PLATO: A NEW PIECEWISE LINEAR SIMULATION TOOL

M.T. van Stiphout J.T.J. van Eijndhoven H.W. Buurman

Eindhoven University of Technology, Department of Electrical Engineering
P.O.Box 513, 5600 MB Eindhoven, The Netherlands
Tel.: 31-40-473710, Telex: 51163, Fax: 31-40-448375, Email: mart@ele.tue.nl

ABSTRACT

This paper describes the basic concepts of a new piecewise linear circuit simulation program called PLATO. Piecewise linear modeling is a very generic and powerful approach to the modeling of electronic components. It allows for the use of macro modeling and mixed-level simulation because all components are modeled in a uniform way. Models are stored in a compact matrix notation. Default models can easily be redefined or modified without affecting the program code but simply by editing the component library. Powerful algorithms are available for solving the piecewise linear equations, yielding global instead of local convergence, this way eliminating convergence problems occurring in more conventional simulators. Latency behavior is utilized by the application of multirate integration techniques together with efficient sparse matrix methods. Results show that computational effort is mainly restricted to active circuit parts.

1. INTRODUCTION.

The use of piecewise linear modeling in circuit simulation has several advantages. Most important is the ability to model a wide range of components in a uniform way. Logic gates, resistors, capacitors, transistors but also more complex subcircuits such as operational amplifiers can be described. This way we are able to mix low level components with macro models in a natural way. Clearly the resulting simulation program is highly suitable for mixed analog-digital circuit simulation. Secondly, piecewise linear solution algorithms show excellent convergence properties where a conventional circuit simulator such as SPICE often suffers from convergence problems. A simulator feature improving its flexibility is the absence of built-in models. All component descriptions reside in a model library which can easily be modified or extended.

A multirate integration method for a set of ordinary differential equations is a method that integrates subsets of equations with their optimal individual stepsize. The potentials of such a method are obvious. The computation time required can be minimized by integrating slowly varying subsets of equations with a large stepsize and fast varying subsets with a smaller one. The concept has been applied before, e.g. in SAMSON2, an event driven SPICE-like circuit simulator which partitions the circuit into subcircuits and assigns individual stepsizes to them.

The piecewise linear simulator described in this paper utilizes a fine grained version of the multirate approach, automatically exploiting the latent circuit parts.

This paper is an attempt to summarize the techniques implemented in PLATO, with a slight emphasis on transient analysis. In the sequel we will shortly introduce the piecewise linear modeling concept (section 2) and piecewise linear solution techniques (section 3) as applied in the simulator. Next the time integration algorithm (section 4) and some aspects of the solution process are discussed in more detail (section 5). Finally some optimizations (section 6) and simulation results are presented (section 7).

2. PIECEWISE LINEAR MODELING.

A general description of a continuous piecewise linear dynamical system [1] is given by equation (2.1).

$$\begin{bmatrix}0\\ \dot{u}\\ p\end{bmatrix} = \begin{bmatrix}A_{11} & A_{12} & A_{13}\\ A_{21} & A_{22} & A_{23}\\ A_{31} & A_{32} & A_{33}\end{bmatrix}\begin{bmatrix}x\\ u\\ q\end{bmatrix} + \begin{bmatrix}a_1\\ a_2\\ a_3\end{bmatrix} \qquad (2.1)$$

$$\text{where } \dot{u} = \frac{\delta u}{\delta t}, \text{ and} \qquad (2.2)$$

$$p^t \cdot q = 0,\ p \geq 0 \text{ and } q \geq 0. \qquad (2.3)$$

The conditions (2.3) imply that p and q contain nonnegative elements only and have zero inner product. Vectors x and u represent the systems terminal variables and the state variables used to model dynamical behavior. The matrix shown can describe 2^n segments, with n the dimension of matrix A_{33}. Let us for the moment ignore the state variables and assume that vector q is identical to zero. Now each segment is determined by a linear mapping $0 = A_{11}x + a_1$ and a set of inequalities defining the domain in which the mapping is valid: $p = A_{31}x + a_3 \geq 0$. The mapping of a neighboring segment can be found by pivoting on an element of submatrix A_{33}. Because of the continuous modeling, the new mapping after pivoting on element A_{33}^{kk} differs from the previous one by a rank one update solely [2]:

$$0 = A'_{11}x + a'_1, \text{ with} \qquad (2.4)$$

$$A'_{11} = A_{11} - \frac{A_{13}^{*k} \cdot A_{31}^{k*}}{A_{33}^{kk}} \text{ and } a'_1 = a_1 - \frac{A_{13}^{*k} \cdot a_3^k}{A_{33}^{kk}}. \qquad (2.5)$$

The often used wildcard notation in e.g. A_{13}^{*k} means that the

2024/90/0000/0235$01.00©1990 IEEE

integration formulas and assume that $r = 1$ throughout the rest of this paper.

The transient analysis proceeds in an event driven manner. We distinguish between two types of events:

— dynamic events and

— pl events.

A pl event occurs if the solution vector reaches a boundary hyperplane of the current segment. This happens if for some leafcell a component of p becomes 0. Pl events can be calculated explicitly using the time derivative of p: suppose

$$\exists_k \ \bar{p}_k < 0 \wedge \bar{p}_k \leq \bar{p}_i, \ i \neq k, \tag{4.8}$$

then the next pl event for this leafcell is at time $t_{now} - (p_k \,/\, \bar{p}_k)$. The integration must be stopped until the piecewise linear equations are solved. A complication occurs if the solution algorithm has to take a step in the direction of a q variable. Now the $\dot{u}$ may change discontinuously in which case the integration has to be restarted for the leafcell involved. Such a restart is not necessary if only simple pivots are performed. Note however that pivoting causes an upper bound on the number of continuous circuit variable derivatives. Unfortunately this restriction implies that nothing can be gained from the application of integration formulas with order greater than 2 [9]. After the piecewise linear equations have been solved, the leafcells affected by changes in $\bar{x}$ have to be examined. If necessary their events and stepsizes have to be corrected. Note that the changes in $\bar{x}$ due to pivoting can be computed easily by solving

$$0 = (A + c \cdot r^t) \cdot \Delta \dot{x} + c \cdot (p + r^t \cdot \dot{x}) \tag{4.9}$$

as described in [8], in which c and r^t are known column and row vectors and p is the member of the leafcell source vector corresponding to the pivot row.

Next we consider the dynamic events. For every leafcell the companion model has been determined and inserted in the overall system matrix A. A dynamic event occurs every time the validity of a companion model expires. The use of a fine grained multirate integration technique implies that every leafcell obtains its own optimal stepsize. Naturally it is possible and even desirable for a number of leafcells to share the same stepsize. This way slowly varying components will be integrated with large time steps while fast varying components receive a small stepsize. Suppose some leafcell l has a dynamic event at time $t_{ev} = t_{n+1}$. First we update the x-, u-, $\dot{u}$- and p-variables corresponding to leafcell l using the approximations of their time derivatives $\bar{x}$, $\bar{u}$, $\bar{\dot{u}}$ and $\bar{p}$. Next a new stepsize h_{n+1} is computed for the next integration step. The previous stepsize h_n is maintained if there is no significant difference between the new stepsize h_{n+1} and h_n. In this case only the $\bar{b}$ vector changes due to the update of $\dot{u}$. The changes in $\bar{b}$ are now used to compute the changes in $\bar{x}$ denoted by $\Delta\bar{x}$. A more complicated situation arises if the stepsize requires adjustment. Although this causes $\bar{b}$ as well as J to change, we can still compute an update $\Delta\bar{x}$ for vector $\bar{x}$ in a way analogous to equation (4.9). Finally, we have to investigate the influence of the changes in $\bar{x}$ on the related leafcells. Since the circuit variable derivatives have changed for those leafcells, it may well be possible that the stepsize substituted is no longer valid. Therefore we have to update the x-, $\dot{u}$-, p- and $\bar{x}$-variables, recompute $\bar{p}$ and check if the applied stepsizes are still appropriate. If necessary, the event times must be adjusted.

5. PIVOTING.

As explained in the previous section, a pl event during transient analysis interrupts the time integration. At this time the pl equations have to be resolved which involves pivoting. Unlike the situation during the initial analysis, the leafcells jacobian matrix is now updated with a rank one update due to the substitution of an integration formula as indicated by equation (4.4). In fact every A_{ij}, $i, j \in \{1, 3\}$ has been updated, so the pivot value and the update vectors are not immediately available. Nevertheless it is still possible to obtain the new jacobian matrix by performing a rank one update.

Lemma: let P denote a pivot operation on an element of submatrix A_{33} and let L denote the substitution of a multistep integration formula. Now $P \circ L = L \circ P$.

Proof: let us extend the results in (4.4) to all submatrices A_{ij}. We find :

$$\bar{A}_{ij} \triangleq A_{ij} + \delta A_{i2} D^{-1} A_{2j} \ \text{for } i, j \in \{i, j\}. \tag{5.1}$$

Let us further define:

$$c_1 \triangleq A_{13}^{*k}, \ c_2 \triangleq A_{23}^{*k}, \ r_1 \triangleq A_{31}^{k*}, \ r_2 \triangleq A_{32}^{k*} \tag{5.2}$$

$$\text{and } p \triangleq A_{33}^{kk}. \tag{5.3}$$

Now define the following matrices where A is a shorthand for matrices A_{ij}:

A' pivot on matrix A

$\bar{A}$ apply (4.3) to matrix A

A'' apply (4.3) to matrix A'

$\bar{\bar{A}}$ pivot on matrix $\bar{A}$

and show that $\bar{\bar{A}}$ equals A''. First we construct A'':

$$A'' = A'_{11} + \delta A'_{12} D'^{-1} A'_{21}, \tag{5.4}$$

$$D = I - \delta A_{22}, \ \ D' = I - \delta A'_{22}. \tag{5.5}$$

Now substitute the expressions for A':

$$A'_{11} = A_{11} - \frac{c_1 r_1}{p}, \ \ c'_1 = \frac{c_1}{p} \tag{5.6}$$

$$A'_{12} = A_{12} - \frac{c_1 r_2}{p}, \ \ c'_2 = \frac{c_2}{p} \tag{5.7}$$

$$A'_{21} = A_{21} - \frac{c_2 r_1}{p}, \ \ r'_1 = -\frac{r_1}{p} \tag{5.8}$$

$$A'_{22} = A_{22} - \frac{c_2 r_2}{p}, \ \ r'_2 = -\frac{r_2}{p} \tag{5.9}$$

$$p' = \frac{1}{p} \tag{5.10}$$

The result contains a complicated matrix inverse which can be eliminated by applying the well known Shermanasterisk can be replaced by any legal index value, i.e. in this case indicating column k of matrix A_{13}.

3. SOLVING THE PL EQUATIONS.

A hierarchical circuit description serves as input to the simulation program. Every leafcell of the circuit hierarchy is modeled by a matrix in the form of equation (2.1). The leafcell submatrices A_{11} together with the interconnection equations are used to build the system matrix A. The system source vector a is composed out of leafcell subvectors a_1 and the input specifications. As soon as A and a are available, the circuit variables are solved from $0 = Ax + a$ in which x is now a global vector contrary to the x used in (2.1). Next we scan all leafcells and use x to compute the p-vectors. While doing so, we assume $u = 0$ as an initial condition and $q = 0$ for the initial segment so $p = A_{31}x + a_3$. An initial solution is found if $p \geq 0$ for all leafcells. If any leafcell has a p vector with a negative component, we have to apply some algorithm to solve what is called the Linear Complementarity Problem (LCP):

$$v = M \cdot i + m,\ v^t \cdot i = 0,\ v \geq 0 \text{ and } i \geq 0. \tag{3.1}$$

For a single leafcell we would simply solve $p = A_{33} \cdot q + (A_{31} \cdot x + a_3)$ so matrix M is explicitly available. In general however matrix M will have to be constructed by eliminating all internal circuit variables so there is a distinction between the M and A_{33} matrices. In practice the existence of M can be emulated as described in [3].

Several algorithms are available for solving the LCP. The one applied in the simulator is a pivoting type algorithm as described by van de Panne [4]. Basically, this algorithm tries to solve the pl equations by performing principal (block) pivots on elements of submatrices A_{33}. For a transient analysis this solution suffices. A dc solution can be obtained by solving:

$$\begin{bmatrix} 0 \\ p \end{bmatrix} = \begin{bmatrix} A'_{11} & A'_{13} \\ A'_{31} & A_{33} \end{bmatrix} \cdot \begin{bmatrix} x' \\ q \end{bmatrix} + \begin{bmatrix} a'_1 \\ a_3 \end{bmatrix} \tag{3.2}$$

with

$$A'_{11} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix},\ A'_{31} = \begin{bmatrix} A_{31} & A_{32} \end{bmatrix}, \tag{3.3}$$

$$A_{13} = \begin{bmatrix} A_{13} \\ A_{23} \end{bmatrix} \text{ and } x' = \begin{bmatrix} x \\ u \end{bmatrix},\ a'_1 = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \tag{3.4}$$

4. TRANSIENT ANALYSIS.

The simulation program solves for the circuit variable derivatives rather than the circuit variables themselves. Therefore we introduce the backward differences $\bar{u}_{n+1}$, $\bar{x}_{n+1}$, $\bar{q}_{n+1}$ and $\bar{\dot{u}}_{n+1}$:

$$\bar{x}_{n+1} \triangleq \frac{x_{n+1} - x_n}{h_n} \qquad \bar{u}_{n+1} \triangleq \frac{u_{n+1} - u_n}{h_n}$$
$$\bar{q}_{n+1} \triangleq \frac{q_{n+1} - q_n}{h_n} \qquad \bar{\dot{u}}_{n+1} \triangleq \frac{\dot{u}_{n+1} - \dot{u}_n}{h_n} \tag{4.1}$$

Subtracting equation (2.1) for time points $t = t_{n+1}$ and $t = t_n$ we find:

$$\begin{bmatrix} 0 \\ \bar{\dot{u}}_{n+1} \\ \bar{p}_{n+1} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} \begin{bmatrix} \bar{x}_{n+1} \\ \bar{u}_{n+1} \\ \bar{q}_{n+1} \end{bmatrix} \tag{4.2}$$

in which the constant source vector $(a_1, a_2, a_3)^t$ has disappeared. From here on subscripts $n+1$ are often omitted. We now eliminate the unknown $\bar{\dot{u}}_{n+1}$ by substituting a linear multistep integration formula with constant step size:

$$0 = \sum_{i=-1}^{i=p} \alpha_i u_{n-i} + h \sum_{i=-1}^{i=p} \beta_i \dot{u}_{n-i} \tag{4.3}$$

After some rearranging we find:

$$0 = J \cdot \bar{x} + \bar{b} \tag{4.4}$$

in which

$$J = A_{11} + \delta A_{12} D^{-1} A_{21}, \tag{4.5}$$

$$D = I - \delta A_{22},\ \delta = -\frac{\beta_{-1}}{\alpha_{-1}} h \tag{4.6}$$

and

$$\bar{b} = -\frac{(\beta_{-1} + \beta_0)}{\alpha_{-1}} A_{12} D^{-1} \dot{u}_n - \frac{A_{12} D^{-1} (\alpha_{-1} + \alpha_0) u_n}{h \alpha_{-1}} - \frac{1}{h\alpha_{-1}} A_{12} D^{-1} \sum_{i=1}^{i=p} (\alpha_i u_{n-i} + h \beta_i \dot{u}_{n-i}) \tag{4.7}$$

from which $\bar{x}$ is rapidly solved. Matrix J will be referred to as the companion model of a piecewise linear component. Once $\bar{x}$ is known, $\bar{u}$, $\bar{\dot{u}}$ and $\bar{p}$ can be solved successively. Of course these results are valid under the assumption that the inversion of D is feasible. For some special cases like backward euler (BE), the trapezoidal rule (TR), the second order backward differentiation formula (BDF2) [5] or a second order A-contractive formula (ACT2) [6], δ and $\bar{b}$ are listed in table 1.

TABLE 1. δ and $\bar{b}$ for several integration rules.

	δ	$\bar{b}$
BE	h	$A_{12}D^{-1}\dot{u}_n$
TR	$\frac{1}{2}h$	$A_{12}D^{-1}\dot{u}_n$
BDF2	$\frac{2}{3}h$	$\frac{2}{3}A_{12}D^{-1}\dot{u}_n + \frac{1}{3}A_{12}D^{-1}\bar{u}_n$
ACT2	$\frac{2}{3}h$	$\frac{14}{15}A_{12}D^{-1}\dot{u}_n - \frac{1}{5}A_{12}D^{-1}\bar{u}_n + \frac{4}{15}A_{12}D^{-1}\dot{u}_{n-1}$

Observe that the update $\delta A_{12} D^{-1} A_{21}$ on matrix A_{11} is a rank r update where r is the rank of matrix A_{22}. In many cases this rank will be equal to one in which case a change of stepsize requires only a simple rank one update on the system matrix. The corresponding LU decomposition can be updated by a very efficient algorithm devised by Bennett [7]. A sparse matrix implementation of this algorithm was already presented in [8]. Note that for a 2-step method like BDF2 the coefficients α and β depend on h_n as well as h_{n-1} and require recomputation if a stepsize changes or if $h_{n-1} \neq h_n$. For simplicity we will restrict ourselves to 1-step

Morrison-Woodbury formula [10,11]:

$$(A + XCY^t)^{-1} = A^{-1} - A^{-1}X(C^{-1} + Y^tA^{-1}X)^{-1}Y^tA^{-1} \quad (5.11)$$

with

$$C \equiv \frac{\delta}{p}, A \equiv D, X \equiv c_2, Y^t \equiv r_2. \quad (5.12)$$

Using the fact that

$$K = (\frac{p}{\delta} + r_2D^{-1}c_2)^{-1} \text{ and } k = r_2D^{-1}c_2 \quad (5.13)$$

are constants and some juggling with terms we finally obtain:

$$A'' = A_{11} + \delta A_{12}D^{-1}A_{21} \quad (5.14)$$

$$- K(c_1 + \delta A_{12}D^{-1}c_2)(\frac{r_1}{\delta} + r_2D^{-1}A_{21}). \quad (5.15)$$

The derivation of an expression for $\bar{\bar{A}}$ is a bit simpler. Combining

$$\bar{\bar{A}}_{11} = \bar{A}_{11} - \frac{\bar{c}_1\bar{r}_1}{\bar{p}} \quad (5.16)$$

with

$$\bar{A}_{11} = A_{11} + \delta A_{12}D^{-1}A_{21} \quad (5.17)$$

$$\bar{A}_{13} = A_{13} + \delta A_{12}D^{-1}A_{23} \rightarrow \bar{c}_1 = c_1 + \delta A_{12}D^{-1}c_2 \quad (5.18)$$

$$\bar{A}_{31} = A_{31} + \delta A_{32}D^{-1}A_{21} \rightarrow \bar{r}_1 = r_1 + \delta r_2D^{-1}A_{21} \quad (5.19)$$

$$\bar{A}_{33} = A_{33} + \delta A_{32}D^{-1}A_{23} \rightarrow \bar{p} = p + \delta r_2D^{-1}c_2 \quad (5.20)$$

we almost automatically find that $\bar{\bar{A}}$ equals A'' which completes the proof

6. EVENT SCHEDULING.

As already pointed out by C.W. Gear [12] there is a disadvantage to the approach sketched in section 4. Since it is impossible to predict the circuit behavior while determining a new stepsize for a specific leafcell, we may be forced to correct it before it reaches its event time due to activities in other parts of the circuit. This causes a lot of computational overhead, involving the update of circuit variables and the recomputation and effectuation of stepsizes. Gear also showed that no merit can be expected from the application of a multirate integration technique if the stepsizes of the subproblems lie close together. So the rigorous application of the multirate principle described above may not be very efficient.

The integration algorithm explained in section 4 is very well suited for the handling of multiple events at the same time. For every dynamic event the source vector $\bar{b}$ is adjusted and the new $\bar{x}$ is determined. Multiple events can be processed by simply catenating their contributions in the $\bar{b}$ vector. Since it is often the case that related components have events at about the same time, it is clear that a lot of cpu time can be saved by clustering dynamical events. As long as every component receives its own optimal stepsize however, it will rarely be possible to service more than one event at a time.

A way to achieve improvements for both drawbacks is the discretization of event times. Forcing nearby events to a grid results in the handling of multiple events. A further reduction of the overhead can be achieved by forcing groups of related leafcells to use the same (minimum) stepsize. This way the recomputation of stepsizes is minimized and the need for reducing them will diminish. Dynamic event times are discretized as follows. Assume the length of the integration interval is given by I, then events are forced to one of the following grids:

$$I \cdot 2^{-k} \text{ for } k = 1, 2, \cdots, N. \quad (6.1)$$

Every time a new event has to be determined, a value of k is chosen such that the desired stepsize can be mapped onto the discrete time axis without violating the time integration accuracy requirements represented in the stepsize h. After the next event has been determined, the leafcells stepsize is set accordingly.

7. SIMULATION RESULTS.

Some simulation statistics are presented to illustrate the concepts explained in this paper. Circuits simulated are an ad converter, an 8-stage shift register, a simple pll, several stage ring inverters and inverter chains. The ad converter contains among others an analog multiplexer, logic gates and more conventional leafcells like resistors and capacitors. The actual ad conversion is implemented using a macro model. A comparison of program statistics for all test circuits is shown in table 2. All simulations were done on a HP9000s835 computer.

8. CONCLUSIONS.

The concepts described in this paper have been implemented in a piecewise linear simulation program called PLATO. Latency is exploited by sparse matrix techniques and the application of multirate integration techniques. The effectiveness of the applied methods is illustrated by some test circuits which show that computational effort is restricted to active circuit parts.

REFERENCES.

[1] W.M.G. van Bokhoven, "Piecewise-Linear Modelling and Analysis," PhD Thesis, Eindhoven, The Netherlands, May 1981.

[2] T. Fujisawa, E. kuh, and T. Ohtsuki, "A Sparse Matrix Method for Analysis of Piecewise-Linear Resistive Networks," *IEEE Trans. on Circuit Theory*, vol. CT-19, no. 6, pp. 571-584, Nov. 1972.

[3] J.T.J. van Eijndhoven, "A piecewise linear simulator for large scale integrated circuits," Ph.D. Thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, Dec. 1984.

[4] C. van de Panne, "A Complementary Variant of Lemke's Method for the Linear Complementarity Problem," *Mathematical Programming*, vol. 7, pp. 283-310, North-Holland Publishing Company, 1974.

[5] C.W. Gear, "The Automatic Integration of Ordinary Differential Equations," *Communications of the ACM*, vol. 14, no. 3, pp. 176-179, March 1971.

TABLE 2. Program statistics

circuit	adc	shift	pll	ring7	ring15	ring21	chain5	chain10
#components	74	56	10	14	30	42	11	21
matrix size	149	164	10	43	91	127	31	61
matrix density	2.2	3.5	24.0	13.4	6.6	4.7	17.3	9.2
pivots (initial)	78	16	6	7	15	21	5	10
pivots (transient)	9870	200	238	195	214	233	11	23
#dyadic updates	13834	1951	297	2251	3248	3718	176	376
#forw/backw subs.	20384	1667	757	4921	5705	6299	401	847
#pl events	2376	200	138	195	217	234	11	23
#dyn. events	5507	1813	251	5966	7243	7980	509	1070
cpu time	52.7	15.8	0.6	30.4	51.2	64.6	1.7	6.0

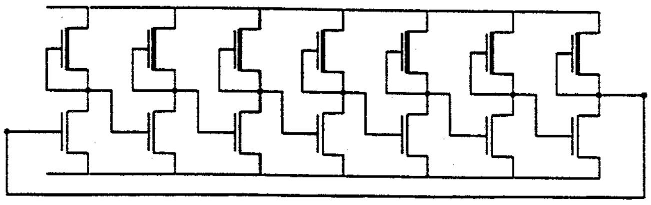

Figure 1. Nmos ring inverter with 7 stages: ring7.

[6] O. Nevanlinna and W. Liniger, "Contractive Methods for Stiff Differential Equations - Part I," *BIT*, vol. 18, pp. 457-474, 1978.

[7] J.M. Bennett, "Triangular Factors of Modified Matrices," *Numerische Mathematik*, no. 7, pp. 217-221, 1965.

[8] J.T.J. van Eijndhoven and M.T. van Stiphout, "Latency Exploitation in Circuit Simulation by Sparse Matrix Techniques," in *Proc. of the Int. Symp. on Circuits and Systems*, pp. 623-626, University of Helsinki, Espoo, Finland, June 7-9, 1988.

[9] I.N. Hajj and S. Skelboe, "Time-Domain Analysis of Nonlinear Systems with Finite Number of Continuous Derivatives," *IEEE Trans. on Circuits and Systems*, vol. CAS-26, no. 5, pp. 297-303, May 1979.

[10] J. Sherman and W.J. Morrison, "Adjustment of an Inverse Matrix Corresponding to Changes in the Elements of a given Column or a Given Row of the Original Matrix," *Ann. Math. Stat.*, no. 20, p. 621, 1949.

[11] M. Woodbury, "Inverting Modified Matrices," Memorandum 42, Statistics Research Group Princeton, New Jersey, 1950.

[12] C.W. Gear, "Automatic Multirate Methods for Ordinary Differential Equations," in *Information Processing 80*, pp. 717-722, North-Holland Publishing Company, 1980.

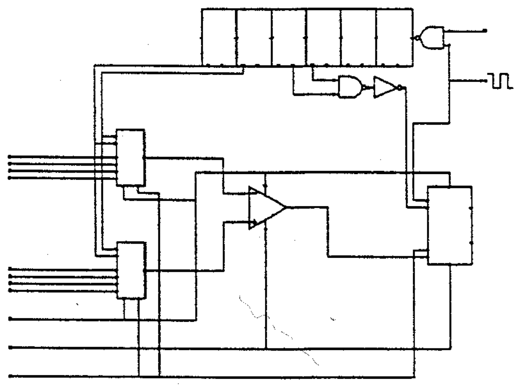

Figure 2. Analog to digital converter.