Process Performance Computer for Adaptive Control Systems

FRANK A. RUSSO AND ROBERT J. VALEK

Abstract—Adaptive control of complex manufacturing processes can provide significant improvements in production rate and product quality. To maximize these advantages, the adaptive control system must be capable of making accurate measurement of process performance. This paper describes the implementation of a general-purpose performance computer which makes use of a combination of two techniques: trainable pattern recognition and linear regression. The description covers the system concept, the training procedures or algorithms, and the detailed design.

Index Terms—Adaptive control system, linear regression, on-line performance measurement, performance computer, threshold logic, trainable pattern recognition.

Introduction

In complex manufacturing processes, adaptive control provides means of decreasing cost, increasing production rate, and reducing scrap [1]–[3]. In the adaptive control concept, on-line measurements are made of system performance and these measurements are used to generate auxiliary control signals which adjust controllable variables to obtain optimum performance.

In implementing a practical system, the degree of improvement in manufacturing efficiency is closely related to the availability of accurate and economical on-line performance measurement equipment. Although on-line performance measurement of complex processes is a very difficult problem, recent advances in data-processing theory and hardware implementation have made techniques available which are believed to enable a reasonable solution to the performance measurement problem. This paper discusses the application and implementation of these techniques in a general-purpose performance computer.

An adaptive control system for a machine tool is illustrated in Fig. 1. As shown in the figure, the machine tool is controlled by two feedback loops: the primary, or conventional, feedback loop and the secondary, or adaptive, feedback loop. In the conventional loop, a numerical control system receives information from paper tape regarding part dimensions, feed rate, and spindle speed. Based on this information, the numerical control system generates tool/workpiece position and spindle speed commands which it supplies to the servos of the machine tool. Actual speed and position feedback data are returned to the numerical control system for comparison with the commands. This control loop would normally cause the machine tool to follow a programmed path at some programmed feed rate and spindle speed. Under adaptive control, however, the controllable variables of the machine tool, feed rate, and spindle speed are modified on-line to maximize productivity in spite of unpredictable variations in the machine tool or the workpiece.

As illustrated in Fig. 1, the elements in the adaptive feedback path are the adaptive controller and the performance computer. The performance computer monitors the cutting process by means of sensors coupled directly to the machine tool. These sensors supply electrical signals indicating the instantaneous values of measurable process parameters, such as cutting temperature, spindle torque, and spindle vibration. Based on the sensor outputs, a set of performance indicators which characterize the process operation is computed. These indicators are then used directly by the adaptive controller as the basis for modifying the controllable variables of the process. For example, in a metal-cutting operation, the performance indicators are tool-wear rate and surface finish. The adaptive controller would modify the controllable variables of feed rate and spindle speed to maintain the desired surface finish and simultaneously minimize the tool-wear rate.

For most industrial processes, the performance indicators are usually not directly measurable on-line, but can only be determined off-line some time after the process operation. For example, the rate of tool wear can only be determined off-line by before-and-after measurements of tool wear. Similarly, the weld depth in a welding process must be determined off-line by destructive testing. Generally, the performance indicators are related to measurable process parameters by complex nonlinear functions.

A performance computer should therefore be capable of implementing nonlinear functional relationships between measurable parameters and performance indicators for a variety of manufacturing processes. Design

Manuscript received April 5, 1968; revised July 17, 1968. This paper was presented at the First Annual IEEE Computer Conference, Chicago, Ill., September 6–8, 1967. The general-purpose performance computer described here was developed under the direction of W. Webster of the Advanced Fabrication Techniques Branch, MATF, Manufacturing Technology Division, Air Force Materials Laboratory, Wright-Patterson Air Force Base, Ohio, under Contract AF 33 (615)-2634.

The authors are with Bendix Research Laboratories, Southfield, Mich. 48075.

1 In some cases prior knowledge of the process permits the use of fixed control strategies; in many processes, the adaptive controller would require a sophisticated search strategy.
of the performance computer is inherently a difficult
problem because the relationships needed to compute
performance often cannot be reduced to a specific
mathematical form. Furthermore, to be a practical com-
ponent in an adaptive control system, production
models of the computer should cost less than conve-
tional stored-program control computers. The perfor-
ance computer to be discussed meets all of these speci-
fications by combining the flexibility of a stored-program
computer with the efficiency of a fixed-wire device. The
flexibility is achieved by the use of a trainable logic
approach which permits the performance computer to
work with any of a number of diverse industrial pro-
cesses. Low cost is maintained not only by the wired
program organization but also by the use of an off-line
training feature.

SYSTEM CONCEPT

The system concept is based on a combination of two
techniques: pattern recognition and linear regression.
In the linear regression technique, a performance indi-
cator $H_c$ is computed as a weighted summation of the
measured parameters. That is

$$H_c = \sum_{M=0}^{N} W_M P_M$$

where the $P$'s are the measured parameters and the $W$'s
are the weighting coefficients (generally $P_0 = 1$). Because
of the nonlinear relations that exist in most industrial
processes, it is unlikely that the above single linear
regression equation will be accurate over all operating
conditions of the process. For example, consider the
hypothesetical relationship shown in Fig. 2 between
the measured parameters $P_1$ and $P_2$ and the perfor-
ance indicator $H$. In the figure, $H$ is plotted perpendicular
to the paper, but is represented in the plane of the paper
by contour lines of constant $H$. The hypothetical surface
can be visualized as a hill. To approximate this surface
with linear regression equations, one equation is needed
for each class as defined by the dashed lines in the figure.
For example, in Class I a linear regression equation
would take the form

$$H = W_1P_1 + W_2P_2$$

where $W_1$ and $W_2$ are positive; in Class III the form
would be

$$H = W_0 - W_1P_1 - W_2P_2$$

where again the $W$'s are all positive. Thus, the surface
of the hill may be visualized as being approximated by
a four-sided pyramid. In the performance computer,
selection of the appropriate equation is accomplished by
means of a trainable pattern recognition network as
described below.

A simplified block diagram of the general-purpose
performance computer is shown in Fig. 3. The measured
parameters for a given operating condition are first ex-
amined by the pattern recognition network and the
operating condition is classified. Based on the assigned
class, the weighting coefficient selector selects an appro-
priate set of predetermined weighting coefficients; in
effect, it selects an equation from a set of stored equa-
tions. Finally, these coefficients together with the values
of the measured parameters are used in the weighting
and summation subsystem to compute the desired per-
formance indicator. The coefficient selector and the
weighting and summation subsystems are implemented
with standard decoding and arithmetic logic networks;
a detailed discussion of the pattern recognition network
follows.
As shown in Fig. 4, the pattern recognition subsystem is composed of a parameter quantizer and a pattern processor. The parameter quantizer is essentially a filter which extracts significant features from the set of input parameters and generates a binary pattern. The quantizer encodes the values of the various parameters according to predetermined quantization levels. The quantization levels are the values of the parameter at which the bits of the pattern representing the parameter undergo transitions. That is, the $F_j$ bit of the pattern is given by

$$F_j = 0 \quad \text{if} \quad P_i < Q_j$$

$$F_j = 1 \quad \text{if} \quad P_i \geq Q_j$$

where $Q_j$ is a quantization level for measured parameter $P_i$. The quantization levels are statistically determined for a given set of measured parameters so that they produce an output pattern which contains a maximum of information indicative of the actual class of the performance indicator, $H_c$. Referring to Fig. 2, optimum quantization levels would be $P_1 = A$ and $P_2 = B$. The output pattern for Class II, for example, would be $F = 1, 0$.

The pattern processor performs the recognition and classification of the operating condition on the basis of the binary pattern. The processor is a network of several nonlinear threshold logic circuits. Threshold logic circuits are circuits which can be adapted or "trained" to perform required logic functions by adjustment of internal parameters.

To introduce this type of logic circuit, a description of a more widely known circuit called linear threshold logic is presented. Fig. 5(a) shows the configuration of a linear threshold logic circuit. The binary inputs to the circuit, $F_0, F_1, \ldots, F_N$, are, for convenience, treated as $-1, +1$ rather than the usual $0, 1$. The input $F_0$ is assumed to have a constant $+1$ value, thus giving the quantity $A_0$ the property of a threshold. Associated with the binary inputs are algebraic weights, $A_1, A_2, \ldots, A_N$ which can have negative or positive values. Each binary input is multiplied by its corresponding weight, and the sum of the weighted inputs is quantized to form the circuit output $G$ in accordance with the following rule.

If

$$\sum_{n=0}^{N} A_n F_n \geq 0$$

then

$$G = 1;$$

otherwise

$$G = -1.$$ 

Thus, the binary output is a function of the binary inputs and the algebraic weights; that is

$$G = G(F_{N+1}, A_{N+1})$$

where

$$F_{N+1} = (F_0, F_1, F_2, \ldots, F_N)$$

represents the input pattern, and

$$A_{N+1} = (A_0, A_1, A_2, \ldots, A_N)$$

represents the weight values. To use linear threshold logic circuits to perform pattern classification, it is necessary to find a set of weight values $A_{N+1}$, which then classify the set of input patterns $F_{N+1}$. As described
previously, the linear threshold logic circuit can separate a set of input data into two classes by a +1 or a −1 output. If more than two classes are desired, several such circuits can be connected in a network as required. In this case, it is necessary to find a set of weights \( A_{N+1} \) for each threshold circuit. The determination of the weights (commonly called training of the circuit) is accomplished by incremental adjustment of the weights and the threshold value, and methods for adjusting these weights are designated as training algorithms. Several such algorithms have been described in the literature [4]–[6].

A limitation of linear threshold logic for large numbers of variables is that a single circuit can realize only a small percentage of the total number of functions; furthermore, only linearly separable functions can be realized. To alleviate this problem, the nonlinear threshold logic circuit shown in Fig. 5(b) has been developed [7]. This circuit differs from the linear threshold logic circuit only in the presence of the encircled elements marked NL (nonlinear). These elements form nonlinear binary-valued functions \( \beta_1, \beta_2, \ldots, \beta_m \) of the input variables. The number of nonlinear functions can be less than, equal to, or greater than the number of input variables. The output of the circuit is based upon the weighted sum of the \( \beta \) values according to the following rule.

If

\[
\sum_{m=0}^{N} \beta_m A_m \geq 0
\]

then

\[
G = +1;
\]

otherwise

\[
G = -1.
\]

The \( \beta \) terms thus take the place of the \( F \) terms in the linear portion of the circuit.

In theory, virtually any form of nonlinear function could be used in a logic circuit of this type. The nonlinear function used in this application is the cross-product, or polynomial, defined as

\[
\beta_m = \frac{\pi}{n:s_m^n = 1} F_m.
\]

This function was chosen since in many pattern-recognition problems the Exclusive-OR operation is needed and is easily realized with cross-product terms. In this case, \( \beta_m \) is equal to the product of certain \( F_m \) terms which are selected by the switching matrix \( s_m^n \). In general, \( \beta_m \) will be equal to +1 when there is an even number of −1 variables for which

\[
s_m^n = 1,
\]

and \( \beta \) will be equal to −1 when there is an odd number of −1 variables for which

\[
s_m^n = 1.
\]

The total number of logic functions which can be realized by a cross-product threshold logic circuit is dependent only upon the number of cross-products used in the circuit. However, the ability of the circuit to realize a given function is directly related to the specific cross-products used in the circuit. Consequently, if the cross-products were fixed, the circuit capacity would be limited to only certain logic functions. Although a circuit having \( 2^N \) cross-products could realize every function of the \( N \) variables, such a circuit would be impractical to build for large values of \( N \). A solution to this problem is to use a circuit with cross-products which can be varied by the training algorithm. In effect, the training algorithm for such a circuit is required to adjust the \( s_m^n \) switching parameters (in addition to the weights and threshold) in order to select the particular cross-product functions that are best suited suited to the problem at hand. Conceptually, the adjustment of the switching parameters may be thought of as the opening and closing of actual switches placed between the input variables and the cross-product operators.

To conserve hardware in the prototype computer, the circuits are trained off-line, using a general-purpose computer. The training algorithm that has been developed for the nonlinear threshold logic circuit is described in the next section. With this algorithm and the nonlinear threshold logic circuit described above, any logic function can be realized; the approach is not limited to linearly separable functions as is the case with linear threshold logic functions. Experimental tests with input patterns of 30 bits show that 20 cross-product terms are generally adequate.

**Training Programs**

To apply the performance computer to a given application, values for four types of training constants are required.

1) Weighting Coefficients, \( W \), for the linear regression equations.
2) Quantization levels, \( Q \), for converting input parameters into binary patterns.
3) A switch specification matrix, \( S \), for controlling the cross-product terms in the nonlinear stage of the threshold logic circuits.
4) Weights, \( A \), for the linear portion of the threshold logic circuits.

These training constants are determined by digital computer analysis of experimental data from a series of experimental tests. For each test, the process is run with the adaptive loop open, and the values of the measurable parameters are recorded. After each test, the actual values of the performance indicators are
determined off-line and also recorded. Both the test conditions and the number of tests are carefully chosen to represent a statistically significant sampling of the actual range of process operation. Typically, several hundred tests are used in the analysis.

The set of programs, shown in Fig. 6, operates directly on experimental data supplied through cards and produces the training constants suitable for entry into the prototype performance computer. The shaded blocks in Fig. 6 indicate the four key programs; the others are of a minor data-conversion nature and are used to automate the procedure. The experimental data is supplied to the multiple linear regression program which determines a set of class boundaries and the weighting coefficients for the regression equations. In the following discussion, the class boundaries are called the classification scheme. Based on the classification scheme, the quantizer design program then determines the quantizing levels and generates the binary pattern which is used by the pattern processor training program. In this latter program, the weights, thresholds, and switch settings for the nonlinear threshold logic circuits are determined. The training constants determined by the first three programs are then supplied to the prototype system simulation program to determine the system accuracy. A discussion of the four key programs is given in the following sections.

**Multiple Linear Regression Program**

Determination of the weighting coefficients for the linear regression equation requires that the experimental data be separated into classes and a linear regression performed on each class. The general form of a linear regression for estimating a dependent variable \( H \) in terms of a set of independent variables \( P_1, P_2, \ldots, P_M \), is

\[
H = W_0 + W_1P_1 + W_2P_2 + \ldots + W_MP_M,
\]

where \( H_0 \) is the computed value of \( H \) and the \( W \)'s are weighting coefficients. The values of the weighting coefficients for the regression equation are generally determined from experimental values of \( H \), \( P_1 \), \( P_2 \), \ldots, \( P_M \) by the method of least squares, which selects the weights such that the sum of the squares of error of estimation is a minimum [8]. That is, the expression

\[
E = \sum_{i=1}^{N} (H_i - H_0)^2,
\]

where \( N \) is the number of experimental points, is minimized.

If the equation for \( H_0 \) is substituted into the expression for \( E \), the minimum value of \( E \) can be found by setting to zero all of the partial derivatives of \( E \) with respect to each \( W \). When this is done, the resulting \((M+1)\) simultaneous equations can be manipulated into the form

\[
\sum H = W_0N + W_1 \sum P_1 + W_2 \sum P_2 + \ldots + W_M \sum P_M
\]

\[
\sum H \cdot P_1 = W_0 \sum P_1 + W_1 \sum P_1^2 + \ldots + W_M \sum P_1P_M
\]

\[
\sum H \cdot P_M = W_0 \sum P_M + W_1 \sum P_1P_M + \ldots + W_M \sum P_M^2
\]

where the summations are performed over all \( N \) points. The compilation of the summations and the solution of the simultaneous equations for the optimum \( W \) values is performed by the regression program.

In separating the experimental data points into classes, the classification scheme used should be the one which yields the minimum error on the subsequent regression. The relative merit of a classification cannot be known, however, until after the regression has been performed. It is therefore necessary to perform linear regression on all possible classification schemes and then select the best.

Fig. 7 is a flow diagram for the multiple linear regression program which enables selection of the best classification scheme. The inputs to the program are a set of potential classification levels for all the variables plus the set of experimental data points. The program selects the first classification level for the first variable and separates the points into two classes. Then it applies the method of least-squares to each class and determines the total error due to the two linear regressions. Finally, it recombines the points and repeats the procedure for the next classification level until all potential classifications have been tested. At this time, the points are permanently divided into the two classes yielding the lowest error. The entire program is now rerun, but this time each classification separates the points into four classes, and four linear regressions are formed. The process is continued until an arbitrary number of classifications, \( S \), have been generated. Thus, each data point is assigned to one of the \( S \) classes. The listing of these assignments is known as the classification scheme.

After the classification scheme has been determined, the pattern-recognition network can be designed to perform the classifications. This is accomplished by two programs, the quantizer design program and the pattern processor training program.

To minimize computer time, relatively coarse classification levels should be used. Typically 10 classification levels per measurable variable were found to be sufficient. Relatively minor improvements could be obtained by certain "coarse-fine" selection procedures.

**Quantizer Design Program**

The flow diagram for the computer program used to
design the quantizer is illustrated in Fig. 8. In the design of the quantizer, each potential quantization level is tested and the level which provides the most information about the class is selected. Quantization levels are selected sequentially until no further information is provided by an additional level or until a predetermined number of quantization levels have been selected. In essence, optimum quantizations are those which separate the points into groups with the least amount of overlap.

The information provided by a potential quantization level $Q_3$ on a parameter $P_m$ in the range $Q_1 \leq P_m < Q_2$ is defined as

$$I_{Q_3} = \left| \frac{1}{k} \sum_{i=1}^{k} (c_i - c_{E}) \right| ;$$

where $c_i$ is the actual classification for each point, $c_{E_i}$ is the previous best estimate of the assigned classification for each point, and $k$ is the number of points in the subrange $Q_1 \leq P_m < Q_2$. The best estimate of the class for each group of points is defined as the average class over all points in the group. In testing each potential quantization level, it is always necessary to consider a potential quantization level $Q_3$ as being between two previously selected quantization levels, $Q_1$ and $Q_2$. The parameter limits of 0 and $P_{\text{max}}$ can be used where applicable. If the summations are performed separately, the information provided is simply the difference between the averages; that is

$$I_{Q_3} = \left| \hat{c} - c_{E} \right| .$$

This may be explained as follows. Without $Q_3$, the average of all points in the range $Q_1 \leq P_m < Q_2$ is $c_E$. If the average of the points in the subrange $Q_1 \leq P_m < Q_2$ is significantly different than $c_E$, then $Q_3$ provides new information about the class. (The summation could also be performed on the points in the other subrange, $Q_3 \leq P_m < Q_2$. The results of this summation would be $| - I_{Q_3} |$, however.)

**Pattern Processor Training Program**

Design of the pattern processor consists of selecting values for the internal parameters which establish a logic transfer function as close as possible to the desired relationship, as determined from the experimental data. This is accomplished by a training algorithm. The training algorithm that has been developed for the nonlinear
threshold logic circuit involves the adjustment of both the switch parameters, represented as a matrix $S_M$, and the weights of $A_{N+1}$. The basic operation of the training algorithm is as follows. In the event of an incorrect output, the first adjustments applied are to the weights, $A_1, A_2, \ldots, A_N$. An increment of plus or minus unity is added to each of the weights according to the following rule.

If

$$E(i) < 0$$

then

$$\Delta A_m = \beta_m(i)$$

otherwise

$$\Delta A_m = -\beta_m(i),$$

where $E(i)$ is the error signal. If application of the above rule is insufficient to make the output correct, an additional increment of plus or minus unity is added to the threshold value $A_1$ by reapplication of the above equation. If the output is still incorrect, adjustment of the switching parameters $S_M$ is used. Parameters stored during the training process determine a single switch for complementation and an appropriate weight for the associated $\beta$ value after the complementation is performed. Basically, the switch which is complemented is the one which results in the largest increase in associated weight. This rule tends toward the generation of an optimum set of cross-products for the problem at hand [9]. By optimum, it is meant that a minimum, or near minimum, number of pattern-classification errors is obtainable by the selected cross-products.

The training algorithm described above is executed on a digital computer according to the flow diagram of Fig. 9. As shown, each pattern is examined and the internal parameters (weights, thresholds, switch settings) are adjusted until the computed output agrees with the desired output. The procedure is repeated for each pattern of quantized test data, finally resulting in establishment of parameter values to be used in the actual processor. Typically, training is completed in less than 20 passes through the experimental data. Parameter values for 12 circuits (for each indicator computed) are required, as described below.

To compute the performance indicators with sufficient accuracy, the operating range is separated into a maximum of eight classes. Classification of the operating range into eight classes can be accomplished with the procedure shown in Fig. 10. Twelve decisions are made with each decision performed by a nonlinear threshold logic circuit. The first decision determines whether the pattern belongs to Classes 1, 2, 3, and 4 or to Classes 5, 6, 7, and 8. Next, depending on the outcome of the first decision, either a “Class 1, 2, or Class 3, 4” decision or a “Class 5, 6, or Class 7, 8” decision is made. After a third decision is made, the pattern is assigned to one of eight classes and the classification procedure could be terminated at this point. However, to improve the accuracy of the classification procedure, additional decision steps are added to the procedure to allow remerging of the decision “branches.” This remerging makes it possible to correct for errors made in early steps of the procedure. For example, suppose a Class 3 pattern is erroneously interpreted to be a Class 4, 5, 6, or 7 pattern by the first threshold logic circuit. If the remerging branches function correctly, the given pattern will still be assigned to Class 3. It should be noted that if the procedure just described, each threshold logic circuit has a specific function and can be trained independently of the other circuits.

**Prototype System Simulation Program**

The result of the above programs is a set of training constants consisting of quantization levels, weights and thresholds, and weighting coefficients. These constants completely specify the operation of the performance computer for a specific performance indicator.

For each indicator, three types of classification schemes have been considered: 1) dependent classification, where the data points are separated only according
to values of the dependent variable $H$: 2) independent classification, where the data points are separated only according to values of the independent variables $P_1, P_2, \ldots, P_M$; and 3) hybrid classifications, where the data points are separated according to values of both the dependent and independent variables. Clearly, the dependent classification schemes present the greatest challenge to the pattern recognition network. The regression program, however, is operated without advance knowledge of the errors that might occur in the pattern recognition network. It is important, therefore, that the regression program must not compare independent classification schemes with dependent classification schemes. Therefore, the entire set of training programs must be run separately for each type of classification. In the analysis of the experimental data, the multiple linear regression program was typically run about four times; two of the runs generated training constants for hybrid types of classifications, and the other two runs generated training constants for the best dependent and independent types of classification. Then, four distinct pattern recognition networks were trained, resulting in four possible sets of training constants for the performance computer. To determine the best of these four, and also to estimate the expected overall accuracy, a computer simulation program is used to exactly duplicate the operation of the performance computer.

The block diagram for the simulation program is illustrated in Fig. 11. This diagram is basically the same as the one for the prototype system described below. For each test sample, the input variables $P_1, P_2, \ldots, P_M$ are first applied to the parameter quantizer, where
they are converted to a binary pattern by comparison with the predetermined quantization levels. The binary pattern is then examined by the pattern processor, which generates an output code indicating the class to which that sample is assigned. The computer value of the performance indicator $H_e$ is then determined by a linear regression equation, using the set of weighting coefficients for the particular class. The operation thus far is identical to that of the prototype system.

Unlike the prototype system, however, the simulation program has available the actual value of the performance indicator $H$ for each data point. The simulation program thus computes the relative error $(H_e - H)/H_{\text{max}}$ at each point, and compiles the errors for tabulation. The output of the simulation program is a tabulation of errors suitable for plotting error distribution curves and selecting the best set of training constants.

As stated earlier, the operation of the performance computer is completely specified by the values of the training constants. There are, therefore, no special requirements regarding the type of classification scheme used.

**System Design and Implementation**

The functional design of the performance computer is illustrated by the block diagram given in Fig. 12. The measured parameters, $P_1$ to $P_{16}$, are normalized by variable-gain amplifiers, multiplexed into an analog-to-digital converter, and stored in binary form in the computer's central memory. Other stored data needed by the quantizer, the threshold logic network, and the linear regression weighting and summation network are entered into the central memory through a paper tape reader.

The computing cycle begins with the quantization of the stored parameters. This is accomplished by the quantizer which receives both the stored measured parameters $P$ and the quantizing levels $Q$ from the central memory. Each measured parameter is sequentially compared with its associated quantization levels. The resulting binary pattern $F$ is temporarily stored in a flip-flop register to be used by the cross-multiplier section of the nonlinear threshold logic network.

Cross-multiplier output bits are formed sequentially by multiplying selected bits of the input pattern $F$ in accordance with a stored switch specification matrix $S$. Each output bit is then multiplied by the corresponding term of the stored weight set $A$ and added to other cross-multiplier stored-weight products in the arithmetic unit. The final sum is compared to a threshold to determine the output of a threshold logic circuit.

The implementation of 12 circuits, as required in classifying the set of input parameters, is achieved by time-sharing one hardware unit for all the circuits, with a different set of stored constants for each circuit. All of the required stored binary terms are obtained from the central memory. Both the implementation approach for one circuit and the technique for obtaining several circuits minimize the required hardware.

The 12-bit classification code $G$ generated by the threshold logic network is stored in a flip-flop register and is operated on by the output decoder. The decoder
generates a one-out-of-eight code, where each of the eight possibilities corresponds to one and only one set of linear regression weighting coefficients \( W \). The weighting coefficients and parameters are sequentially obtained from the central memory, and the linear regression equation, which determines the performance indicator \( H_n \), is implemented by the arithmetic unit. To minimize hardware requirements, the same arithmetic unit is used in both the linear regression and threshold logic computations.

With the linear regression computation, the computing cycle for one performance indicator is concluded. When more than one performance indicator is to be computed for a set of measured parameters, the computing cycle is repeated with a different set of stored data for each additional indicator. A separate storage register and a digital-to-analog converter are provided for each of the five performance indicators, \( H_1 \) to \( H_5 \), which can be computed. The digital-to-analog converters are useful for connecting the computer to a strip-chart recorder used for displaying and recording the values of the computed indicators.

The implementation of the performance computer described above was accomplished using readily available components. The system, shown in Fig. 13, was built with 500 integrated-circuit packages and a 4096 10-bit-word core memory. The integrated circuits and the circuitry for five digital-to-analog converters, an analog signal multiplexer, and an analog-to-digital converter are all mounted on 20 printed circuit cards of the type shown in Fig. 14.
Operating Results and Concluding Remarks

The performance computer is a new generation system which brings to process control the techniques and hardware of modern digital computer technology. Successful results have been achieved for the performance computer operating in conjunction with a lathe and an electron beam welder. Table I shows the parameters measured and the performance indicators computed for these two processes. Plots which show the error distribution for performance indicators computed for the lathe and welder are shown in Figs. 15 and 16, respectively. These plots indicate that high accuracy was obtained, thus demonstrating that it is feasible both in theory and practice to compute performance indicators from measurable parameters. Furthermore, with the implementation approach used and the off-line training technique, low-cost production models of the computer can be realized. Therefore, accurate and economical on-line performance measurement equipment is now available, and significant improvements in manufacturing efficiency can be achieved.

A Delay Line and Logic Circuits Utilizing Charge-Storage Subharmonic Parametric Oscillators

ROBERT H. MITCHELL, THOMAS WILLIAMS, MEMBER, IEEE, AND WILLIAM D. RYAN, MEMBER, IEEE

Abstract—The charge-storage subharmonic parametric oscillator may be switched from one state to the other within one cycle of pump voltage. This property is used in the design and construction of a 6-element delay line. This delay line is shown to perform predictably when the elements are connected in a loop to give a self-switching arrangement.

For the construction of logic circuits, it is shown that problems exist due to switching transients. Providing these transients can be overcome, logic circuits, circulating stores, and eventually a complete system may be built.

Index Terms—Charge storage, delay line, logic, oscillator, parametric, subharmonic.

Manuscript received April 1, 1968; revised July 19, 1968. This paper was presented at the First Annual IEEE Computer Conference, Chicago, Ill., September 6–8, 1967.

R. H. Mitchell is with the Department of Electrical Engineering, Queen’s University, Kingston, Ontario, Canada.

T. Williams is with Texas Instruments Incorporated, Dallas, Tex.

W. D. Ryan is with the Department of Electrical Engineering, Queen’s University, Belfast, Northern Ireland.

List of Symbols

\( D \) = duration of SHPO output pulse, radian
\( L \) = circuit inductance, \( H \)
\( C \) = circuit capacitance, \( F \)
\( E_d \) = dc bias voltage
\( E_b \) = barrier potential of coupling diodes
\( Q \) = minority-carrier charge stored in the diode neutral regions
\( L_s \) = series inductance added to delay line, \( H \)
\( K = L/L_s \)
\( k \) = ratio of coupling inductance to circuit inductance
\( \tau \) = storage lifetime
\( \omega_0 \) = SHPO resonant angular frequency = \( \sqrt{1/LC} \)
\( \beta \) = phase angle of pump voltage at diode recovery
\( V \) = peak value of sinusoidal pump voltage

References