PHASE COMPUTERS

by B.R. GAINES and P.L. JOYCE *

Abstract.

The phase computer is a modular, all-digital, data-processing and control system, in which many of the advantages of analog and digital computers are combined through the use of an incremental digital processor under program control. The incremental processor enables pseudo-analog computing loops to be established so that complex operations, such as division, square-rooting, rectangular/polar/hyperbolic co-ordinate transformations, and so on, are performed with the same speed and simplicity as simpler operations, such as addition and data transfer. The programming facility enables complex computations to be performed as a sequence of elementary operations, and hence overcomes the inordinate hardware demands of previous parallel incremental computers.

1. New Directions in Computer Science.

The boundaries between digital and analog computation, which at one time seemed very clear and well-established, have been eroded during recent years. The increase in the complexity of logical control of analog computers, from a simple *mode* switch to individual solid-state switching of integrator modes and interconnections through a logic patchboard in the hybrid computer, [1] has opened up possibilities of *programmed sequences* of computations on the analog computer. [2] The increasing speed of digital computers, together with increasing sophistication and efficiency of integration routines, has made all-digital simulation of analog computations feasible, economically and often in real time. [3]

This overlap between the applications of analog and digital computers is symptomatic of the increasing flexibility and freedom of choice which we have in selecting a data-processing or control system. There are two major factors affecting the extent of this freedom — one is the availability of large-scale integrated sub-systems, which will shortly give us the arithmetic power of a small digital computer in a single package, plus such items as digital/analog convertors, multiplexers, and operational amplifier configurations, again in small packages — the other factor is our ability to utilize these sub-systems and design complete data-processing systems around them.

The ease of setting up a programmed sequence of instructions, which will cause the single processor of a conventional digital computer to implement a particular computation, has increased tremendously during recent years. In particular the introduction of compilers for high-level languages has made this feasible without detailed knowledge of the computer structure. Any expansion of our computing capabilities by the introduction of new elements, even if they have advantages in size, economy and computing power, must also make available the equivalent of compilers to interface between the professional skills of the user and the structure of the computer.

With analog computers the concessions to the user have been more slight, perhaps because their prime use has been in engineering where knowledge of electronics and differential equations is to be expected. There have, however, been notable advances in set-up and check-out procedures based on the use of elements such as servo-set potentiometers, and these are being further extended with the development of FORTRAN-like languages for controlling analog computers. [4]

There are other computing techniques where virtually no concessions have been made to the general user, and which have therefore remained known by name and principle only, except to the few who have found their particular advantages of sufficient value to make the effort involved in using them worthwhile. In particular incremental digital computers, such as the DDA [5, 6] and operational-digital computer, [7, 8] have been very successful in limited areas of application, but have a very poor literature and cannot be regarded as well-established.

All-digital systems, however, offer many advantages over analog and hybrid systems in terms of accuracy, small-size, long-terme storage, reliability, low temperature-dependence, ease of digital control and micro-circuit fabrication. Now that cheap, reliable integrated circuits, capable of performing complex arithmetic functions are becoming increasingly available, the cost of parallel incremental computers is falling to a level competetive with that of equivalent analog devices, and their further development is very attractive.

What is lacking in established incremental digital computers is the control of the interconnections and functions of the computing elements by digital logic. This is the essence of the conventional stored-program digital computer, in that it enables the arithmetic unit to be placed under program control, and is also the feature which differentiates hybrid machines from conventional analog computers. Machines have been described recently in which an analog, or analog/digital, arithmetic unit is placed under program control, [9, 10] and an incremental digital unit may be used in a similar configuration. By replacing the parallel binary arith-

(*) Standard Telecommunication Laboratories, London Rd., Harlow, Essex, U.K.

metic unit of a conventional digital computer with a complete DDA-like computer, the advantages of pseudo-analog computing loops may be gained in an all-digital, programmed system.

The advantages of the programmed incremental digital computer in some applications may be traced to the basic distinction between *analog* (not merely electronic analog) and *non-analog* computers. A pair of DDA integrators may, for example, be connected to rotate a two-dimensional vector through an angle, and, under program control, this operation can be caused to take place by a single instruction. A conventional digital processor will not normally have this facility — its basic operations are ADD, SHIFT, COMPLEMENT, TRANSFER, etc., and these must be combined in an appropriate sequence to rotate a vector through an angle. The DDA integrators offer a direct analog of vector rotation and, in computations which require this operation extensively, the instruction sequence for a programmed incremental computer will be very much shorter than that for a conventional digital computer. This may have important consequences, such as the elimination of the need for a core-store in computations where there is little data storage but a long program.

It is suggested that the on-line control and communication computer of the future will be truly hybrid, combining analog, incremental-digital and parallel-digital arithmetic. It will be under program-control, but much more of the program will be in read-only store than at present. The analog elements will be used for high bandwidth channels and their parameters will be under digital control. The incremental digital arithmetic will be used wherever medium-speed analog or slow-speed digital operations would at present be used. The parallel-digital arithmetic will be used where high speed is essential, and for control and logical operations, such as address determination, character-handling, and so on.

Separation into different modes and applications of the computing techniques in this way, however, will be more conceptual than relating to actual hardware, since the building blocks of the computer will be a few general-purpose elements whose function will be specialized by their control and inter-connection. Compilers for problem-orientated languages associated with the computer will not only determine the program sequence but also the hardware required to implement it. Constraints, such as speed, cost and flexibility, must be taken into account when deciding upon the particular hardware/program combination to be used.

The development of the basic modules for a computer system of this type, with the concomitant languages and compilers, is still in the future. At present there are a number of peripheral system developments concerned with graphic displays, radar systems, message-switching systems, telephone line equalization, process control, flight simulation, and so on, where special-purpose hardware is being used to supplement general-purpose digital computers. As systems engineering becomes an increasingly prominent discipline, it may be expected that the main computer and peripheral hardware will become considered increasingly as a whole, especially where a number of computers are operating together in one system. The « real-time » compilers which will have to be developed for these systems will specify the data-rates and formats for peripheral devices, and hence the optimization of hardware/software combinations may come about naturally in the development of real-time computer systems.

In the meantime there are many aspects of the hardware for future hybrid computers which may be developed at present. Fast, solid-state switching of analog devices and the fabrication of monolithic analog/digital circuits, such as D/A convertors and multiplexers, are rapidly expanding technologies which offer much for the future. The development of incremental digital computing techniques and programmed incremental computers also has many attractive possibilities, and it is this which forms the main topic of the following technical section of this paper.

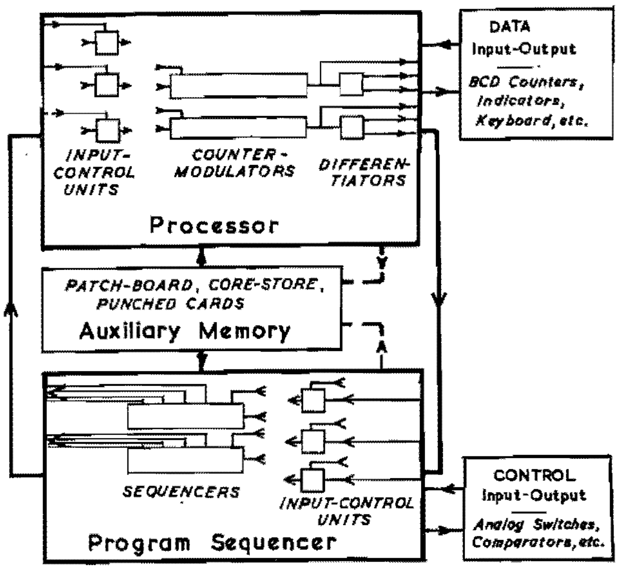


Fig. 1. — Phase Computer Schematic

2. The Phase Computer.

The original impetus to the design of the phase computer arose in a study of the potential applications of *stochastic computers* in radar systems. The stochastic computer [11, 12, 13] is an incremental digital computer whose internal sequences are probabilistic and, through their lack of patterning, simplify the hardware required in computations such as multiplication.

One particular problem of great interest was the realization of the least-squares smoothing and prediction equations for automatically tracking radar targets. [14] Previously these had been implemented on a general-purpose digital computer, but it was desired to use them in circumstances where no conventional computer was available and the simplest and most economical equation-solver was required. These equations involve addition, subtraction, multiplication, division, sine/cosine generation, and inverse sine/cosine resolution, so that they encompass a wide range of arithmetic operations.

Incremental computing techniques, as used in the stochastic computer, were an obvious possibility — analog computing techniques were another, but the low data-rate necessitated a long storage-time, and the computational accuracy required was prohibitive.

Some important features of these equations were that they arose in a data-sampling system, and also that, whilst as a whole very formidable, they could be reduced to a simple sequence of elementary operations. This led to the consideration of a *hybrid* stochastic computer in which mode-control logic was used to switch the integrators, and their interconnections, to perform each of the elementary operations in turn — a programmed incremental computation.

It was found in doing this that some of the defects of the stochastic computer, in particular the random variance of its results, could be eliminated, since it was possible to generate pseudo-stochastic sequences which were « uncorrelated » over one computational cycle. One of the pseudo-stochastic sequences was a repetition one, as used in the DDA and (less accurately) in the operational digital computer. The other sequence was equivalent to a digital mark/period ratio, and its generation by comparing the relative *phase* of two cycling counters was naturally incorporated in the programming sequence of the computer — this process also gave the computer its name.

A block diagram of the phase computer is shown in Figure 1, and a photograph of the laboratory experimental prototype in Figure 2. The machine has a conventional four part structure, consisting of processor, program sequencer, auxiliary storage, and input/output devices, but the processing section is incremental digital, the sequencing section is very extensive and may operate several concurrent programs, and the auxiliary storage in the laboratory prototype is realized entirely through a patch-board. A detailed description of the elements which make up this block diagram is given in the following sub-sections, and their use in addition, subtraction, multiplication, division, sine/cosine generation, and so on, is outlined, together with some examples of input/output devices for numeric and analog data.

2.1 *Counter/Modulator Computing Elements*

The active storage registers of the phase computer are unidirectional, synchronous binary counters which, at a clock pulse, increment by unity if their INPUT line is ON, and reset to zero if their RESET line is ON. Although the overall counters operate synchronously, they may be internally asynchronous, as shown in Figure 3.

If the proportion of ON logic levels on the INPUT line to a digital counter is considered as the input variable, then the counter may be regarded as a discrete version of an analog integrator, in that its stored count will be proportional to the input times the period of integration. The counter alone, however, lacks an output in the same form as its input — the integral is available as a binary number rather than a sequence of logic levels. If the counter contains N flip-flops, so that its maximum count is $2^N - 1$, and the count in it is k, then an output sequence is required in which the

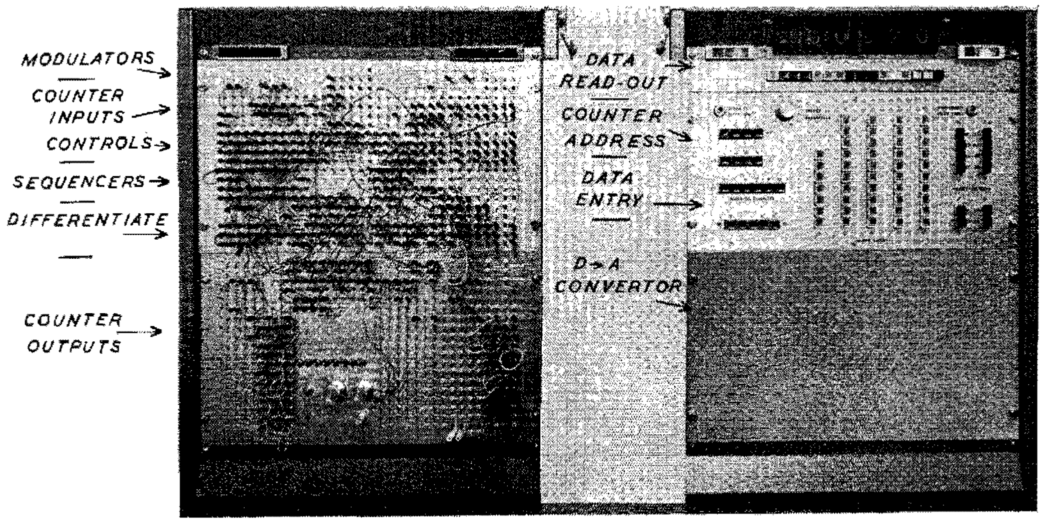


Fig. 2. — Laboratory prototype Phase Computer

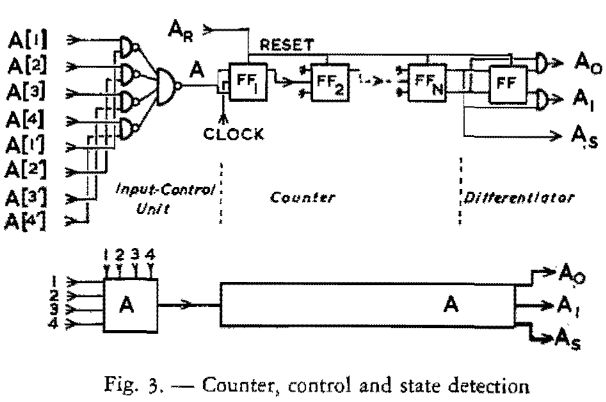


Fig. 3. — Counter, control and state detection

proportion of ON logic levels is $k/2^N$; that is, the number stored in the counter regarded as a binary fraction. With this output the resemblance to an analog integrator is complete, and the counters may be cross-connected to perform the functions of analog computing loops, as they are in the DDA and operational-digital computers.

Several alternative techniques for generating incremental digital sequences have been investigated in different versions of the phase computer. For the first laboratory prototype a synchronous version of the binary rate multiplier was developed. This device was originally used at MIT for linear interpolation in driving the stepping motors of automatic machine tools, [15] and more recently by Schmid in his operational hybrid computer. [16]

It may be shown that the incremental sequences generated by the binary rate multiplier are not the best that could be obtained, [17] in the sense that the distribution of ON logic levels is uneven and could be smoother. This leads to inaccuracy in computation and, with increasing availability of multiple full-adder elements [18] at low cost, it is advantageous to use an « add-and-overflow » technique to generate incremental digital sequences, as used in conventional DDAs. This has the additional advantage that, by slight extension of the hardware, a *predictive* technique may be used (open trapezoidal integration) which takes account of the changing state of the register and leads to a further improvement in accuracy. The various techniques for generating incremental sequences have been described in the literature. [16, 5, 19]

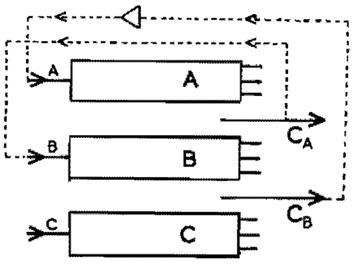


Fig. 4. — Ensemble of counters

Whatever technique is used for generating the incremental digital sequences, the basic counters with incremental outputs can be represented in block form, as illustrated in Figure 4. Counters A and C, for example, have an output line, C_A, which corresponds to the fractional binary number in A *modulating* the cycling of the counter C. If the INPUT line to A is OFF and that to C is ON, then the proportion of ON logic levels on the line C_A is equal to the fractional binary number in the register A. If the INPUT line to C is OFF then so is the output line C_A. If the INPUT line to A is ON together with that to C, then the proportion of ON logic levels on C_A corresponds, in some sense, to the varying number in the register A. The modulator counter, C, may occasionally be a virtual counter, not necessarily existing in hardware, but, in general, it is a normal counter equivalent in its function to other counters such as A. Thus any counter/register is potentially capable of modulating the counting of another such counter/register.

2.2 *Pseudo-Analog Computing Loops*

Because the counters are synchronous and controlled by logic levels rather than pulses, they may be used in pseudo-analog computing loops even though the counting is unidirectional. Figure 4 illustrates one common form of interconnection in which the two counters, A and B, both modulating the counter, C, are connected in cascade with negative feedback around the pair. The output, C_A, of A is connected to the input of B, and the output, C_B inverted, is connected to the input of A (simple inversion could not be used with a device whose output was a pulse rather than a logic level). The counters with this feedback are equivalent to a pair of analog integrators in cascade with negative feedback around the pair, a configuration which will undergo simple harmonic oscillation. The counters show similar behaviour but, because their topology is different (an increment from maximum count leads to minimum count rather than limiting), the waveforms generated differ in some ways from those of the analog equivalent.

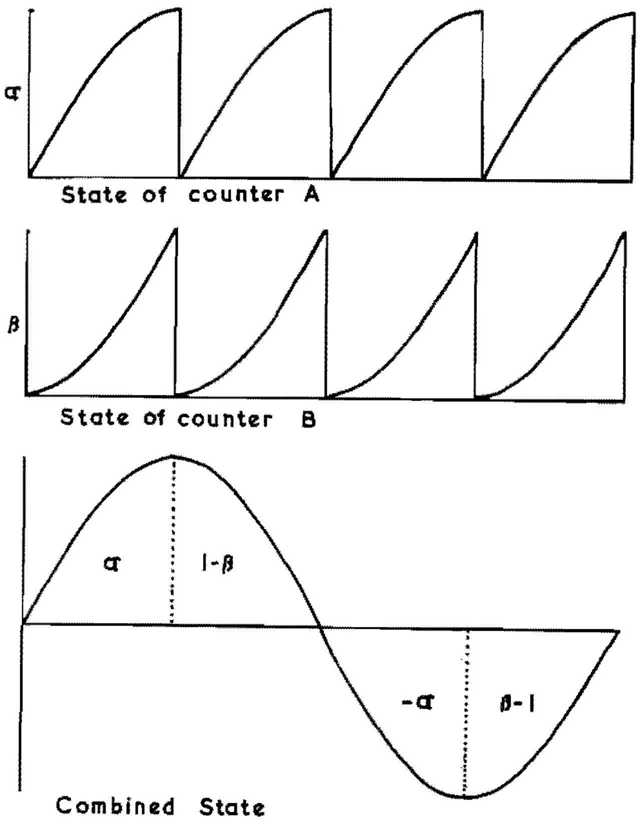


Fig. 5. — Digital Sine Generator

Consider both counters initially at zero, and the input to C ON. The two lines, C_A and C_B, will both be OFF, and hence counter A will start counting at maximum rate because it receives C_B inverted, whilst counter B will not count at all. As A begins to fill, however, C_A begins to come ON and counter B starts to count. As

B begins to fill, C_B begins to come ON, and the count-rate of A decreases. The net result is that the counts in A and B follow the repetetive patterns shown in Figure 5, each cycle of which corresponds to one quadrant of simple harmonic generation. If we consider the fractional binary number, α, in A for one quadrant, followed by one minus that in B, $1 - \beta$, for the next quadrant, followed by $-\alpha$, followed by $\beta - 1$, then we obtain a complete sine wave as shown in the lower part of Figure 5. A cosine wave may be constructed similarly, and hence the counters of Figure 4 may be used as a four-quadrant, pulse-density sine/cosine generator if appropriate logic is used to select their outputs according to the quadrant.

2.3 *Interconnection Control*

An example has been given in the preceeding section of the use of phase computer counters with modulation facilities in a simple DDA configuration. This possibility of cross-connecting the registers to generate, in a pseudo-analog fashion, functions which are difficult to realize with parallel digital arithmetic is an important feature of the phase computer. In general, however, the loops used will be far simpler than those of conventional DDAs, the majority of counters generating ramps rather than complicated functions, and the basic computational operations realized in this way, such as addition, subtraction, multiplication and division, are combined in programmed sequences to yield more complex overall behaviour.

The essence of programming, in all computers, is the control of the interconnections between processor elements by logic levels on CONTROL lines. The configuration required for this control is an AND/OR combination, fundamental in Boolean algebra, and readily fabricated with NAND gates. An example of a four-way AND/OR gate is shown in Figure 3 at the input to the counter. It implements the function :

$$A = A[1] . A[1'] + A[2] . A[2'] + A[3] . A[3'] + A[4] . [4'],$$

and may be regarded as a digital selector switch in which the INPUT line A $[j]$ is switched through to the output if the CONTROL line A $[j']$ is ON alone. The symbol for this gate, called an input/control unit in the phase computer, is shown in the lower part of Figure 3.

Input/control units placed at the inputs of phase computer counters enable them to be interconnected in several alternative patterns, any one which may be selected by activating the appropriate CONTROL lines. In order to cause the computer to go through a programmed sequence of operations, it is necessary to activate the CONTROL lines corresponding to various interconnection patterns in the required sequence at the correct times, and a unit called a *sequencer* is used to perform this function.

2.4 *Sequencing Computations*

A *sequencer* may be regarded as the digital equivalent of a rotary switch which activates one, and only one, of

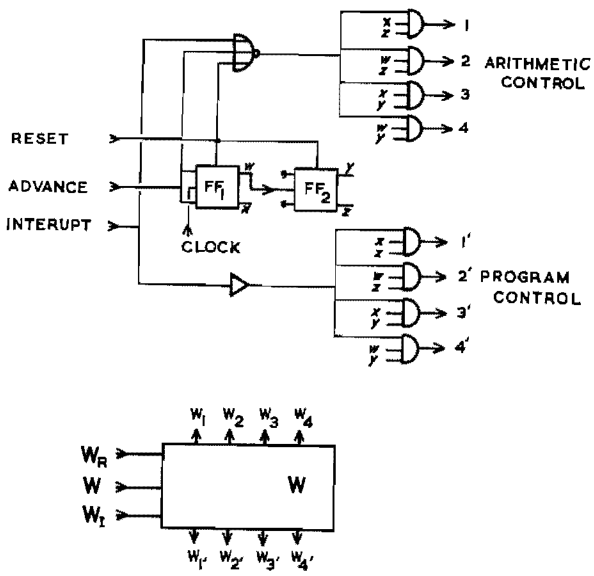


Fig. 6. — Four-way sequencer

a number of output lines in turn. It is realized in practice, as illustrated in Figure 6, by a small counter, to each of whose states there corresponds an OUTPUT line which is ON when the counter is in that state. The state of the counter may be changed at a clock pulse by ADVANCE and RESET lines which cause it to increment by unity, or reset to zero count, respectively; there is also inhibitory logic to the outputs whose function will be described in Section 2. 7. Input/control units are normally located in the inputs of the sequencer so that advancing and resetting may also be controlled by sequencer outputs.

The outputs of sequencer units are used to activate CONTROL lines, and different states of the sequencer correspond to different interconnection patterns amongst the phase computer counters. In any particular computation, signals will be required to advance the sequencer as each step in the computational sequence is completed. These are obtained by means of units called *differentiators* which detect the states of the counters, in particular the zero and mid-range states.

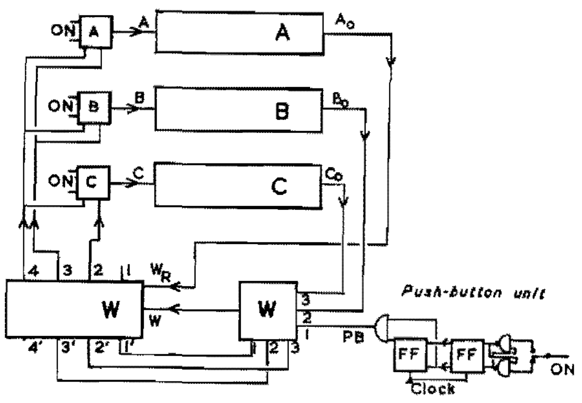
The function of a digital differentiator in the phase computer is to indicate that a counter has entered the state 00000...00000, in which its count is zero, or, alternatively, that it has entered the state, 10 000... 00000, in which its count is 2^{N-1} (that is, mid-range for a counter with N flip-flops). These states may be recognized by detecting a change in the most significant bit of the counter, and logic for this is illustrated at the last stage of the counter in Figure 3. A delay flip-flop stores the previous state of the most significant bit, and gates are used to detect any difference in the present state. The symbol for a differentiator may be incorporated in that for a counter, and, in the lower part of Figure 3, the differentiator output lines A_0 (ON when counter A has just entered the 00000... 00000 state) and A_1 (ON when A has just entered the 10 000...00000 state) are shown as outputs from the counter.

The differentiator also forms a convenient means of interfacing control signals from other sources into the phase computer. A push-button unit is illustrated on the right side of Figure 7, consisting of an RS flip-flop, synchronizing JK flip-flop, and a digital differentiator. Depressing the push-button gives an ON level for one clock interval, which may be used to manually advance a sequencer and start a computation.

The use of differentiators and sequencers is best shown by means of a simple example, and the next section illustrates the importance of the counter states detected by differentiators with a simple computation which also leads to the general principle of a *phase counter*.

2.5 *Phase Counting Principle*

Consider the simple operation of reproducing the binary number in one counter, B, in another counter, C (it will be assumed, unless explicitly stated, that all counters have the same range). This might be required for data transfer, or counter C might have a BCD (binary-coded decimal) representation and drive numerical indicators for readout.



	State	1	2	3	4
Seq. W	Reset			A_0	A_0
	Advance	PB	C_0	B_0	
Counter Input	A	0	0	1	1
	B	0	0	1	1
	C	0	1	0	1

0 ≡ OFF
1 ≡ ON

Fig. 7. — Example of simple computation - data transfer

Figure 7 illustrates a configuration, consisting of three counters and a four-way sequencer together with input/control units, for performing this operation. It is assumed that a push-button is used to initiate the transfer, that a third counter, A, is available whose count is initially zero, and that counter C must be zeroed before the transfer commences. The table in the lower half of Figure 7 shows the signals at the inputs to the counters for each state of the sequencer, and also lists the differentiator output which will cause the sequencer to advance from one state to another, or to reset.

State$_1$ of the sequencer corresponds to no activity and the inputs to all the counters are OFF. State$_2$ is entered on a signal from the push-button and counter C starts to count. When it enters its zeroth state the sequencer steps on to state$_3$ and counters A and B start counting until B enters its zeroth state and the sequencer advances to state$_4$. In this state all three counter inputs are ON and the counters rotate until A goes into its zeroth state whence the sequencer is reset to state$_1$ and remains there. It is possible for counters A and B to enter the zeroth state simultaneously (corresponding to zero in B), in which case the sequencer resets directly to state$_1$ from state$_3$.

The sequence of operations in the data transfer thus correspond to passivity in state$_1$ of the sequencer, initialization of C to zero in state$_2$ (if this is not done the computation becomes addition, $B + C \rightarrow C$, instead of transfer, $B \rightarrow C$), and data transfer in states 3 and 4. These last two states have a particular interest because they illustrate the phase counter principle — if the quantity stored in B is taken to be the difference between the binary number in B and the binary number in A, then this quantity is invariant provided the inputs to A and B are either both ON, or both OFF. It will be noted that this condition holds in the table of Figure 7, and that states 3 and 4 taken together correspond to a complete rotation of A from its zeroth state back to its zeroth state.

Counter A, used in this way, is said to act as a *phase reference* for counter B, and the use of the differentiator output B_0 to advance the sequencer from state$_3$ to state$_4$ is a means of reading out the fractional binary number stored in B as the mark/period ratio, number of clock pulses in which output 4 of the sequencer is ON divided by total number of clock pulses in which output 3 or output 4 is ON. It is the extensive use of this technique to read out stored quanties, in an alternative form to the incremental sequence obtained by modulation, that gives the Phase Computer its name.

2.6 *Unsigned Arithmetic Operations*

The table in the lower half of Figure 7 gives all the information required to set up the configuration shown in the upper part of the figure, and further examples will be given in tabular form only. The effect on a register of any computational sequence can readily be ascertained from its table for example, if the initial quantity in counter C is γ (fractional binary number), then the quantity added is : zero in state$_1$ of the sequencer, $1 - \gamma$ in state$_2$, zero in state$_3$, and β (quantity in B) in state$_4$. The total is $\gamma + 1 - \gamma + \beta = 1 + \beta$, which is transfer of the quantity in B with one passage through zero ($1 + \beta$ and β are equivalent so far as the counter is concerned).

Slight variations in the table of Figure 7 will give rise to different computations — it has already been noted that omission of the operations in the column corresponding to state$_2$ leads to the quantity in B being added to that in C and the result left in C. Leaving the reset and advance conditions as before, the following tables correspond to transferring $1 - \beta$ to C, and transferring β to C whilst zeroing B, respectively : —

State	1	2	3	4	and	State	1	2	3	4
A	0	0	1	1	»	A	0	0	1	1
B	0	0	1	1	»	B	0	0	1	0
C	0	1	1	0	»	C	0	1	0	1

Multiplication, division and squaring offer interesting examples of the simplicity of arithmetic operations in the phase computer. Computations in which the result is to replace one of the operands,

$$C \times B \rightarrow C,\ C/B \rightarrow C,\ B^2 \rightarrow B,$$

will be considered, since using another register for the result is simpler and readily derived from these The phase reference counter, A, is additionally required to have a modulator output, A_B, but otherwise the configuration is closely similar to that of Figure 7.

Multiplication, $C \times B \rightarrow C$, is realized in the table ·

State	1	2	3
Reset	0	A_0	A_0
Advance	PB	C_0	0
A	0	1	1
B	0	0	0
C	0	1	A_B

If the fractional binary numbers in B and C are β and γ, respectively, then the final quantity in C is ·

$$\gamma + 0 + 1 - \gamma + \beta\gamma = 1 + \beta\gamma \equiv \beta\gamma .$$

Division, $C/B \rightarrow C$, is realized in a similar table by interchange of the inputs to A and C during state_3. Overflow may occur, and this may be recognized by detecting a second output on C_0 before there is one on A_0. In the following table this is used to cause the computation to terminate in state_4 when there is overflow.

State	1	2	3	4
Reset	0	A_0	A_0	0
Advance	PB	C_0	C_0	PB
A	0	1	A_B	0
B	0	0	0	0
C	0	1	1	0

Squaring, $B^2 \rightarrow B$, gives the first example of generation of functions more complex than a simple ramp in phase computation. The input to counter B is connected to its own modulator output, A_B, so that the quantity in it increases as a parabola. The table for squaring is shown below :

State	1	2	3
Reset	0	A_0	A_0
Advance	0	B_0	0
A	0	1	1
B	0	1	A_B

The resemblance of this table to that for multiplication is clear, and square-rooting may be obtained by the same transformation that led to division, interchange of the inputs to A and B in state_3.

Many variations on these computations are possible with the same simple configuration. For example, if the modulator output A_C is also available, then a configuration similar to that of Figure 4 may be used to effect vector rotation, and hence polar/rectangular coordinate conversion. In the examples so far, however, the quantities used in computation have been the unsigned fractional binary numbers in the counters. The following section extends the arithmetic operations to quantities taking both positive and negative values.

2.7 *Signed Arithmetic Operations*

A twos-complement representation is used for signed quantities in the counter registers, with the sign bit reversed so that zero quantity corresponds to mid-range of the counter, e.g. 01111...11111 in an N-bit register is the binary fraction $-\frac{1}{2}^{N-1}$. The differentiator outputs corresponding to the state 10 000...00000, A_1, B_1, etc., are used in reading out the sign and magnitude of a quantity using the phase counter principle. For example, consider the counter B in Figure 7 to hold some quantity, β, in (modified) twos-complement form, and counter A to act as its phase reference. In states 3 and 4 of the sequencer A goes through one complete cycle, and the number of clock pulses between A_1 and B_0 coming ON is a measure of the magnitude of β. If B_0 comes ON first then β is positive, whereas if A_1 comes ON first then β is negative.

When a register holding a quantity in twos-complement form is used to modulate the cycling of a counter, then its most significant bit is disregarded in the modulation but used to invert the incremental digital sequence at the modulator output. The output of the modulator modified in this way represents the magnitude of the quantity in the register (labelled A_B for B modulating A), and the sign bit (B_S) is also available to give a sign + magnitude representation.

Computations with signed quantities are more complex than with unsigned quantities, generally involving two or more sequencers to handle the sign information, and there is scope for great variety in setting up similar computations according to the initial and final data formats and locations required. Addition and subtraction are as simple as before, a configuration similar to that of Figure 7 being sufficient. Multiplication of a signed quantity by an unsigned quantity is also comparatively simple, and is described in the following paragraph.

Consider the computation $C \times B \rightarrow C$, where counter C contains a quantity in twos-complement representation and B contains an unsigned constant. It will be assumed that counter A, initially at zero, is available to act as a phase reference to B, and that counter D is available to be modulated by B. Two four-way sequencers, Y to control counter A and Z to control counter C, are required — the inputs to B and D are OFF and

ON respectively. The tables for the two sequencers are :

State	—	Y_1	Y_2	Y_3	Y_4
Advance	—	PB	A_1	$C_o + Z_3$	A_o
A	—	0	1	D_B	1

State	—	Z_1	Z_2	Z_3	Z_4
Advance	—	PB	C_o	$A_1 + Y_3$	A_o
C	—	0	1	D_B	1

Multiplication of two signed quantities may be performed in several alternative configurations — the tables above may be extended to six-way sequencers which include further steps to reverse the sign of the quantity in counter C if the sign bit of register B is set; a third sequencer may be used to provide a sub-routine for sign reversal, only entered if B_S is set; if the computation is $C \times B \rightarrow E$, where E is another counter, then a very simple configuration is possible in which D acts as a phase reference to E.

Division, squaring, vector rotation, and so on, are all readily performed with bi-polar quantities, but the configuration used will vary with the overall computation, since several operations on different registers will generally be performed at the same time. At present there are no algorithms for optimizing a configuration according to the overall effect required, and a major problem to be solved in the future development of the phase computer is to establish such algorithms for use in a hardware/program compiler.

2.8 *Branching, Sub-routines, Interrupts*

The tables of Section 2.7 offer an example of conditional branching in a phase computer program — the state $Y_3 . Z_2$ is entered if A_1 occurs before C_o, whilst the state $Y_2 . Z_3$ is entered if C_o occurs before A_1; it will also be noted that the tables are linked so that $Y_3 . Z_3$ cannot occur. The last three tables of Section 2.6 also offer examples of conditional branching — if A_o and C_o (B_o in the final table) occur together there is an immediate return to state$_1$ without entry of state$_3$. In general branching will occur through a choice of advancing or resetting a single sequencer, or a choice of changing the state of one or more of a set of sequencers.

Sub-routines are realized very simply in the phase computer since the operation of one sequencer is readily controlled by another. Each sub-routine will generally be performed by a single sequencer, but the operation carried out may be varied by changing parameters of the routine (entries in the sequencer table). For example, the following table is a variation on that of Figure 7, in which data transfer may be carried out, with or without sign inversion, from B to C or from C to B :

State	—	U_1	U_2	U_3
Reset	—	0	A_o	A_o
Advance	—	$I + J + K + L$	$B_o + C_o$	0
A	—	0	1	1
B	—	0	$I + K + L$	J
C	—	0	$J + K + L$	I

The control lines, I, J, K, L, when ON, each cause the operation :

$$\begin{aligned} I &— C + B \rightarrow C,\ O \rightarrow B \ ; \\ K &— C - B \rightarrow C,\ O \rightarrow B \ ; \\ J &— B + C \rightarrow B,\ O \rightarrow C \ ; \\ L &— B - C \rightarrow B,\ O \rightarrow C \ , \end{aligned}$$

Hence if sequencer V uses U as a sub-routine and $I = V_i$, $L = V_{i+1}$, and A_o causes advance from both V_i and V_{i+1}, then the net effect of passage through these two states is to use sub-routine U twice to reverse the sign of the quantity in register B.

One feature of the sequencer of Figure 6 which has not yet been discussed is the provision of two sets of outputs. Sequencer outputs are used to control both the inputs to counters (processing elements) and its own inputs plus, possibly, those of other sequencers (programming elements). Advancing and resetting the sequencer must inhibit the operation of controlled processing elements but not that of programming elements; this is apparent in the example of Figure 7. It is also necessary to have the facility of interrupting the operation of a sequencer completely, e.g. when different elements share the same output device, and this is provided through an INTERRUPT line.

2.9 *Input/Output Devices for the Phase Computer*

Since most applications of the phase computer are in real-time data-processing, control and display systems, data input and output has been an important consideration in the design of the computer, and facilities for BCD/binary conversion, and analog/digital conversion, fit naturally into the structure of the machine.

Any counter in the computer is potentially capable of being set directly with a binary number, and also of being read out directly from the condition of its flip-flops. It is more convenient, however, to utilize special interface counters for binary input and output, and transfer from these to the required register by a sub-routine. The internal counters, used in computations, can then have a very simple structure with a minimum of connections. Interface counters having a BCD representation may then be used for automatic decimal $\rightarrow$ binary, and binary $\rightarrow$ decimal, conversion at the input and output of the computer. The laboratory prototype phase computer has an addressing system enabling any of its fourteen internal counters to be reset, set with signed number from decimal keyboard, incremented or decremented by that number, or non-destructively read out in decimal or binary form, signed or unsigned.

Input and output of analog data on many parallel channels is performed in the phase computer by a single digital $\rightarrow$ analog convertor, plus analog track/hold and comparator units for each channel. The D/A convertor generates a ramp by outputting the quantity in a cycling counter in analog form, and this is compared with analog inputs to control the counting of other counters which act as input registers, or sampled by

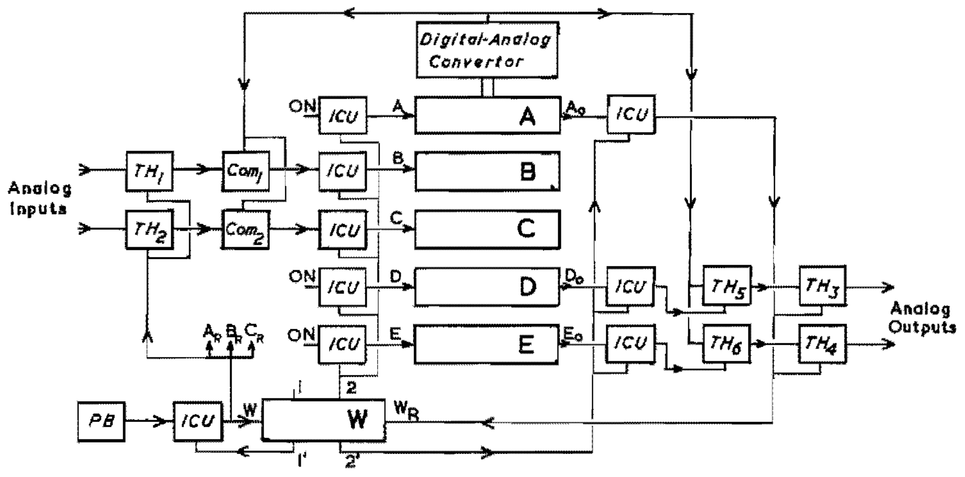


Fig. 8. — Analog-digital interface

track/hold units at the output which are themselves controlled by the counters whose value is to be read out.

Figure 8 illustrates a configuration for reading analog data into two input registers, B and C, and simultaneously reading out, non-destructively, two other registers, D and E, in analog form. Counter A acts as a phase reference to counters D and E, and is also coupled to the D/A convertor to generate a ramp. In state$_1$ of the sequencer none of the counter inputs are forced ON, and, in this state, computations may be carried out with the contents of A, B, C, D and E, under the control of other sequencers.

When the push-button is depressed (or an input signal is received from another source), the analog inputs are sampled by track/hold units, TH_1 and TH_2, counters A, B and C, are zeroed, and the sequencer advances to state$_2$. In this state the inputs to counters A, D and E are ON, whilst those to B and C are connected to comparators, each of whose outputs is ON if their analog input is greater than the output of the D/A convertor (they are bound to be ON initially if the analog inputs are within range). As counter A cycles the output of the D/A convertor increases and will eventually become greater than each of the analog inputs. When it does so the comparator output of that channel goes OFF and the corresponding counter stops counting — analog $\rightarrow$ digital conversion is then complete. As D and E cycle their differentiator outputs, D_0 and E_0, will come ON and cause the D/A convertor output to be sampled and held in TH_5 and TH_6 respectively. When, finally, counter A returns to zero and A_0 comes ON, the sequencer is reset to state$_1$, and the analog values stored in TH_5 and TH_6 are transferred to the output channels, TH_3 and TH_4, respectively — digital $\rightarrow$ analog conversion is then complete.

3. Summary and Conclusions.

The phase computer is a modular, all-digital, data-processing and control system, in which many of the advantages of analog and digital computers are combined through the use of an incremental digital processor under program control. The incremental processor enables pseudo-analog computing loops to be established so that complex operations, such as division, square-rooting, rectangular/polar/hyperbolic co-ordinate transformations, and so on, are performed with the same speed and simplicity as simpler operations, such as addition and data transfer. The programming facility enables complex computations to be performed as a sequence of elementary operations, and hence overcomes the inordinate hardware demands of previous parallel incremental computers.

The speed of the incremental processor in performing simple operations is less than that of a parallel digital processor because of the counting technique used, and the disparity becomes greater with increasing precision in computation. This factor is very much decreased in computations where a number of complex operations may be carried out at the same time. At present, with a 4 Mcs clock frequency, operations can be carried out with 10-bit precision and accuracy (using the predictive modulator) in times of about 250 microseconds, and with 12-bit accuracy in about 1 millisecond. These speeds and accuracies have proved to be ample for a large class of data-processing and control applications, and enable advantage to be taken of the economy of phase computing.

There are only five basic modules in the phase computer : counters, modulators, input/control units, sequencers and differentiators. These may be fabricated with standard integrated circuits and utilize fully the high packing-density devices, such as four-bit counters and quad-adders. By their simplicity and uniformity of structure, and their few interconnections, the modules and module-systems are eminently suitable for large-scale integration.

Applications investigated to date, in which the use of phase computation offers considerable advantages in cost and size, include : area coverage systems for aircraft navigation systems based on VOR/DME beacons; sweep generators for PPIs in surveillance radar systems; marker/vector generators with rectangular/polar co-ordinate conversion facilities in PPI consoles; automatic tracking of targets by least-squares smoothing and prediction for small radar systems; identification of process parameters for purposes of adaptive control, and on-line analysis of auto-analyser data for biochemical assay.

Apart from its intrinsic advantages, the phase computer is also of interest as an example of the multi-processor hybrid computing configurations discussed in the first section of this paper. It is suggested that the full exploitation of the tremendous advantages in economy, reliability and small-size, offered by the advent of large scale integration of micro-circuits, depends upon the further development of computing techniques which maintain a balance between hardware and software. Not only are many of the problems in the implementation of on-line computer systems at present due to undue reliance on software modification of standard computers, but also the proportionate cost of the computer hardware in these systems is so low that little advantage can be taken of the rapidly decreasing cost of integrated circuits.

It should be possible, at the present state of knowledge, to dispense with arbitrary distinctions between analog, digital, hybrid, incremental digital, and so on, computers, and treat the technology of data-processing and control in a unified and coherent manner. The major problems to be solved are the development of small families of universal, hybrid computing modules suitable for large scale integration, and the development of problem-orientated languages with associated compilers to specify both hardware and program based on these modules.

3.1 *Acknowledgements*

We are grateful to many colleagues in ITT for discussions and suggested applications of the phase computer — in particular to Mr Kjell Mellberg of SRT, Stockholm, and Mr R.A. Shemer of STL, for their many contributions.

REFERENCES

[1] KORN, G.A., KORN, T.M. — *Electronic Analog and Hybrid Computers,* McGraw Hill, 1964.

[2] SCHMID, H. — Sequential Analog-Digital Computer (SADC), AFIPS FJCC *27*(1) 1965, 915.

[3] CLANCY, J.J. — Digital Simulation Languages : A Critique and Guide, AFIPS FJCC *27*(1) 1965, 23.

[4] OWEN, A., BELL, D., GRIFFIN, A. — Help — A Hybrid Computer High Level Language Compiler and Operating System, 5th Int. Congr. AICA 1967.

[5] MAYOROV, F.V., CHU, Y. — *Digital Differential Analysers,* Iliffe Books, 1964.

[6] FORBES, G.F. — *Digital Differential Analysers,* 1953.

[7] GRABBE, E.M. — *Handbook of Automation and Control Vol. II,* Wiley, 1959.

[8] ANON. — Digital-Operational Techniques, Computer Design, November 1963, 12.

[9] GRANDINE, J.D., HAGAN, T.G. — A Parallel/Sequential, Stored Program Hybrid Signal Processor, Simulation, Jan. 1965.

[10] HAGAN, T.G., TREIBER, R. — Hybrid Analog/Digital Techniques for Signal Processing Applications, AFIPS SJCC *30* 1967.

[11] GAINES, B.R., Stochastic Computing, AFIPS SJCC *30* 1967.

[12] GAINES, B.R., Techniques of Identification with the Stochastic Computer, IFAC Symposium on Identification, 1967.

[13] GAINES, B.R., Stochastic Computers, in *Encyclopaedia of Information, Linguistics and Control,* Pergamon Press 1967.

[14] SIMPSON, H.R. — A Method of Processing Radar Data to Obtain Position, Velocity and Turn Information, RRE Memo. 1924, 1962.

[15] ROSENBERG — Automatic Machine Tool Control, in *Computer Control Systems Technology,* McGraw Hill, 1961.

[16] SCHMID, H. — An Operational Hybrid Computing System, IEEE Trans. EC *12* 1963.

[17] YANG, H.Z. — Determination of Maximum Error of a Binary Multiplier, Automation and Remote Control *21*(7) 1961, 709.

[18] TEXAS INSTRUMENTS — Type SN7483N 4-Bit Binary Full Adder, New Product Bulletin SC-9469 November 1966.

[19] BRADLEY, R.E., GENNA, J.F., Design of a One-Megacycle Iteration Rate DDA, AFIPS SJCC *21* 1962.