United States Patent [19]

Dickinson

[11] **3,927,312**

[45] **Dec. 16, 1975**

[54] **VECTOR ROTATOR**

[75] Inventor: **Warren D. Dickinson,** Huntsville, Ala.

[73] Assignee: **The United States of America as represented by the Secretary of the Army,** Washington, D.C.

[22] Filed: **Oct. 31, 1974**

[21] Appl. No.: **519,735**

[52] **U.S. Cl.** **235/189;** 235/156; 235/186

[51] **Int. Cl.**[2] **G06F 7/38**

[58] **Field of Search** 235/152, 156, 186, 189, 235/164, 197

[56] **References Cited**

UNITED STATES PATENTS

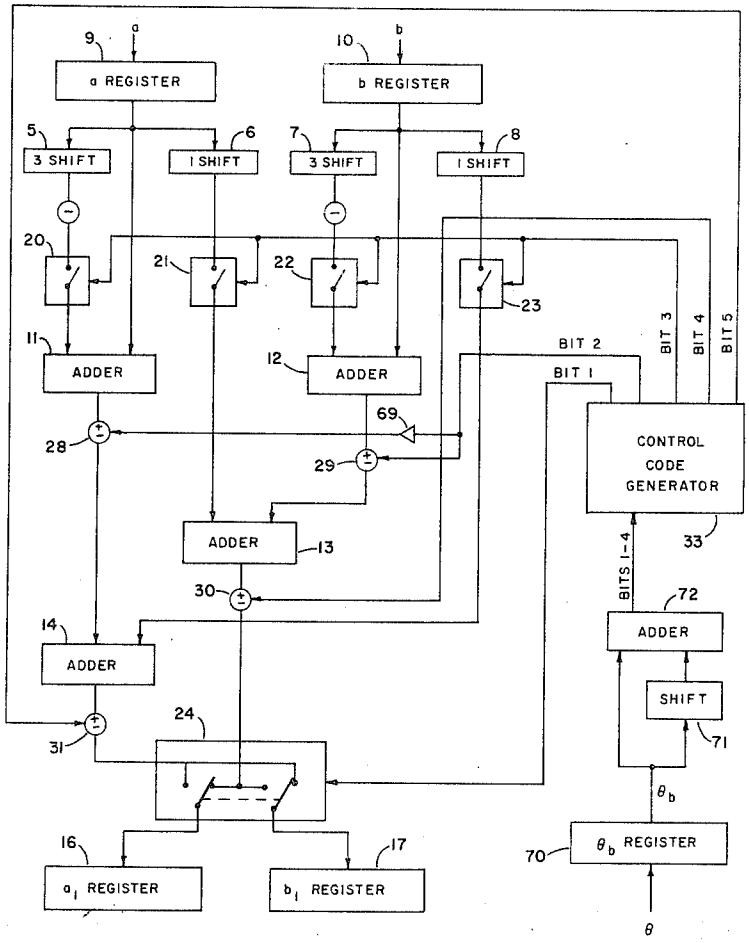
3,684,876	8/1972	Sutherland	235/156 X
3,725,686	4/1973	Ustach	235/156
3,767,905	10/1973	Garde	235/156

Primary Examiner—Joseph F. Ruggiero
Attorney, Agent, or Firm—Lawrence A. Neureither; Joseph H. Beumer; Robert C. Sims

[57] **ABSTRACT**

A two stage approach is used in which rotation angles are specifically selected for easy mechanization. First stage provides rotations in even 30° increments so that multiplications are replaced by simple shift or shift-/add operations. Rotation angles available in the second stage each have sines with at most two nonzero bits in their binary representations.

7 Claims, 6 Drawing Figures



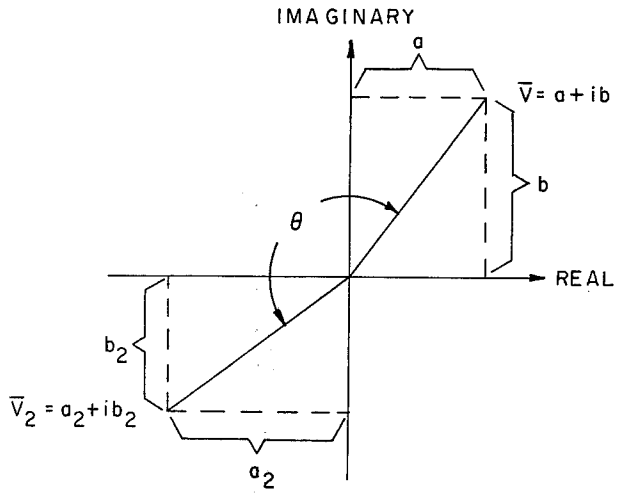

IMAGINARY
a
$\overline{V} = a + ib$
b
θ
REAL
b_2
$\overline{V}_2 = a_2 + ib_2$
a_2

FIG. 1

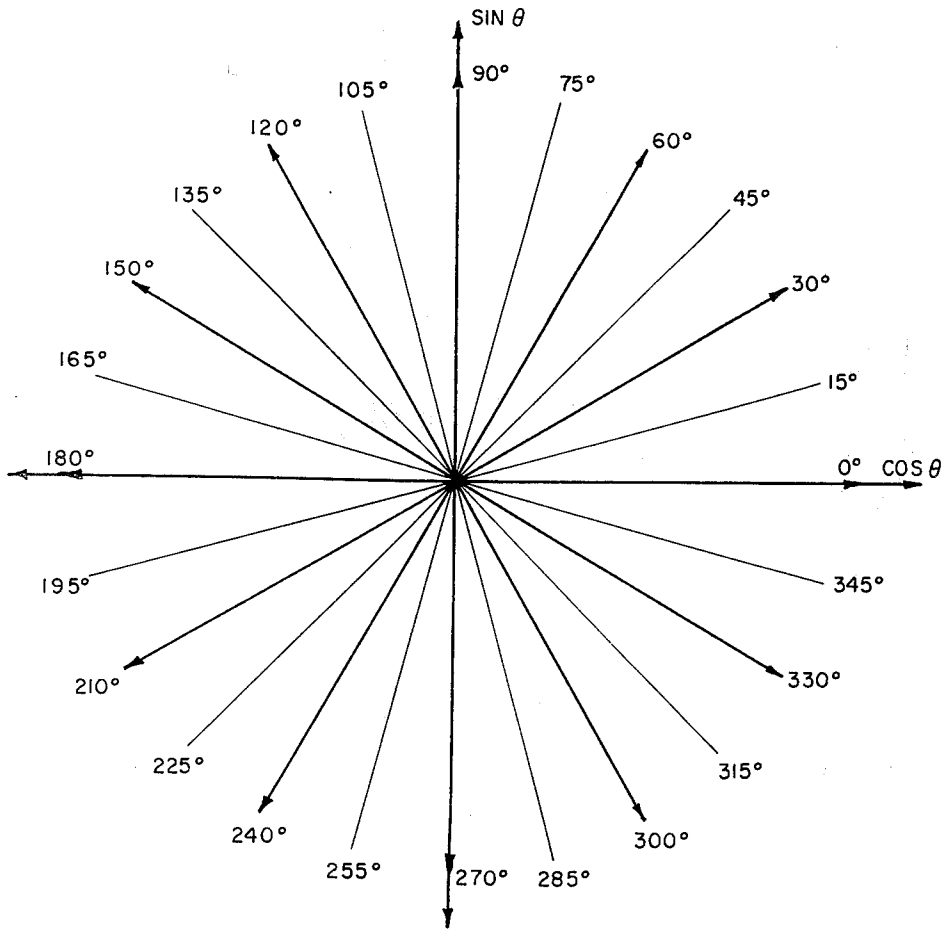

SIN θ
90°
75°
105°
120°
60°
135°
45°
150°
30°
165°
15°
180°
0° COS θ
195°
345°
210°
330°
225°
315°
240°
300°
255°
270°
285°

FIG. 2

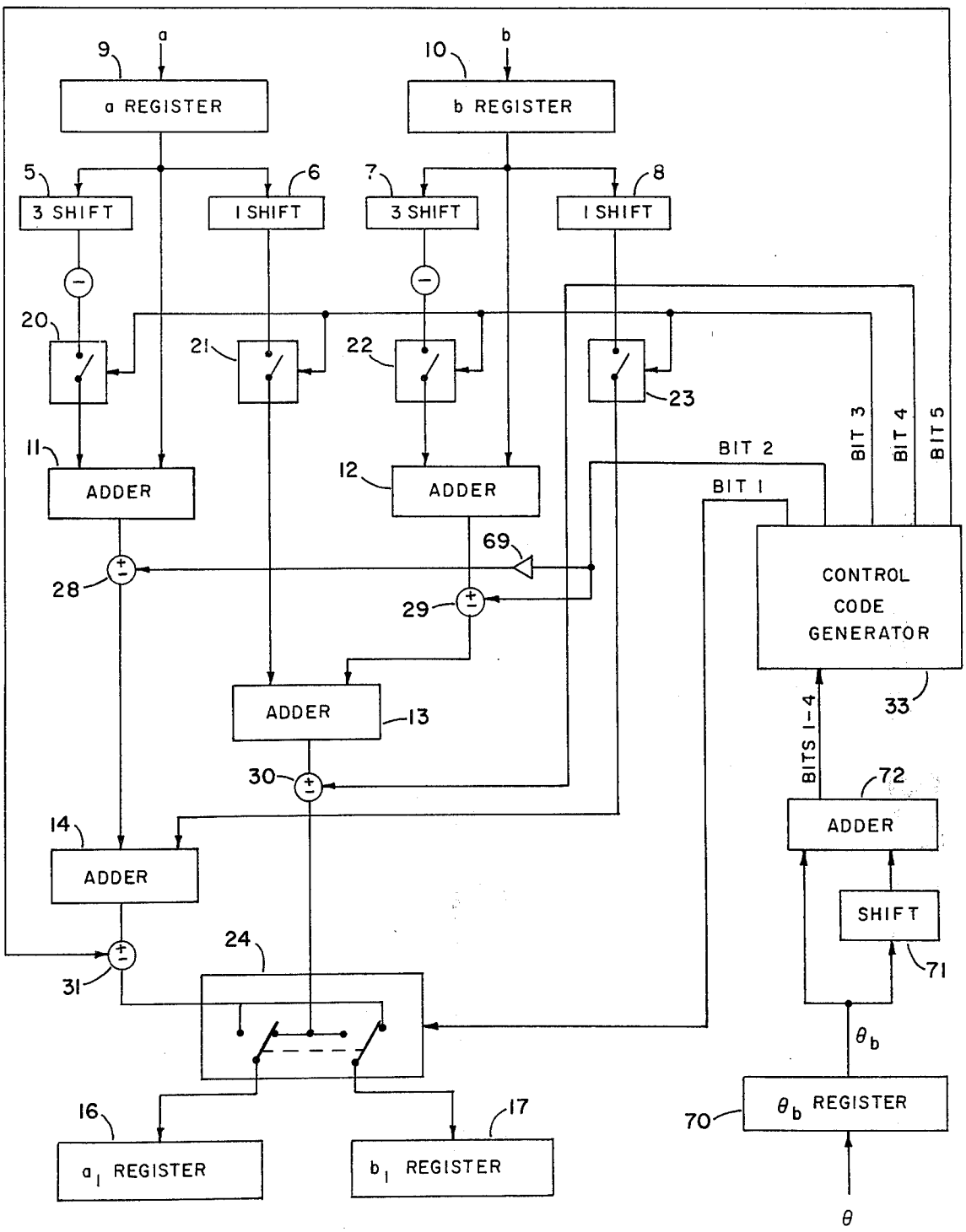

a
b
9
10
a REGISTER
b REGISTER
5
6
7
8
3 SHIFT
1 SHIFT
3 SHIFT
1 SHIFT
20
21
22
23
BIT 3
BIT 4
BIT 5
BIT 2
BIT 1
11
12
ADDER
ADDER
69
28
29
CONTROL
CODE
GENERATOR
ADDER
13
33
30
BITS 1-4
72
14
ADDER
ADDER
SHIFT
24
71
31
θ_b
16
17
70
θ_b REGISTER
a_1 REGISTER
b_1 REGISTER
θ

FIG. 3

1ST. STAGE ROTATION REQUIREMENTS

ROTATION ANGLES θ_1		CORRESPONDING BITS FOR θ_c	REQUIRED OUTPUTS	
+	−	1 2 3 4	a_1	b_1
0°	0°		a	b
30°			$a\ (1-2^{-3})-b2^{-1}$	$a2^{-1}+b\ (1-2^{-3})$
60°			$a2^{-1}-b(1-2^{-3})$	$a(1-2^{-3})+b2^{-1}$
90°			-b	a
120°			$-a2^{-1}-b(1-2^{-3})$	$a(1-2^{-3})-b2^{-1}$
150°			$-a(1-2^{-3})-b2^{-1}$	$a2^{-1}-b(1-2^{-3})$
180°	-180°	1011	-a	-b
210°	-150°	1010 1001	$-a(1-2^{-3})+b2^{-1}$	$-a2^{-1}-b(1-2^{-3})$
240°	-120°	1000 0111	$-a2^{-1}+b(1-2^{-3})$	$-a(1-2^{-3})-b2^{-1}$
270°	-90°	0110 0101	b	-a
300°	-60°	0100 0011	$a2^{-1}+b(1-2^{-3})$	$-a(1-2^{-3})+b2^{-1}$
330°	-30°	0010 0001	$a(1-2^{-3})+b2^{-1}$	$-a2^{-1}+b(1-2^{-3})$
0°	0°	0000	a	b

FIG. 4

2ND. STAGE ROTATION REQUIREMENTS

$\sim\theta_2^\circ$	SIN θ_2	COS θ_2	$\triangle$ COS θ_2	CORRESPONDING θ_c BITS 5-9	REQUIRED OUTPUTS FOR $\pm\theta_2$	
					a_2	b_2
0°	0	1	–	00000	a_1	b_1
0.9°	2^{-6}	1	2^{-13}	00001 00010	$a_1 \mp b_1(2^{-6})$	$b_1 \pm a_1(2^{-6})$
1.8°	2^{-5}	1	2^{-11}	00011 00100	$a_1 \mp b_1(2^{-5})$	$b_1 \pm a_1(2^{-5})$
2.7°	$2^{-5}+2^{-6}$	1	2^{-9}	00101 00110	$a_1 \mp b_1(2^{-5}\ 2^{-6})$	$b_1 \pm a_1(2^{-5}+2^{-6})$
3.6°	2^{-4}	1	2^{-9}	00111 01000	$a_1 \mp b_1(2^{-4})$	$b_1 \pm a_1(2^{-4})$
4.5°	$2^{-4}+2^{-6}$	$1-2^{-7}$	2^{-9}	01001 01010	$a_1(1-2^{-7}) \mp b_1(2^{-4}+2^{-6})$	$b_1(1-2^{-7}) \pm a_1(2^{-4}+2^{-6})$
5.4°	$2^{-4}+2^{-5}$	$1-2^{-7}$	2^{-10}	01011 0110–	$a_1(1-2^{-7}) \mp b_1(2^{-4}+2^{-5})$	$b_1(1-2^{-7}) \pm a_1(2^{-4}+2^{-5})$
7.2°	2^{-3}	$1-2^{-7}$	2^{-15}	0111– 10000	$a_1(1-2^{-7}) \mp b_1(2^{-3})$	$b_1(1-2^{-7}) \pm a_1(2^{-3})$
8.1°	$2^{-3}+2^{-6}$	$1-2^{-7}$	2^{-13}	10001 10010	$a_1(1-2^{-7}) \mp b_1(2^{-3}+2^{-6})$	$b_1(1-2^{-7}) \pm a_1(2^{-3}+2^{-6})$
9.0°	$2^{-3}+2^{-5}$	$1-2^{-6}$	2^{-8}	10011 1010–	$a_1(1-2^{-6}) \mp b_1(2^{-3}+2^{-5})$	$b_1(1-2^{-6}) \pm a_1(2^{-3}+2^{-5})$
10.8°	$2^{-3}+2^{-4}$	$1-2^{-6}$	2^{-9}	1011– 110--	$a_1(1-2^{-6}) \mp b_1(2^{-3}+2^{-4})$	$b_1(1-2^{-6}) \pm a_1(2^{-3}+2^{-4})$
14.5°	2^{-2}	$1-2^{-5}$	2^{-11}	111--	$a_1(1-2^{-5}) \mp b_1(2^{-2})$	$b_1(1-2^{-5}) \pm a_1(2^{-2})$

FIG. 5

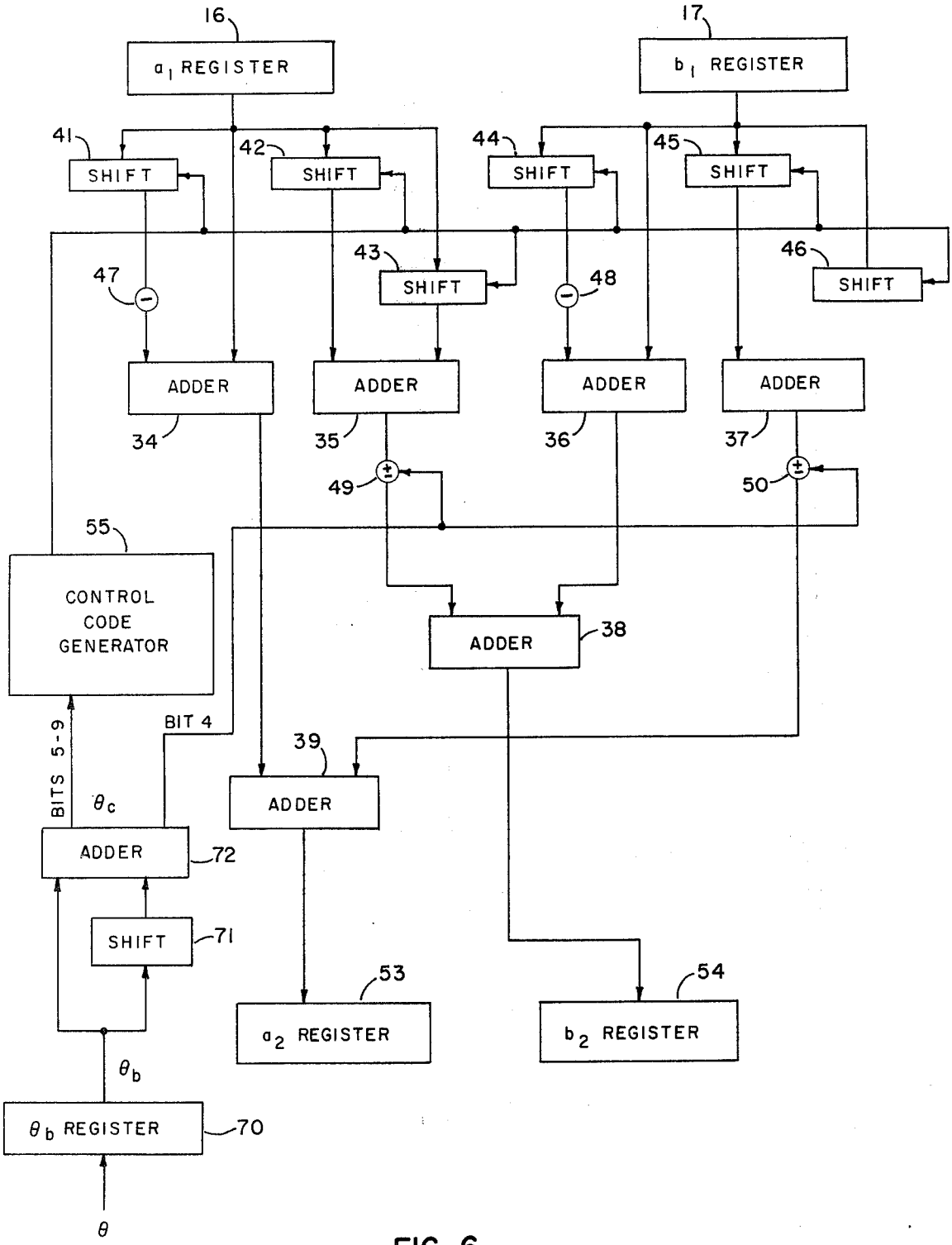

16
a_1 REGISTER
17
b_1 REGISTER
41
SHIFT
42
SHIFT
44
SHIFT
45
SHIFT
43
SHIFT
46
SHIFT
47
48
ADDER
34
ADDER
35
ADDER
36
ADDER
37
49
50
55
CONTROL CODE GENERATOR
ADDER
38
BIT 4
BITS 5-9
θ_c
39
ADDER
72
SHIFT
71
53
a_2 REGISTER
54
b_2 REGISTER
θ_b
θ_b REGISTER
70
θ

FIG. 6

VECTOR ROTATOR

SUMMARY OF THE INVENTION

The invention performs vector rotation in a binary machine (computer). Inputs are the rotation angle and the real and imaginary components of the vector to be rotated. Outputs are the real and imaginary components of the vector after rotation. Vector rotation occurs in 2–6 sequential stages depending upon accuracy requirements. Rotation angles at each stage are selected for easy mechanization on a binary computer.

Rotation angles available in the first stage are in even 30° increments so that rotations can generally be accomplished by shift/add operations, sign changes and reversals of real and imaginary components. Rotation angles available in second and succeeding stages are selected such that the binary representations of their sines have at most 2 nonzero bits. This characteristic assures that multiplications can be replaced by shift-/add operations.

This vector rotator scheme has been configured for use in taking fast fourier transforms. Special cases of vector rotation include conversion from polar to rectangular coordinates and computations of sines and cosines.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a graphic illustration of the vector rotation proper;

FIG. **2** is a graphic representation of the rotation of the vector in the first stage;

FIG. **3** is a block diagram showing the first stage functional circuit;

FIG. **4** is a table showing the first stage rotation requirements;

FIG. **5** is a table showing the second stage rotation requirements; and

FIG. **6** is a block diagram showing the second stage functional circuit.

DESCRIPTION OF THE PREFERRED EMBODIMENT

The function of this method is to perform vector rotation in a binary machine (computer). The method performs this rotation in a sequence of rotations which have unique properties allowing the rotations to be performed either faster or with less hardware than conventional techniques.

FIG. **1** illustrates the problem being solved by this invention. Vector $\overline{V}$ specified by its real and imaginary components a and b is to be rotated by an angle θ. It is desired to determine the real and imaginary components a_2 and b_2 of the vector $\overline{V}_2$ following rotation.

Mathematically,

$$\overline{V} = a + ib$$
$$\overline{V}_2 = \overline{V}\, e^{i\theta}$$
where $i = \sqrt{-1}$

By a familiar identity,

$$e^{i\theta} = \cos\theta + i \sin\theta$$

and thus,

$$\overline{V}_2 = (a + ib)(\cos\theta + i \sin\theta)$$
$$= a\cos\theta + i\, a \sin\theta + i\, b \cos\theta - b \sin\theta$$
$$= (a\cos\theta - b\sin\theta) + i\,(a\sin\theta + b\cos\theta)$$
$$= a_2 + i\, b_2$$

Thus

$$a_2 = a\cos\theta - b\sin\theta$$

and

$$b_2 = a\sin\theta + b\cos\theta$$

To define the function of this method more clearly, its inputs and outputs are as follows:

Inputs: a, b, θ
Outputs: a_2, b_2

The solution in common practice proceeds in three parts as follows:

1. Determine sin θ and cos θ;
2. Perform four multiplications, a $\times$ cos θ $a \times$ sin θ, $b \times$ cos θ, $b \times$ sin θ; and 3. Perform two additions $[a \cos\theta] + [-b \sin\theta]$ and $[a \sin\theta] + [b \cos\theta]$.

This conventional approach can be expensive due to the need for calculating sin θ and cos θ as well as because of the difficulty of performing four multiplications.

Two primary differences exist when compared to conventional methods as follows:

1. This invention does not require explicit calculation of sin θ and cos θ; and
2. This invention replaces the referenced four multiplications by shift/add operations which are, in general, much simpler.

The invention being claimed is more general than the specific embodiment described. This embodiment was developed for one particular application known to the inventor. The invention is applicable, with minor design modification, to a wide variety of requirements.

The example embodiment selected is for a special purpose data processor to perform Fast Fourier Transforms (FFT). This data processor takes as an input a large number of complex time samples of an electrical signal and calculates the Digital Fourier Transform thus producing the frequency spectrum of the signal. Calculation of the FFT requires an enormous number of vector rotations, and indeed, the greatest portion of FFT cost is related to these vector rotations.

The example embodiment is for 4,000 point transform being computed by 12 sequential stages arranged in a "pipeline" fashion. The data processor uses the Sande-Tukey algorithm in a radix 2 design (two points transformed at a time).

Random cancellations and enhancements produced by combining a large number of samples makes it possible to achieve adequately precise overall performance with an accuracy of roughly 1% at each rotation stage. Thus, sin θ and cos θ need only be determined to the 7th binary bit. Actually, performance is adequate even if the 5th and 6th bits are occasionally in error.

Vector rotation for this embodiment is performed in two stages. The first stage performs a rotation θ of $n.30°$
where $n = 0, 1, 2, 3, \ldots, 12$.
The vector being transformed is within $\pm$ 15° of the desired fianl position after completion of this first rotation.

The second stage performs rotation by an angle θ_2 whose sine possesses at most two nonzero binary bits. Since only two nonzero bits exist, multiplication of a number by these sines can be implemented as a simple shift/add operation.

The cosines of angles less than **15°** are very close to unity and for the θ_2 angles have the property that the term $(1 - \cos\theta_2)$ has only one nonzero bit out to the 7th binary place. Hence, multiplications by cos θ_2 can also be implemented as a shift/add operation.

FIG. **2** shows as dark lines the 12 rotation angles possible in the 1st stage. Rotations by the 4 angles given by $90°.n$ do not require multiplications at all, but can be accomplished with sign changes and by reversals of real and imaginary components. The remaining **8** vectors each have a sine or a cosine equal to $\pm 2^{-1}$. Multiplication by this angle can be accomplished by a simple one

bit shift. Assume for the moment that it is the sine which is 2^{-1}, and let us find the cosine.

$$\cos\theta = \sqrt{1-\sin^2\theta} = (1-\sin^2\theta)^{1/2}$$

by the binomial expansion,

$$\cos\theta = 1-2^{-1}\sin^2\theta - 2^{-3}\sin^4\theta - 2^{-4}\sin^6\theta - 5\cdot 2^{-7}$$
$$\sin^8\theta - 7\cdot 2^{-8}\sin^{10}\theta \ldots$$

for $\sin\theta = 2^{-1}$

$$\cos 30° = 1 - 2^{-3} - 2^{-7} - 2^{-10} - 2^{-13} - 2^{-14} \ldots$$

For this application it is permissible to ignore the 2^7 term and succeeding terms. Thus,

$$\cos 30° \cong 1 - 2^{-3}$$

The cos 30° multiplication can be accomplished by shifting three bits to the right and subtracting from the unshifted number. FIG. **4** shows a table of the mathematical results required for each of the 12 rotation possibilities. Note that $\overline{V}_1$ is the vector which emerges from the first rotation and that

$$\overline{V}_1 = a_1 + i\,b_1$$

Note also that when $\cos\theta = 2^{-1}$, then $\sin\theta \cong 1-2^{-3}$.

FIG. **3** shows the functional circuit diagram for implementing this technique. Major components are shifters **5–8,** input registers **9** and **10,** adders **11–14,** and output registers **16** and **17.** The control signals necessary to set switches **20–24** and change signs of sign generators **28–31** are generated by a separate subsystem known as the control code generator **33.**

The $\overline{V}_1$ vector which emerges from the **1st** Rotator state will be within ± 15° of the desired final angle. The angular rotation accomplished by the second stage θ_2 is restricted to 12 possible values each of which has the property that sin θ_2 has at most 2 nonzero bits in its binary representation. Thus, multiplication by sin θ_2 requires only two shifts and one add.

Since θ_2 is small, cos θ_2 is very close to unity. It happens that for each of the 12 θ_2 angles $(1 - \cos\theta_2)$ has only one nonzero binary bit out to the 7th binary place. Thus multiplication by cos θ_2 can be accomplished by shifts and adds.

FIG. **5** shows a table of the 12 θ_2 angles; the respective values of sin θ_2 and cos θ_2; and reveals the maximum errors $\Delta\cos\theta$ introduced by using the cos θ_2 values shown. The greatest error introduced is 2^{-8} and this is negligible for this application. Required outputs in mathematical form are also given in the table of FIG. **5.**

Note in FIG. **5** that θ_2 angles generally increase in 0.9° steps thus introducing ±0.45° errors. Between the last two θ_2 values, however, a gap of 3.7° exists. Thus, errors as large as ±1.8° can exist after second stage rotation. This error is judged to be acceptable for this application since errors are within ±0.9° for 90 % of the required rotation angles.

If an application requires greater accuracy, however, then a third rotation stage can be used without greatly expanded hardware. The third stage would require at most two shift registers and four adders. It would have a set of shift angles θ_3 having sin θ_3 values identical to the sin θ_2 values of FIG. **5,** except that all nonzero bits would be shifted 3 bits to the right. Thus, the greatest angular error emerging from the third stage would be ±0.23° which corresponds to errors in sin θ_3 beyond the 7 bit accuracy generally used in this application. Note that for the third stage cos θ_3 would be assumed equal to unity, since differences would be less than 2–11.

FIG. **6** shows the functional circuits needed to perform second stage rotation. Major components are **6** adders **34–39, 6** variable shift registers **41–46,** four sign generators **47–50,** and **2** output registers **53** and **54.** Necessary control signals are provided by control code generator **55** to variable shift registers **41–46.**

The particular Fast Fourier Transform application being described uses the Sande-Tukey algorithm where points are transformed in order using radix 2 pipeline stages.

The angular rotations required for each pair of points vary in a regular progression according to the formula,

$$\theta° = -360°k\,\frac{2^{(m-1)}}{2^M}$$

where θ is given in degrees

M = the number of stages in the pipeline

m = the stage of the pipeline being addressed,

$m = 1, 2, 3, \ldots, M$

And k is an index variable which follows a counting sequence

$$k = 0, 1, 2, \ldots, (2^{(M-1)}-1)$$

Thus the required angular rotations start at zero and progress in even increments through negative angles up to −180°.

Define a term (or register **70** in FIGS. **3** and **6**) $\theta_b = \theta°$. The θ_b register has the bit values shown below.

	1	2	3	4	5
b =	90°	45°	22.5°	11.25°	5.625°

Note that the first bit of θ_b reveals the quadrant location of the desired rotation angle without reference to any other bit value. This being true because the rotations only goes up to −180°. If the contents of the θ_b register are multiplied by the fraction ¾, then a new term (or register) is formed whose bit values are favorable for determining rotator control bits. The denominator 4 can be dropped as this only shifts the result two places to the left. This is taken care of by the interpretation of the value of the bit positions by generators **33** and **55.** Since the binary representation of 3 is 11, this multiplication is accomplished by one shift by shifter **71** and one add by adder **72.** This new term θ_c has the bit positions (with interpretation) shown below:

	1	2	3	4	5	6
θ_c =	120°	60°	30°	15°	7.5°	3.75°

The proper **1st** stage rotation angle can be determined from the first 4 θ_c bits above as given in the table of FIG. **4.** The 0° through +150° are not used in the progression of θ_1. These four bits can be used to address a **7** word decode memory contained in control code generator **33** which produces **5** control bits selected to directly operate switches **21–24** and change signs in the circuit shown in FIG. **3.**

The 4th bit of θ_c shows the direction of the rotation required for the second stage. It can be used directly as one of the control bits for that stage to control the sign generators **49** and **50.**

The first 3 bits of θ_c can be ignored completely in determining second stage control bits since the **1st** stage rotation exactly eliminates these bits.

The control code generator **33** uses the **5th** and smaller valued bits of θ_c to produce control bits for the second rotator stage. The first requirement for generating these control bits is to determine the θ_2 angle which is closest to the desired rotation. The table of FIG. **5** shows the programming for the angles. The − mean

that generator **55** will produce the same code output whether that bit position is a 1 or a 0. Available θ_2 angles, of course, are those which have at most two nonzero bits in their binary sine values. Thus it is worthwhile to investigate the bit correspondence between θ_c and sin θ_2.

	1	2	3	4	5	6	7	8	9
θ_c	120°	60°	30°	15°	7.5°	3.75°	1.875°	0.95°	0.48°
sin θ_2			30°	14.5°	7.2°	3.6°	1.8°	0.9°	
			1	2	3	4	5	↑	
								Least significant sin θ_2 bit	

Note that the 30° values correspond exactly and that other bits differ at most by 0.5°. Differences for smaller angles are within 0.1°. This correspondence is not surprising in view of the fact that sin θ is a nearly linear function for small angles.

The bit value correspondence between registers θ_c and sin θ_2 makes it possible to use only bits **5–9** of θ_c to determine the proper θ_2 rotation. These five bits can reference a **12** word decode memory also contained in generator **55** which outputs a plurality of easily mechanizable control bits. These bits control the amount of shift of each variable shifter **41–46.** If an open circuit is desired at a shifter, then that shifter shifts its input to the right a number of places more than the total places in its input. In this way its output is zero and a open circuit.

For small angles, the ratio of bit values between θ_c and sin θ_2 approaches $\pi/3$. For applications requiring greater accuracy than the example, closer correspondence between the θ_c and sin θ_2 registers can be obtained by multiplying θ_c by $\pi/3$. This multiplication requires only two shifts and two adds to yield 11 bit accuracy due to the sparcity of **1's** in this particular number.

In operation the first stage of rotation - FIG. **3** - has fed into it the coordinates of the vector to be rotated: a, b, and θ. The value of the θ is recorded in the register **70** as θ_b. θ_b is changed to θ_c by shifters **71** and adder **72.** The output of adder will have its 1–9 bits representing θ_c; however, only bits 1–4 are used in the first stage rotation to select the control words from control generator **33.**

The output of control generator **33** is shown to have **5** bit outputs. Bit **1** controls the position of switch **24,** bit **3** controls the positions of switches **20–23,** bit **4** controls the sign of sign generator **30,** and bit **5** controls the sign of sign generator **31.** Bit **2** controls the signs of sign generators **28** and **29;** however, since these sign generators should be opposite from each other, an inverter **69** is inserted in series between bit **2** and sign generator **28.** As can be seen from reference to FIG. **4** all the required outputs of a_1 and b_1 are obtained by this first stage rotational circuit. Control generator **33** can be any of the well known "programmable read only" devices such as the 74186 PROM (see "TTL Data Book for Design Engineers" copyright 1973 by Texas Instruments).

The output of the a_1 register **16** and the b_1 register **17** is also fed to the inputs of the second stage rotational circuit FIG. **6.** Again adder **72** produces all the bits 1–9; however, bits 1–3 are not needed for second stage rotation. Bit **4** is used directly to control the sign of the rotation by direct connection to sign generators **49** and **50.** Bits 5–9 of θ_c are used to address the control code generator **55** so as to produce proper word output. The word output from control generator will contain a plurality of bits which are individually connected to shift registers **41–46** (shown in cable form in FIG. **6**). Shift registers **41–46** may take the form of the well known shift registers which have their amount of shift controlled by a clock which in turn is controlled by the bits from control generator **55.** In this way the required outputs for θ_2 can be derived by programming the word outputs of control generator **55** in accordance with the table shown in FIG. **5.** The outputs of the a_2 register **53** and b_2 register **54** represent the desired coordinates of the vector after rotation. Control generator **55** can be any of the well known "programmable read only" devices such as the 74186 PROM (see "TTL Data Book for Design Engineers" copyright 1973 by Texas Instruments).

The shift registers **41–46** could also be each a plurality of shift registers and a switch each in parallel with the others. The switches could then be controlled by the outputs from control generator **55.** For example shift register **41** could take the shape of three shift registers in series with three switches each shift register and switch being in parallel with each other. One shift register being a shift **7** places shift register, the second shift register being a shift **6** places shift register, and the third shift register being a shift **5** places register. The control generator having bits connected to each of the switches to control their operation. Looking at FIG. **6** one can see that adder **34** produces an output corresponding to the first half of the a_2 register as depicted in FIG. **5.** Adder **37** corresponds to the second half of the a_2 register, likewise the b_2 register has its first and second parts corresponding to the outputs of adders **36** and **35.**

OTHER EMBODIMENTS

If vector $\overline{V}$ is specified in polar coordinates as,

$$\overline{V} = Re^{j\theta} = R\cos\theta + jR\sin\theta$$

and it is desired to find the real and imaginary components, then a simplified vector rotation technique can be employed. Note that this problem is a special case of vector rotation in which $a = R$ and $b = 0$. The first stage of the rotator requires only 2 rather than 4 adders.

If vector V is specified as,

$$\overline{V} = 1 \cdot e^{j\theta} = \cos\theta + j\sin\theta$$

then sin θ and cos θ are directly computed by the rotator device. The 1st rotation stage requires no adders and the second stage is reduced by almost half if only the sine or the cosine need be computed.

Innumerable variations of the rotator device are possible depending upon specific applications and upon specific hardware capabilities. Applications requiring high degrees of accuracy will need as many as four or five stages and will need to perform some actual multiplications due to the significant numbers of nonzero bits which exist in the cosine functions in bit positions beyond 10.

Applications which are not keyed to a rigid time schedule can take advantage of cases in which convergence is quicker than worst case predictions. For these applications it may be worthwhile to include more possible rotation angles in each stage. For example, one could add θ_2 angles having nonzero bits in the following bit positions, (6 and 7), (5 and 7), (4 and 7) and (3 and 7).

I claim:

1. A method for determining the vector components of a vector to be rotated $\theta°$ comprising the steps of:

determining the components of the vector rotated $\theta_1°$ in even 30° increments such that θ_1 equal θ within a $\pm$ 15° error;

selecting rotation angles from 0 to 15° in steps to represent $\theta_2°$ such that $\theta_1 \pm \theta_2$ can be selected such that any error between the algebraic sum of them and θ is within $\pm$ one half of one of said steps;

rotating $\theta_2°$ the vector components of the vector already rotated $\theta_1°$; and

selecting θ_2 such that the combination of θ_1 and $\pm \theta_2$ is most nearly equal to θ.

2. A method as set forth in claim 1 further comprising the step of:

selecting the stepped values of θ_2 such that the sign of each θ_2 has at most two nonzero bits in its binary representation.

3. A method as set forth in claim 2 further comprising the steps of:

representing θ as a binary number;

multiplying this number by 3 so as to obtain a second binary member having a plurality of output bits;

determining θ_1 by the progression of the first four highest bits.

4. A method as set forth in claim 3 further comprising the step of:

utilizing fifth highest bit and lesser bits for representing in steps θ_2.

5. A method as set forth in claim 4 further comprising the step of:

performing said θ_1 rotation of the components by shift/add operations, sign changes and reversals of real and imaginary components.

6. A method as set forth in claim 5 further comprising the step of:

performing the required multiplications in the θ_2 rotation by shift/add operations.

7. A method as set forth in claim 4 further comprising the step of:

utilizing the fourth highest bit to determine the sign of θ_2.

* * * * *