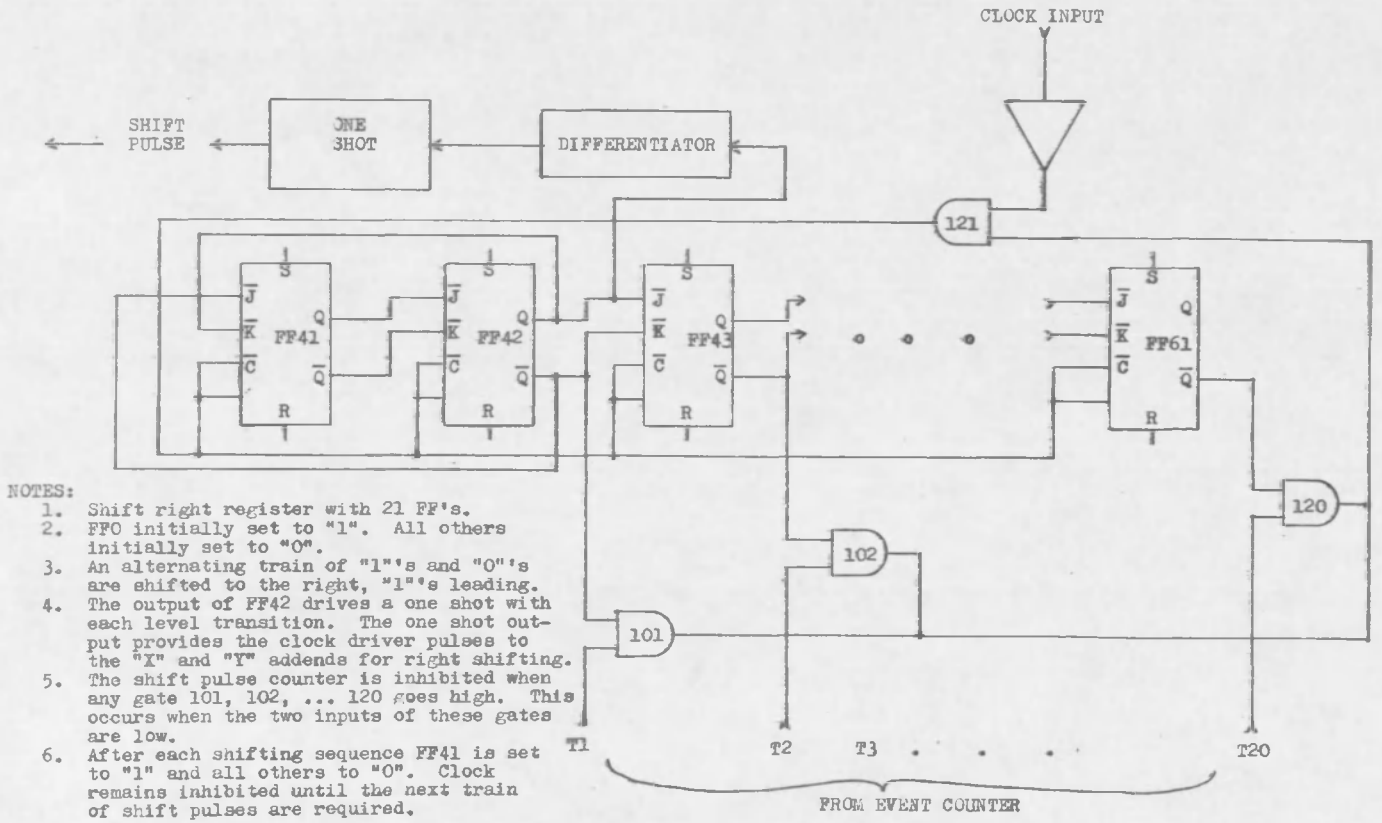


Fig. 5. Cross-Addition Computer System Flow Chart

determined by the minimum gate cycle in the X and Y registers. The clock starts with the reception of Load X, Load Y and Load Θ gate pulses from the main computer. It terminates upon transmission of the Dump X, Dump Y and Dump Θ gate pulses to the main computer. The single instruction step control slows the clock down for manual sequencing of all computer functions.

Event Counter

The event counter sequentially generates the gate control pulses (shift, add and complement commands to the various registers and accumulators). It is composed of four ring counters, logic gates and all pulse amplifiers or gate drivers needed for the system, Fig. 6 (in pocket) and Fig. 7. The event counter is initiated and terminated by the real-time clock. Flip flops 21 through 40 provide ten sequential event pulses. Initially flip flop 21 is set to one and all others set to zero. The real-time clock pulses control signal input C to effect right shifting of the one pulse through the flip flops. At event time 8 the clock pulses to the counter are inhibited for one half of a machine full add cycle. At event time 9 the



NOTES:

1. Shift right register with 21 FF's.
2. FF0 initially set to "1". All others initially set to "0".
3. An alternating train of "1"'s and "0"'s are shifted to the right, "1"'s leading.
4. The output of FF42 drives a one shot with each level transition. The one shot output provides the clock driver pulses to the "X" and "Y" addends for right shifting.
5. The shift pulse counter is inhibited when any gate 101, 102, ... 120 goes high. This occurs when the two inputs of these gates are low.
6. After each shifting sequence FF41 is set to "1" and all others to "0". Clock remains inhibited until the next train of shift pulses are required.

FIG. 7. SHIFT PULSE COUNTER

clock pulse is again inhibited for one half of a machine full add cycle. At event time 10 an end-around shift brings the one pulse back to flip flop 26 and the sequence is repeated. Flip flops 31 through 40 comprise a special counter which stops the master clock after 18 loops from flip flop 6 to flip flop 10 have occurred.

Flip flops zero through 20 sequentially gate the angle increment memory. Initially flip flop zero is set to a one and all others to zero. The one state advances sequentially through the shift register.

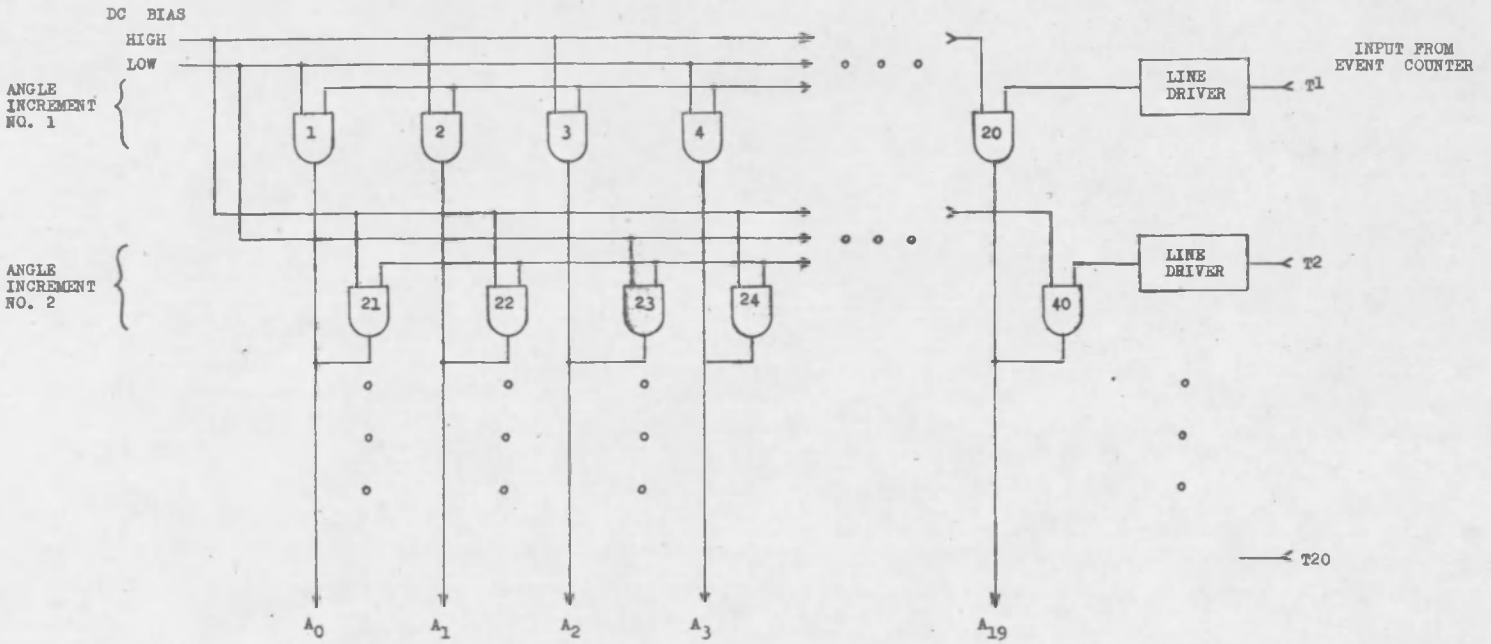
Flip flops 41 through 61 count and generate the required number of shift pulses needed in the XY adder. Initially flip flop 41 is set to a one and all others to zero. Flip flops 41 and 42 are wired to toggle. Thus, as the shift register is clocked, an alternating train of ones and zeroes advances to the right. The changing output of flip flop 42 drives the one-shot. These trigger pulses then clock the addend registers in the XY adder for right shifting. These clock pulses cease when any of the gates 101 through 120 go high thus inhibiting gate 121 and the clock pulses to the XY adder.

Angle Increment Memory

This memory is a 20 word, 20 bit word length storage device. This storage device has a non-destructive read-only capability. (Each word is preset, thus a write capability is not required.) The read cycle access time is comparable to the add cycle of the angle adder. Both the angle increment and its complement are stored in this memory. Each word is addressed sequentially from the event counter and gated to angle buffer register A., Fig. 8.

Angle Adder

This unit consists of the angle adder accumulator, the two angle buffer registers and logic gates for accumulator complementation, Figs. 9 and 10. Addition is performed in 1's complement arithmetic with a double rank adder. The adder section contains a ripple carry adder. The odd numbered gates 1 through 79 complement angle buffer register A. The even numbered gates 2 through 80 load buffer register A. All buffer registers contain 20 JK flip flops. Flip flop 43 when clocked with signal SAI at event time 2 controls complementation in the XY adder unit. Flip flop



OUTPUT TO ADDEND BUFFER REGISTER
IN ANGLE ADDER SECTION

NOTES:
1. All gates are two input, OR/NOR output, without pull down resistors.
2. 20 gates per word.
3. 20 words in memory.
4. Line drivers amplify and invert interrogation signal from event counter.
5. All gates are DC biased high or low as required for a "1" or "0" output.

FIG. 8. ANGLE INCREMENT MEMORY

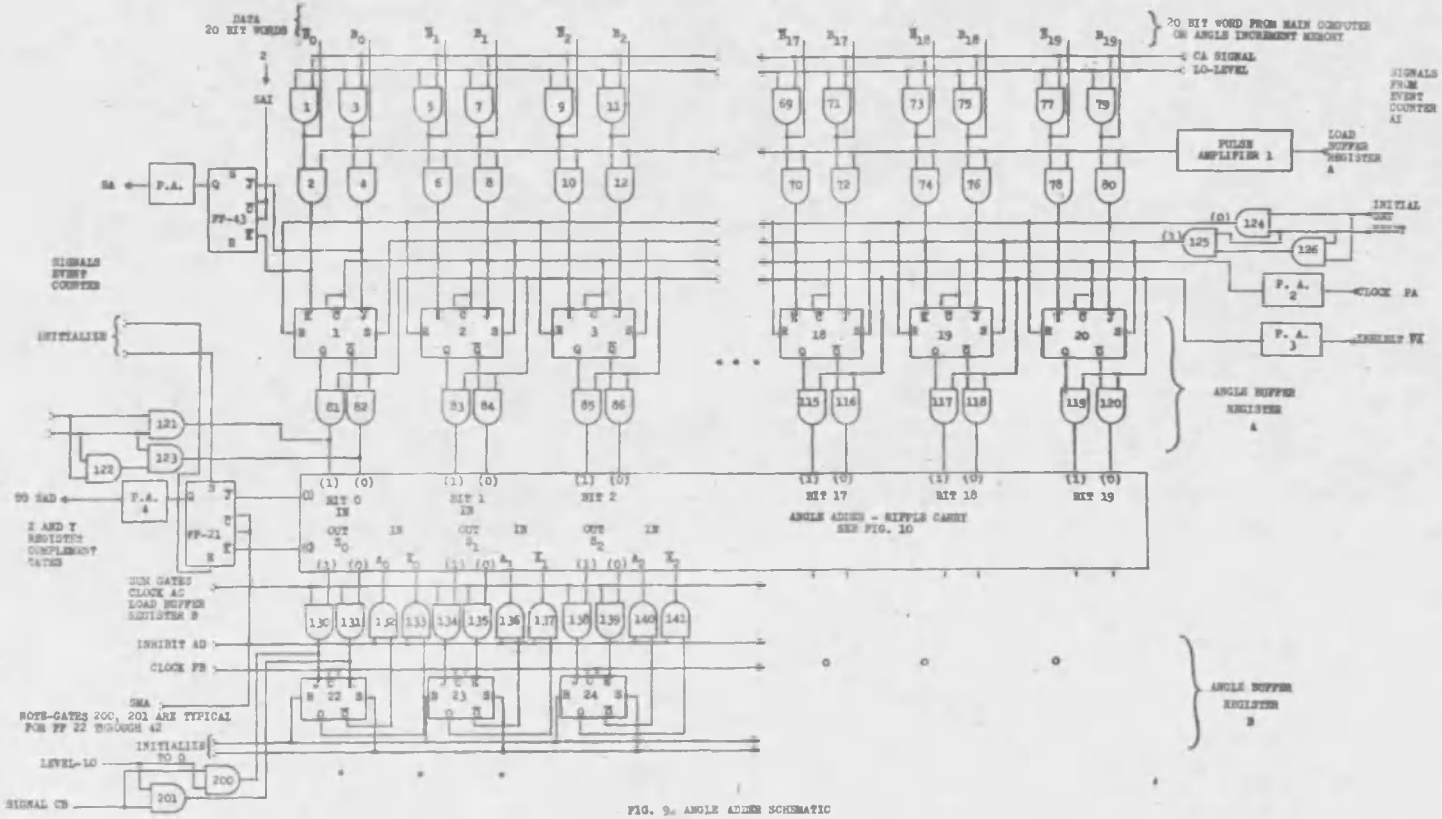
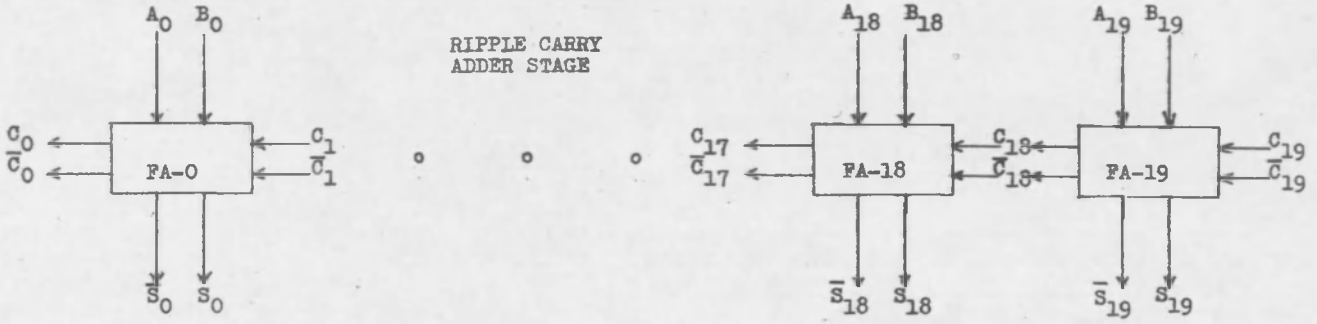


FIG. 9. ANGLE ADDER SCHEMATIC



NOTES:

1. FA-0 is full adder for sign bit.
2. FA-1 through FA-19 are full adders for the 19 most significant bits with FA-1 for the most significant bit.
3. A_0, A_1, A_2 ... are bits from the addend registers.
4. B_0, B_1, B_2 ... are bits from the accumulators.
5. Full adders generate SUM, $\overline{\text{SUM}}$, CARRY and $\overline{\text{CARRY}}$.
6. Full adders require A, B, CARRY IN and $\overline{\text{CARRY IN}}$.

FIG. 10. ADDERS

21 when clocked with signal SMA controls complementation of buffer register A and the X addend and Y addend registers in the XY adder unit.

XY Adder

The XY adder section performs a sequential cross-addition algorithm to calculate R sin θ and R cos θ with two separate adders, the X adder and the Y adder. Level logic is used to minimize timing problems. Figs. 11, 12 and 13 depict the data and signal flow among the gates and registers. Two 18 bit words from the main computer enter the 18 most significant bits of the adders with the Y input and X input gates. The nineteenth and twentieth bits of each 20 bit adder are gated zeroes. Since the cross-addition algorithm requires an initial transfer of X data or its complement to the Y adder and an initial transfer of Y data or its complement to the X adder, the Y input and X input gates are hard-wired to implement this transfer. Thus, the X input gates load the X accumulator and Y addend register. The Y input gates load the Y accumulator and X addend register.

The X adder and Y adder are ripple-carry double-rank adders. These adders are similar to the angle adder previously described. All complementing

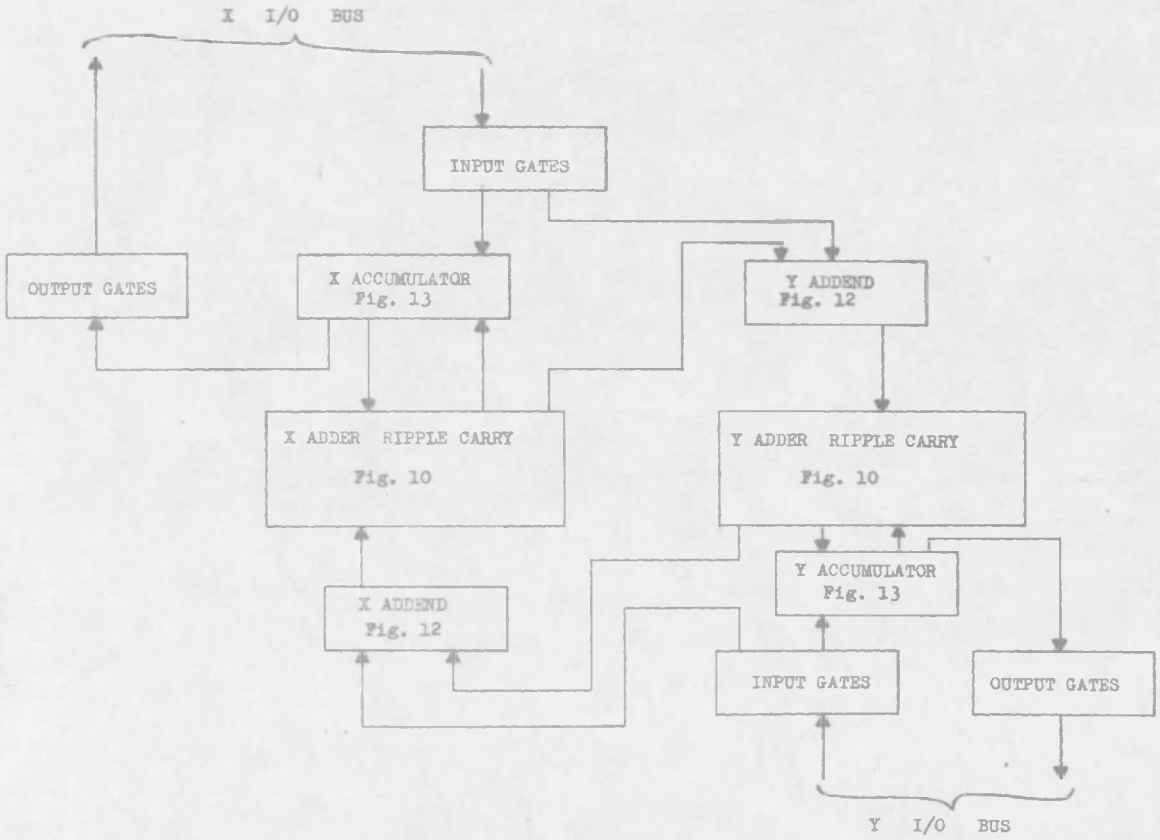
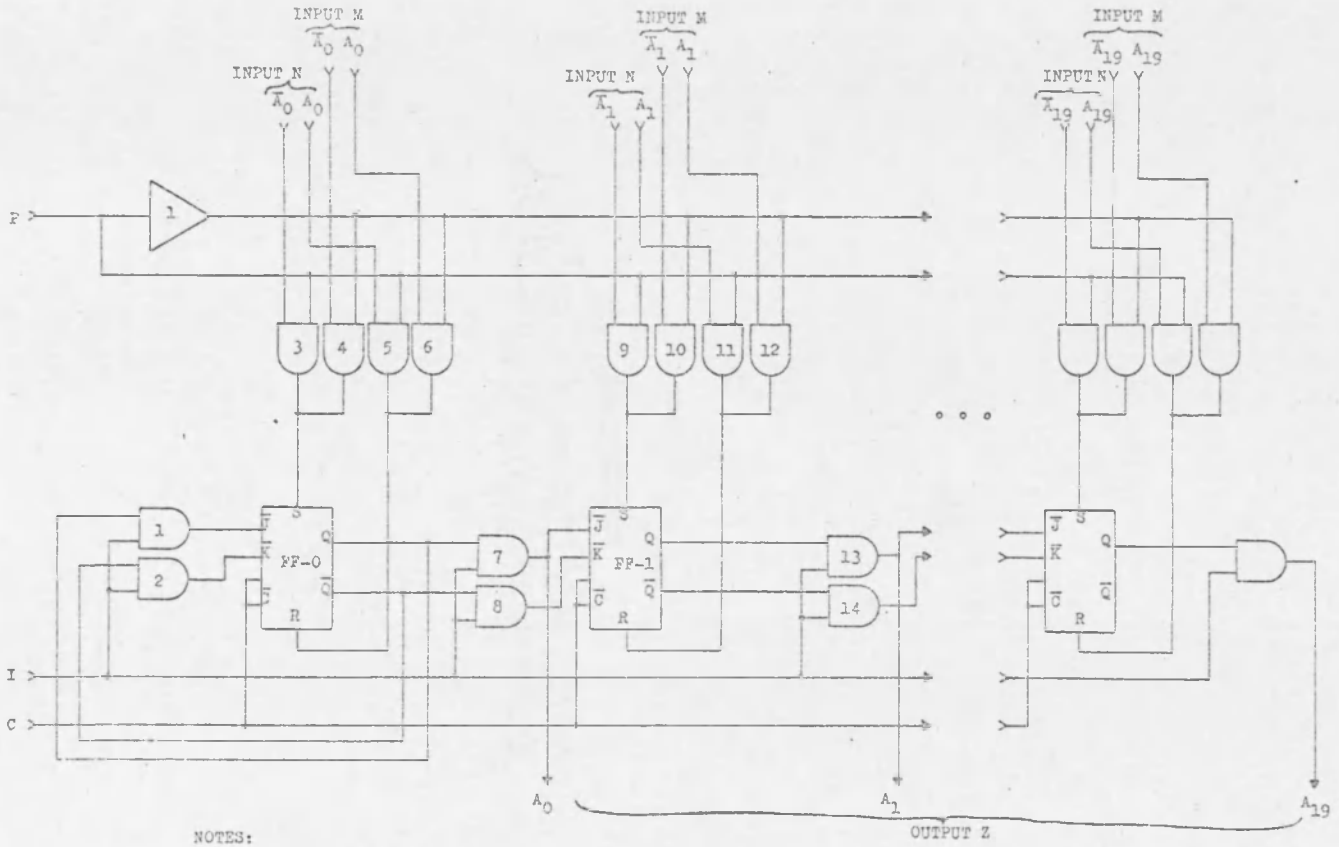


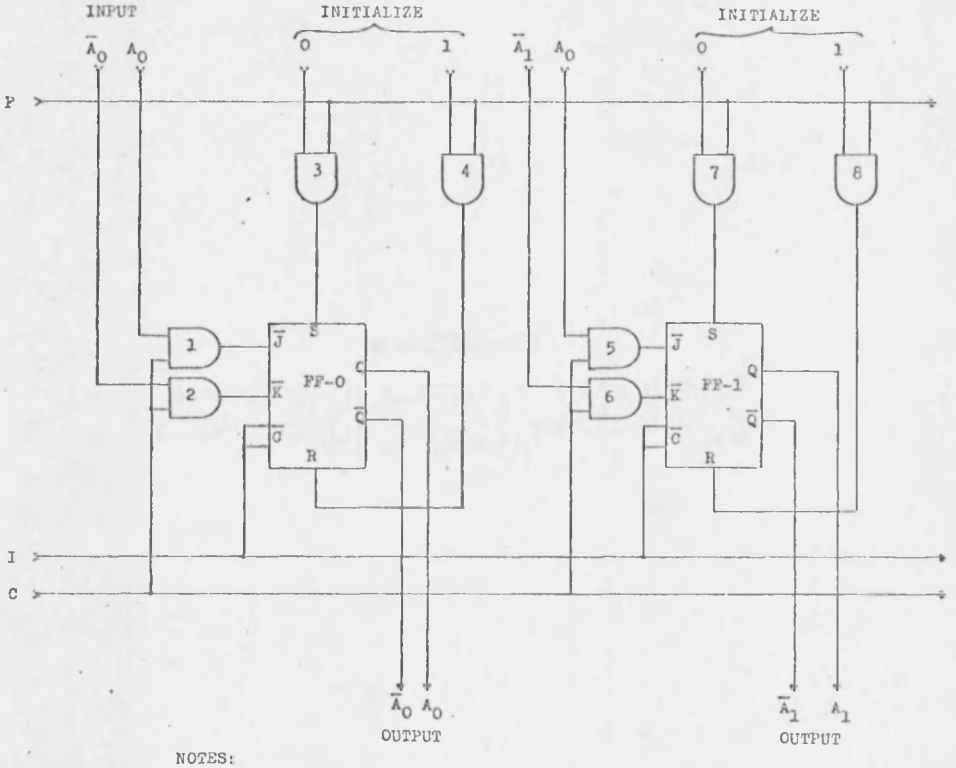
FIG. 11. FLOW DIAGRAM XY ADDER UNIT



NOTES:

1. 20 bit shift right register with complementing capability.
2. A high level on "P" allows data transfer through gates 4 and 6, etc., and inhibits data transfer through gates 3 and 5.
3. Complementing is accomplished with a high level on "I" and the next positive going clock waveform on "C".
4. A shift right is accomplished with a low on "I" and the next positive going clock waveform on "C".
5. Worst case propagate delay includes parallel inputing, complementing, and a shift right. This time delay is approximately 3 clock periods.

FIG. 12. ADDEND BUFFER REGISTER



NOTES:

1. 20 bit buffer register with complementation.
2. NOR gates 3, 4, 7, 8, etc., used for initial set of FF's. A high level on "P" sets all flip flops.
3. NOR gates 1, 2, 5, 6, etc., used for complementing. A high level on "I" and the next positive going clock pulse complements each flip flop.
4. Worst case propagate time includes one data input and one complementation cycle. Two gate delays and two clock cycles.

FIG. 13. ACCUMULATOR REGISTER

is accomplished in the registers. Most gates other than those within the X and Y adders are two input gates to minimize gate delays. Timing is controlled from the event counter.

Once the addend and augend are gated to the adders, addition, transfer/shift and gating out of the adders is automatic. Some logical delays are required for synchronization. Complementation of the X addend register and Y addend register is then performed when the angle adder sign check pulse is received simultaneously with the event counter pulse. The next addition cycle starts with the next event time pulse.

System Operating Sequence

The following timing schedule lists the necessary sequential gating for synchronous computer operation. At each event time, the event counter generates a pulse which is distributed as shown on Fig. 14 (in pocket).

Event Time	Function
0	Main computer starts real-time clock. (one gate delay)

1 Main computer loads argument and initial values of X and Y in angle addend buffer register, X accumulator and Y addend register and Y accumulator and X addend register respectively.
(one gate delay)

2 Algebraic sign of argument checked and gated as signal SA to control possible complement of angle, X and Y accumulators by examining FFO in the angle addend buffer register A.
(two gate delays)

3 Complementing is effected where required.
(two gate delays)

4 Addend and augend in angle adder unit are gated to the adder. Addition is performed.
(one gate delay)
(one add cycle delay)

5 Next angle increment gated from angle increment memory to angle addend buffer register A. Angle adder sign checked.
(one gate delay)

6 Angle adder sign change effects complementation of angle addend buffer register, X addend and Y addend registers.
(two gate delays)

All adders again perform addition.
(one add cycle delay)
(one shift cycle)

7 All addends and augends reloaded.
(one gate delay)

8 Repeat event 5.

9 Repeat event 6.

10 Repeat event 7.

. .

. .

. .

23 Gate sine and cosine to main computer.

24 Reset all registers and counters.

The entire computer sequence requires 133 gate delays and 19 full add cycles. The use of the ripple carry scheme in all adders requires at most 20 gate delays. Thus the full execution time is approximately 500 gate delays. Based on the PDP-9 computer simulation of the cross-addition algorithm, the combined truncation and machine round-off errors should render inaccuracies in at most the nineteenth and twentieth least significant bits.

Table 8 lists some typical execution times for the hard-wired Cross-Addition Digital Resolver. Hardware speeds have been derated by the required DC and AC output loading factors on the NOR gates and flip flops. Worst case execution time is based on the longest time to compute the sine and cosine. A complementation after each pseudo-rotation is one such sequence. TTL hardware speed is based on suggested operating times for SUHL modules manufactured by Sylvania Corporation. All remaining hardware speeds are based on Motorola Corperation specifications.

Table 8

A Comparison of Cross-Addition Digital Resolver Execution Times With Various Hardware

Logic Type	Typical Worst Case Execution Time (usec)
MECL II	5
TTL	12
MECL I	15
MDTL	60
MRTL	100

CHAPTER VIII

CONCLUSIONS

Machine-programmed subroutines for the sine and cosine functions require a floating-point capability either through floating-point hardware or a library subroutine in a fixed-point digital computer. Those without floating-point hardware then require considerable execution time essentially to imitate floating-point manipulations in order to retain the full accuracy of the machine. The cross-addition algorithm would not be implemented by a computer subroutine but rather by a special peripheral which marries the multiplication algorithm to an incremental rotation scheme to calculate values of R sin Θ and R cos Θ. Thus, the execution time is shortened considerably and is essentially dependent on hardware speed. High speed gates with propagation delays of 10 nanoseconds allow the Cross-Addition Digital Resolver to calculate the sine and cosine with 20 binary bit word lengths in less than 15 usecs with an 18 bit accuracy. Execution times for library

subroutines with floating-point capability require at least 18 milliseconds for an 18 bit accuracy for both sine and cosine. Execution times for subroutines without floating-point capability require equal time; however, machine-introduced errors deny the full accuracy available with subroutines with floating-point hardware. Table look-up software subroutines are somewhat faster, but also fail to provide the speed and accuracy demands of real-time computation. The DDA methods with execution times of 10 to 20 microseconds provide a .001 per cent of full scale accuracy.

Table 9 lists the execution time for the methods of calculating the sine and cosine analyzed in this paper. It is assumed that the argument is 20 bit word length in fixed-point format before and after execution. Subsequently, if floating-point format is required, these methods, except where noted, inherently require an additional 444 microseconds overhead time in format conversion sequences (DEC Science Library, 1967). All methods except the Rational Function and Digital Resolver require a quadrant search sequence for the argument. As shown in Chapter II, this requires an additional 20

Table 9

Execution Time and Accuracy
For Several Sine Calculation Methods

Method	Execution Time (usec) T	Accuracy (Binary Bits) N
Three Term Taylor Series	100	10
Three Term Chebyshev Series	100	9
Rational* Function	9400	20
DDA (30 MHz)	525	17
Table Look-Up	70	11
Cross-Addition Digital Resolver (30 MHz)	10	19

*Floating point to fixed-point conversion subroutines not required.

microseconds for the PDP-9 computer. A basis for comparison would be the product of bit accuracy and execution time, "F = TN". Such a "figure of merit" for the Digital Resolver exceeds all others shown by a factor of four.

The Cross-Addition Digital Resolver is well-suited for coordinate conversion problems. Its high-speed coupled with full utilization of the digital computer accuracies attendant with floating point hardware offer the Resolver as a desirable adjunct to real-time computation.

APPENDIX A

PDP-9 PROGRAMS
SINE FUNCTION

I. Taylor Series Approximation (Three Term) for $0 \leqslant X \leqslant .5$

```
LAC X
DAC .+3          LAC C          MUL          0
DAC .+3          LAC X          MUL          0
DAC .+3          LAC X          MUL          0
LRS 3            CMA            DAC XT
LAC X            ADD XT
DAC .+3          LAC X          MUL          0
DAC .+3          LAC B          MUL          0
CMA              DAC XT
LAC A            ADD XT
DAC .+3          LAC X          MUL          0
```

Decimal

C, 050000

B, 016667

A, 100000

Octal

PDP-9 PROGRAMS
SINE FUNCTION

II. Taylor Series Approximation (Three Term)
for $.5 \leq X \leq 1.0$

```
LAC X
DAC .+3          LAC X          MUL          0
LLS 1
DAC .+3          LAC X          MUL          0
LLS 1
DAC .+3          LAC C          MUL          0
LRS 1            CMA            DAC XT
LAC X            ADD XT
DAC .+3          LAC X          MUL          0
LLS 1
DAC .+3          LAC B          MUL          0
CMA              LLS 1          ADD A
DAC .+3          LAC X          MUL          0
```

Decimal

```
C,    500000
B,    016667
A,    100000
```

Octal

PDP-9 PROGRAMS
SINE FUNCTION

III. Chebyshev Series Approximation (Three Term)
for $0 \leqslant X \leqslant 1.0$

```
LAC X
DAC .+3          LAC X          MUL          0
LLS 1
DAC .+3          LAC X          MUL          0
LLS 1
DAC .+3          LAC C          MUL          0
LRS 1            CMA            DAC XT
LAC X            ADD XT
DAC .+3          LAC X          MUL          0
LLS 1
DAC .+3          LAC B          MUL          0
CMA              LLS 1          ADD A
DAC .+3          LAC X          MUL          0
```

Decimal

C, 007853
B, 041111
A, 157079

Octal

PDP-9 PROGRAMS
SINE FUNCTION

IV. Cross-Addition Algorithm
for $0 \leqslant X \leqslant 1.00000$ Radians

```
        LAC THETA     SMA           JMP PLUS      CMA
MINUS,  LAC Y         DAC XT        LAC X         TAD THETA
        TAD (1        DAC YT        LAC I 10
        DAC THETA     JMP LOOP
PLUS,   LAC Y         CMA           TAD (1        DAC XT
        LAC X         DAC YT        LAC I 10      CMA
        TAD (1        TAD THETA     DAC THETA
LOOP,   SMA           JMP SUM
DIFF,   LAC YT        XCT SHIFT     TAD XT        DAC X
        LAC XT        XCT SHIFT     CMA           TAD (1
        TAD YT        DAC YT        LAC X         DAC XT
        LAC I 10      TAD THETA     DAC THETA     JMP INCRNT
SUM,    LAC XT        XCT SHIFT     TAD YT        DAC Y
        LAC YT        XCT SHIFT     CMA           TAD (1
        TAD XT        DAC XT        LAC Y         DAC YT
        LAC I 10      CMA           TAD (1        TAD THETA
        DAC THETA
INCRNT, ISZ SHIFT     LAC SHIFT     SAD (660517
        JMP ANSWER    LAC THETA     JMP LOOP
ANSWER,
```

```
10,     111111      200000      100000
        045620      023755      012104      005054
        002427      001214      000506      000243
        000122      000051      000024      000012
        000005      000003      000001      000000
X,      115563
Y,      000000
```

LIST OF REFERENCES

Booth, A. D. "A Note on Programming Polynomials For Trigonometric Functions," Mathematical Tables and Other Aids to Computation, Vol. 9, pp. 21-23 (January 1955).

Booth, A. D., and K. H. V. Booth. Automatic Digital Calculators. London: Butterworths Ltd., 1953.

Bradley, R. E., and J. F. Genna. Design of a 1 MC Iteration Rate DDA. Indianapolis: Hazeltine Technical Development Center, Inc., 1961.

Braun, E. L. "Design Features of Current Digital Differential Analyzers," Electronic Computers and Information, IRE Convention Record, Part 4, pp. 87-97 (1954).

Bush, Vannevar. "Differential Analyzer," J. Franklin Institute, Vol. 212, No. 4, pp. 447-448 (October 1931).

Bush, Vannevar, and A. H. Caldwell. "A New Type of Differential Analyzer," J. Franklin Institute, Vol. 240, No. 4, pp. 255-326 (October 1945).

Callan, J. D. "MOS Integrated Digital Differential Analyzer," Application Notes. Hicksville, Long Island: General Instrument Corporation, August, 1966.

Campbell, Lewis M., and Howard H. Hicks. "Theory and Application of the Digital Resolver," Noltr 66-232. White Oak, Maryland: United States Naval Ordnance Laboratory, 1967.

Chrystal, G. Algebra, Part II, Seventh Edition. New York: Chelsea Publishing Company, 1964.

Clenshaw, C. W. "A Note on the Summation of Chebyshev Series," Mathematical Tables and Other Aids to Computation, Vol. 9, pp. 118-120 (July 1955).

Clenshaw, C. W. "Polynomial Approximations to Elementary Functions," Mathematical Tables and Other Aids to Computation, Vol. 8, pp. 143-147 (1954).

Connelly, M. E. "Real-Time Analog-Digital Computation," IRE Transactions on Electronic Computers, Vol. EC-11, pp. 31-41 (February 1962).

DEC Science Library. PDP-9 Advanced Software System. Maynard, Massachusetts: Digital Equipment Corporation, 1967.

Dickinson, M. M. "A Comparison of Digital Differential Analyzers and General-Purpose Equipment in Guidance Systems," Computers in Control, AIEE Control Computer Sessions, Publication No. S-132, pp. 208-209 (September 1961).

Elliot, David. "Truncation Errors in Two Chebyshev Series Approximations," Mathematics of Computation, Vol. 19, p. 234 (April 1965).

Elliot, David, and George Szekeres. "Some Estimates of the Coefficients in the Chebyshev Series Expansion of a Function," Mathematics of Computation, Vol. 19, p. 25 (January 1965).

Flores, Ivan. The Logic of Computer Arithmetic. New Jersey: Prentice-Hall, 1963.

Garrett, Lane S. "Mecl 70 MHz J-K Flip Flop," Application Note Integrated Circuits AN-280. Phoenix: Motorola Semiconductor Products, Inc., 1967.

Garrett, Lane S. "The Mecl Line of Digital Integrated Circuits," Application Note Integrated Circuits AN-244. Phoenix: Motorola Semiconductor Products, Inc., 1966.

Gilliland, M. C. "The Iterative Differential Analyzer Function," Instruments and Control Systems, pp. 675-679 (April 1961).

Gilliland, M. C. "The Spectral Evaluation of Iterative Differential Analyzer Integration Techniques," Proceedings of the Western Joint Computer Conference, Vol. 18, pp. 507-518 (May 1961).

Handbook of Mathematical Functions. Washington: National Bureau of Standards Applied Mathematics, Series 55, 1966.

Harris, J. N. "A Programmed Variable-Rate Counter for Generating the Sine Function," IRE Transactions on Electronic Computers, Vol. EC-5, pp. 21-26 (March 1956).

Harris, L. G. Serial Digital Computational Units: Technique Developments. San Diego: United States Navy Electronics Laboratory, November, 1965.

Hastings, Cecil, Jr. Approximation for Digital Computers. Princeton: Princeton University Press, 1955.

Heid, J. P. A Hybrid Polar/Cartesian Coordinate Converter. Horsham, Pennsylvania: Drexel Dynamics Corporation, 1963.

Howell, J. V. Digital Approach to Coordinate Conversion. Los Angeles: Packard Bell Computer Corporation, June, 1960.

Kella, J., and A. Shani. "A Note on the Accuracy of DDA's," IEEE, Vol. EC-16, No. 2, pp. 230-231 (April 1967).

Korn, Granino A., and T. M. Korn. Electronic Analog and Hybrid Computers. New York: McGraw-Hill, 1964.

Lanczos, C. Tables of Chebyshev Polynomials. Washington: National Bureau of Standards Applied Mathematics, Series 9, 1952.

Ledley, Robert S. Digital Computer and Control Engineering. New York: McGraw-Hill, 1960.

McCracken, D. D., and W. S. Dorn. Numerical Methods and Fortran Programming. New York: John Wiley and Sons, Inc., 1964.

Mitchell, J. M., and S. Ruhman. "The Trice - A High Speed Incremental Computer," IRE National Convention Record, Part 4, pp. 206-216 (March 1958).

Nelson, D. J. "DDA Error Analysis Using Sampled Data Techniques," Proceedings of the Western Joint Computer Conference, 1962.

Peterson, G. R. Study of Digital-Differential Analyzer Arithmetic, Ph. D. Dissertation, University of Arizona, 1961.

Scientific Computing Services. Program Library Manual. Hughes Aircraft Company, No. B1.101, pp. 1-3, August, 1966.

Shileiko, A. V. The Digital Differential Analyzer. New York: Macmillan Co., 1964.

Specker, W. H. "A Class of Algorithms for Ln X, Exp. X, Sin X, Cos X, Tan^{-1} X, and Cot^{-1} X," IRE Transactions on Electronic Computers, Vol. EC-14, pp. 85-86 (February 1965).

Truitt, T. D. Introduction to DDA's. Princeton: Electronic Associates, Inc., 1961.

User Handbook PDP-9. Maynard, Massachusetts: Digital Equipment Corporation, 1967.

Vichnevetsky, R. "A Spectral Method of Analyzing the Truncation Error in the Finite Difference Technique," Technical Report of the European Computation Center. Brussels, Belgium: Electronic Associates, Inc., 1961.

Volder, Jack E. "The Cordic Trigonometric Computing Techniques," IRE Transactions on Electronic Computers, Vol. EC-8, pp. 330-334 (1959).

Von Handel, P. "Digital Differential Analyzers," Electronic Computers. Englewood Cliffs, New Jersey: Prentice-Hall, 1961.

Wadel, L. B. "An Electronic Differential Analyzer as a Differential Analyzer," Association for Computing Machinery, pp. 128-136 (July 1954).

Wait, J. V. Hybrid Analog-Digital Differential Analyzer System, Ph. D. Dissertation, University of Arizona, 1963.

Ware, Willis H. Digital Computer Technology and Design, Vol. 1. New York: John Wiley and Sons, Inc., 1963.