INFORMATION PROCESSING 68 - NORTH-HOLLAND PUBLISHING COMPANY - AMSTERDAM (1969)

A MODULAR PROGRAMMED DDA FOR REAL-TIME COMPUTATION

B. R. GAINES
Department of Electrical Engineering Science, University of Essex, Colchester, Essex, U.K.
and
Standard Telecommunication Laboratories Ltd., London Road, Harlow, Essex, U.K.

This paper presents a modular computing system in which DDA techniques are combined with a hardware program structure to realise a versatile family of special-purpose machines amenable to LSI. For real-time navigation and control problems the pseudo-analog computing loops of the DDA offer simple and direct means of performing the complex computations involved in multi-dimensional co-ordinate conversion, estimation and adaptive control.

1. INTRODUCTION

The increasing availability of low-cost integrated circuits and the even lower costs predicted from LSI are causing a new look to be taken at special-purpose computers and in particular at incremental digital systems such as the DDA [1-5]. For real-time navigation and control problems the pseudo-analog computing loops of the DDA offer simple and direct means of performing the complex computations involved in multi-dimensional co-ordinate conversion and adaptive control.

Previous interest in DDA-like machines has been sporadic and the literature is slight and very variable in accuracy. High quality work on DDAs has taken place largely in classified areas and the machines themselves have never gathered the universal following of the stored-program general-purpose computer. All early DDAs and most commercially viable ones have employed a rotating mass memory to hold the registers with a very simple processor to update them one by one [6,7]. A few parallel machines have been proposed and constructed [8-10] but there is no established organisation which may be called *the* parallel DDA.

Thus the present surge of interest in DDAs under the influence of LSI has little direction from prior art and must establish its own system rationale. In this much may be learnt from the organisation of the CPU in a conventional general-purpose computer. Here complex operations are generated as a branching *sequence* of very simple basic operations in the arithmetic unit [1]. The whole of a DDA may be regarded as a single arithmetic unit with a capability of generating functions far more complex than ADD, SHIFT, COMPLEMENT, as in the normal arithmetic unit, but these functions are generally not varied in the computation. Thus the DDA and GP computers stand at opposite ends of a continuum - the one decoding a problem statement into a single complex operation, the other decoding it into a multitude of minor operations specified by a program.

The optimum approach to a special-purpose computer for a class of problems would seem to be between that of a GP machine and a DDA. If the basic computational operations common to all the problems (such as vector rotation in navigation) are made the basic operations of the computer's arithmetic unit, and are programmed in sequence for a particular computation, then the computer will be simple to use and the length of its programs will be minimised. In a truly general-purpose system the universal operations are necessarily simple and basic, but for a special-purpose such as navigation they can become more complex.

This paper presents a modular computing system in which DDA techniques are combined with a hardware program structure (most similar to GP micro-programs) to realise a versatile family of special-purpose machines amenable to LSI. The particular machine described, the Phase Computer [12], is only one of the many possible configurations - the DDA section in particular is extremely simple - but it serves to illustrate both the structure and potential applications of special-purpose data processors based on a simple set of computing modules.

2. THE PHASE COMPUTER

The particular problem which first stimulated the development of the phase computer was the realisation of the least-squares smoothing and prediction equations for automatically tracking radar targets [13, 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, or economically feasible, and the simplest and cheapest special-purpose equation solver was required.

The tracking equations involve addition, subtraction, multiplication, division, squaring, square-rooting, sine/cosine generation and inverse sine/cosine resolution, so that they encompass a very wide range of arithmetic operations. They may be regarded as Kalman-Bucy equations for optimum system identification [15] and hence any system for their solution has immediate relevance to adaptive control problems. The data rate in both surveillance radars and control systems is fairly low (typically between 0.1 and 100 samples/second) and incremental computing techniques offered the possibility of solving the equations with very simple hardware. Since the data was already in sampled form, it was attractive to decrease the cost still further and add to the flexibility of the computer by executing the computations as a sequence of simpler operations.

Apart from its sequential programmed operation, the phase computer has one other feature not found in previous parallel DDAs, and that is the use of unidirectional counters throughout the computer rather than the bi-directional, up/down counters of the DDA. This feature, which not only decreases the cost but also greatly simplifies the operation, is made possible by use of the 'phase counting' principle in which stored quantities are represented as the *difference* in the counts contained in two counters. Incrementing both counters does not change the stored value - incrementing one and not the other either increases or decreases the stored value, according to which counter is taken positively (the Store) and which negatively (the Reference). The use of this technique is not uneconomic in hardware, because one counter generally acts as Reference to a number of others.

A block-diagram of the phase computer is shown in fig. 1. It has a conventional four-part structure, consisting of arithmetic unit, program sequencer, auxiliary storage, and input/output channels, but the arithmetic unit is incremental

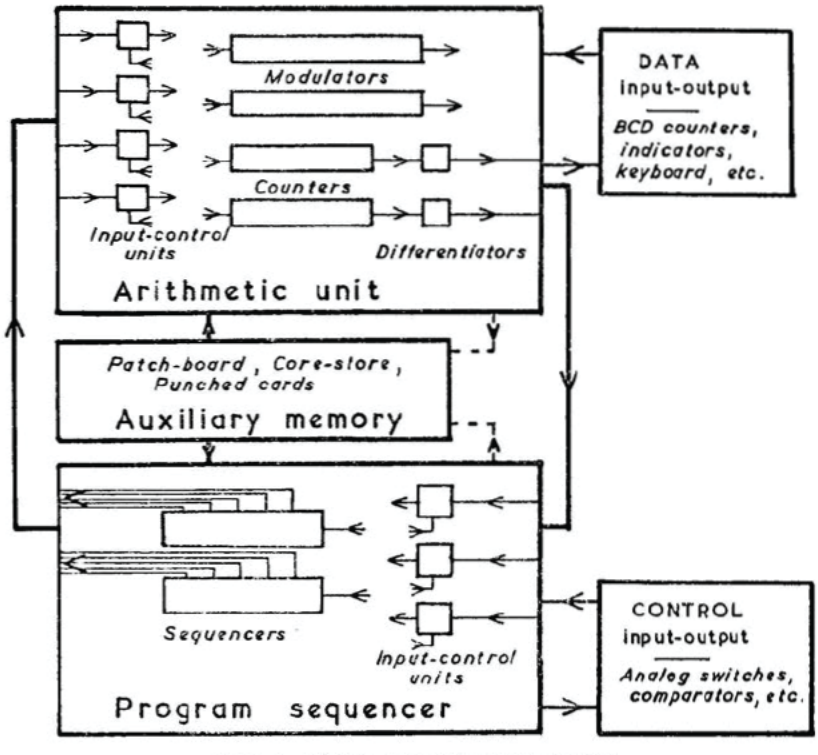


Fig. 1. Phase computer schematic.

digital, the sequencing section is very extensive and may operate several concurrent programs, and the auxiliary storage in the laboratory prototype machine is realised entirely through a patch-board.

3. COUNTERS AND MODULATORS

The active storage registers of the phase computer are undirectional 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 fig. 2.

If the proportion of ON logic levels on the INPUT line of the 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 form of its input - the integral is available as a binary number rather than an incremental sequence.

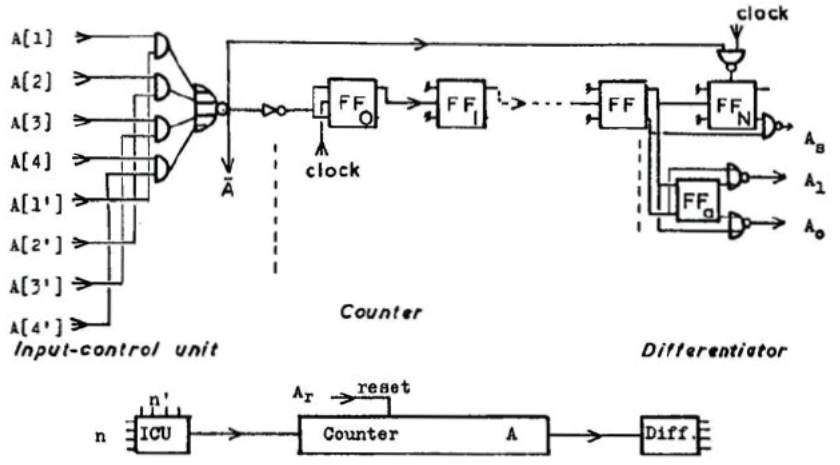


Fig. 2. Counter, control and state-detection.

An incremental sequence in which the proportion of ON logic levels is equal to the fractional binary number stored in a counter is obtained, as shown in fig. 3, from a modulator element which adds (through full-adders) the number in the counter to that in a register (*D*-type flip-flops) and outputs the overflows. In order to maintain accuracy when the count in the counter is changing, the quantity added into the register is the mean of the present count and the predicted next count (open trapezoidal integration). The modulator has input lines for controlling additions to the register and for resetting it.

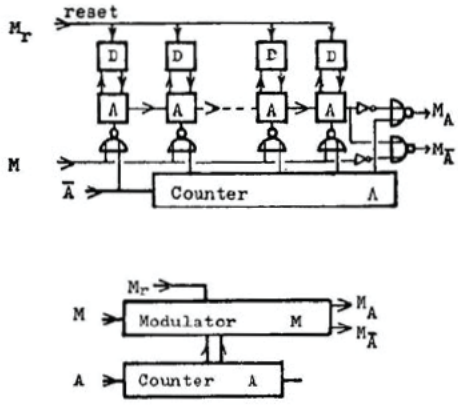


Fig. 3. Modulator.

4. SEQUENCING COMPUTATIONS

By connecting the outputs of modulators to the inputs of counters, pseudo-analog computing loops may be set up in the arithmetic unit of the phase computer and used to generate operations which are difficult to realise with parallel digital arithmetic. In general, however, the loops used will be far simpler than those of conventional DDAs, the majority of counters generating ramps rather than nonlinear functions, and the basic computational operations realised in this way, such as multiplication, division, addition and subtraction, 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

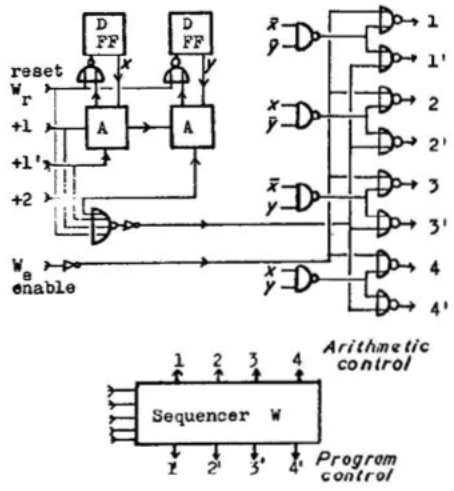


Fig. 4. 4-way sequencer.

processor elements by logic levels on CONTROL lines. The input-control unit, shown in fig. 2, is a gate enabling processor connections to be controlled - it implements the function:

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

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. In some applications two, or more, CONTROL lines may be required for each input.

Input-control units placed at the inputs of phase computer counters enable them to be interconnected with modulators in several alternative patterns, any one of which may be selected by activating the appropriate CONTROL lines. A sequencer unit, shown in fig. 4, is used to activate the CONTROL lines corresponding to different interconnection patterns in a programmed sequence. It is made up of a small register, to each of whose states there corresponds an OUTPUT line which is ON when the register is in that state. An adder and gating enables the state of the sequencer to be reset to zero or incremented by any amount (generally unity for normal operation and two for a branch), reset taking precedence over incrementing.

The signals causing the sequencers to change their states, corresponding to steps and branches in the program, are obtained from digital 'differentiators' which give an output for the clock pulse following that in which the counter enters a certain state - in particular, the zero (A_0), and mid-range state (A_1). The outputs of the differentiators are used to change the states of sequencers when certain counters accumulate pre-determined counts during a computation.

The use of these various elements and their interconnection to perform computations is best made clear by example, and the following section describes phase computer configurations for a wide variety of basic arithmetic operations.

5. UNSIGNED ARITHMETIC OPERATIONS

A small phase computer processor might consist of three counters, A, B, C, one or two modulators, L, M, a 3-way sequencer, W, and associated input-control units and differentiators. The sequencer would be used to switch the inputs to counters either ON or OFF, or to the outputs of modulators, and the connections associated with each state of the sequencer can be specified by the signals at the counter inputs. The outputs from the differentiator will be used to change the state of the sequencer or reset counters, and this program control can be specified by the changes that may occur, and their causes, in each state of the sequencer.

Thus any particular computation may be specified completely by a table of the following form:

Sequencer -	State -	W_0	W_1	W_2
	Reset -	0	A_0	A_0
	Advance -	P	B_0	0
Counters -	A input -	0	1	1
	B input -	0	1	1
	C input -	0	0	1
	C reset -	P	0	0

This table corresponds to data-transfer, or reproduction of the quantity stored in counter B in counter C, assuming that counter A initially is at zero and that the computation is initiated by a pulse, P. In the zeroth state of the sequencer, W_0, all counter inputs are OFF - this is the passive state. The pulse P resets counter C to zero and advances the sequencer to its first state W_1 in which the inputs to counter A are both ON (written as '1'). When counter B overcounts to zero, the differentiator output B_0 causes the sequencer to advance to state W_2 in which the input to counter C is also ON. Finally the sequencer returns to the passive state W_0 when counter A overcounts back to its initial state of zero.

This computation illustrates the phase counter principle, for if the quantity stored in B is taken to be the difference in counts between counter A and counter B then this is invariant provided the

inputs to the two counters always receive the same signal - it will be noted from the table that this is so. Since counter A starts and finishes in zero, it has no initial or final effect on the stored quantity which may be taken as the fractional binary number in B. The table may be abbreviated:

	W_0	W_1	W_2
W_γ	0	A_0	A_0
$W(+1)$	P	B_0	0
A	0	1	1
B	0	1	1
C	0	0	1
C_γ	P	0	0

Data transfer - $B \rightarrow B,\ C$

If the initial resetting of C is omitted, then this becomes addition of the quantity in B to that in C:

	W_0	W_1	W_2
W_γ	0	A_0	A_0
$W(+1)$	P	B_0	0
A	0	1	1
B	0	1	1
C	0	0	1

Addition - $B \rightarrow B,\ B+C \rightarrow C$

Subtraction is a simple variant on this in which C counts during W_1 rather than W_2:

	W_0	W_1	W_2
W_γ	0	A_0	A_0
$W(+1)$	P	B_0	0
A	0	1	1
B	0	1	1
C	0	1	0

Subtraction - $B \rightarrow B,\ C-B \rightarrow C$

If C initially contains γ, since the count added whilst the sequencer is in W_1 is $1-\beta$, the final total in counter C is $\gamma + (1-\beta) \equiv \gamma - \beta$.

For multiplication, B is connected to the modulator, M, and its output is OFF throughout the computation so that its state does not change. The output of M, M_B, is a sequence of logic levels in which the proportion of ON levels is equal to β, the fractional binary number in B. This is connected to the input of C between C entering zero and A returning to zero, a period proportional to the quantity in C, γ - so that the final quantity in C is $\gamma\beta$:

	W_0	W_1	W_2
W_γ	0	A_0	A_0
$W(+1)$	P	C_0	0
A	0	1	1
B	0	0	0
C	0	1	M_B
M	0	0	1
M_γ	P	0	0

Multiplication - $B \rightarrow B,\ C \times B \rightarrow C$

Division is realised in a similar fashion, except that the connections to A and C are reversed, so that during the sequencer state W_2, the quantity γ has to be supplied to A from the source M_B, and hence the quantity transferred to C during this period is γ/β:

	W_0	W_1	W_2
W_γ	0	A_0	A_0
$W(+1)$	P	C_0	0
A	0	1	M_B
B	0	0	0
C	0	1	1
M	0	0	1
M_γ	P	0	0

Division - $B \rightarrow B,\ C/B \rightarrow C$

Squaring the quantity in B offers the first example of a modulator being used to generate a nonlinear function through connection to a changing counter. The output from the modulator M during W_1, whilst B counts from its initial value up to zero would be:

$$\int_{\beta}^{1} u\,du = (1-\beta^2)/2\ ,$$

if the modulator and counter inputs were ON

together. By connecting the counter input to a source whose proportion of ON logic levels is $\frac{1}{2}$ (a toggling flip-flop), the number of clock intervals for B to return to zero is doubled, and so is the above integral. This quantity is counted into A, so that at the end of W_1, A contains $1-\beta^2$, and the remainder to be counted in during W_2 is β^2, which also enters B:

	W_0	W_1	W_2
W_γ	0	A_0	A_0
$W(+1)$	P	C_0	0
A	0	M_B	1
B	0	$\frac{1}{2}$	1
M	0	1	0
M_γ	1	0	0

Squaring - $B^2 \rightarrow B$

Taking the square root of the quantity in B is performed in a similar fashion except that inputs to counters and modulator are interchanged between sequencer states W_1 and W_2. The net effect is that the quantity transferred to A to bring it back to zero during W_2 is both equal to β, the initial quantity in B, and to the square of the final quantity in B, λ^2 say - hence $\lambda = \gamma\beta$:

	W_0	W_1	W_2
W_γ	0	A_0	A_0
$W(+1)$	P	B_0	0
A	0	1	M_B
B	0	1	$\frac{1}{2}$
M	0	0	1
M_γ	P	0	0

Square root - $B^{\frac{1}{2}} \rightarrow B$

Rectangular/polar co-ordinate conversion requires an additional modulator, L, coupled to counter A. The vector (x, y) in Cartesian co-ordinates, whose components are stored in counters C and B respectively, is to be converted to (r, θ), polar co-ordinates stored in the same locations. First, during W_1, $1-x$ is transferred to A, and then counters A and B are cross-coupled with the modulator output M_B at the input of A, and the inverted modulator output $L_{\bar{A}}$ at the input of B. This may be represented by the differential equations:

$$dy/dt = x$$

$$d(1-x)/dt = y$$

so that

$$\ddot{x} + x = 0 ,$$

corresponding to rotation of the vector (x, y). This ceases when A returns to 0, in which event the x-component has become zero and the y-component in B is equal to $r = (x^2+y^2)^{\frac{1}{2}}$. The quantity accumulated in counter C during W_2 is proportional to the angle of rotation [2], θ, in units such that one right-angle corresponds to unity in the counter, and the conversion from radians is accomplished by counting from a source in which the proportion of ON logic levels is $2/\pi$ (obtained from a modulator with constant, hardwired input):

	W_0	W_1	W_2
W_γ	0	A_0	A_0
$W(+1)$	P	C_0	0
A	0	1	M_B
B	0	0	$L_{\bar{A}}$
C	0	1	$2/\pi$
L	0	0	1
M	0	0	1

Rectangular/Polar Co-ordinate Conversion

$(B^2+C^2)^{\frac{1}{2}} \rightarrow B$, $\tan^{-1}(B/C) \rightarrow C$

6. SIGNED ARITHMETIC AND DATA INPUT/OUTPUT

There is no scope in the present paper to describe signed arithmetic operations and input/output of numeric and analog data in the phase computer. Briefly, signed quantities are represented in twos-complement form and the branching capability of the sequencer is used to take into account the signs of quantities in particular computations. Input/output of binary coded decimal data is simple in the phase computer through the use of BCD rather than straight binary counters at the interface, and input and output of analog data on many channels is performed by a single digital/analog converter coupled to a counter together with track-hold and comparator units for each channel.

7. SUMMARY AND CONCLUSIONS

The phase computer is a modular, 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 arithmetic unit 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, may be 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 DDAs.

Applications investigated to date include: area coverage systems for aircraft navigation based on VOR/DME beacons; direct readout of latitude and longitude for hyperbolic marine navigation based on differential transmission paths; sweep generators for PPIs in surveillance radar systems; marker/vector generators with rectangular/polar co-ordinate conversion facilities in PPI consoles; automatic tracking of radar targets by least-squares smoothing and prediction for small radar systems; identification of process parameters for purposes of adaptive control; and on-line product quality monitoring with reject-rate prediction.

ACKNOWLEDGEMENTS

I am grateful to many colleagues in ITT for discussions and suggested applications of the phase computer - in particular to Mr. P. L. Joyce of STL who built the first laboratory prototype machine.

REFERENCES

[1] L. C. Hobbs, Effects of large arrays on machine organisation and hardware/software tradeoffs, AFIPS FJCC 29, 1966.
[2] J. Hatvany, The DDA integrator as the iterative module of a variable structure computer, IFAC International Pulse Symposium, Budapest, April 1968.
[3] R. A. Shemer, A hybrid-mode modular computing system, IFAC International Pulse Symposium, Budapest, 1968.
[4] R. W. Ward, Shrinking a military calculator, Electronics, May 29, 1967.
[5] B. R. Gaines, The place of incremental techniques in future computer systems, IEE Colloquium on Integrated Circuits in Logic Design, London, November 1967.
[6] F. V. Mayorov and Y. Chu, Digital differential analysers, Iliffe, 1964.
[7] G. F. Forbes, Digital differential analysers, 1954.
[8] R. E. Bradley and J. F. Genna, Design of a one megacycle iteration rate DDA, AFIPS SJCC 21, 1962.
[9] J. M. Mitchell and S. Ruhman, The TRICE - A high speed incremental computer, IRE Nat. Conv. Rec. Pt 4, 1958.
[10] M. W. Goldman, Design of a high speed DDA, AFIPS FJCC 27 (1965) 1.
[11] B. R. Gaines, Varieties of computer - Their interrelationships and applications, IFAC International Pulse Symposium, Budapest, 1968.
[12] B. R. Gaines and P. L. Joyce, Phase computers, Proc. 5th Int. Congr. AICA, Lausanne, 1967.
[13] H. R. Simpson, A method of processing radar data to obtain position, velocity and turn information, RRE Memo. 1924, 1962.
[14] K. Mellberg, Experience from automatic tracking in ATC, IEE Conf. Publ. no. 28, Symposium on ATC Systems.
[15] R. E. Kalman and R. S. Bucy, New results in linear filtering and prediction theory, Trans. ASME, March 1961.