THE SIMULATION OF LARGE-SCALE INTEGRATED CIRCUITS

by

A. R. Newton

Memorandum No. UCB/ERL M78/52

July 1978

THE SIMULATION OF LARGE-SCALE INTEGRATED CIRCUITS

by

A. Richard Newton

Memorandum No. UCB/ERL M78/52

July 1978

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Abstract

Electronic circuit simulation programs can accurately predict voltage and current waveforms for small integrated circuits but as the size of the circuit increases, e.g. for Large-Scale Integrated (LSI) Circuits involving more than 10000 devices, the cost and memory requirements of such analyses become prohibitive.

Logic simulators can be used for LSI digital circuit evaluation and design if only first-order timing information based on user-specified logic gate delays is required. If voltage waveforms and calculated delays are important, a timing simulator may be used. In many circuits, however, there are critical paths or analog circuit blocks where more accurate circuit analysis is necessary.

This dissertation describes the *hybrid* simulation program SPLICE, developed for the analysis and design of LSI Metal-Oxide-Semiconductor (MOS) circuits. SPLICE allows the designer to choose the form of analysis best suited to each part of the circuit and logic, timing and circuit analyses are performed concurrently. The use of an event scheduling algorithm and selective-trace analysis allows the program to take advantage of the relatively low activity of LSI circuits to reduce the cost of the simulation.

SPLICE is between one and three orders of magnitude faster than a circuit simulation program, for comparable analysis accuracy, and requires less than ten percent of the data storage used in a circuit analysis. SPLICE is written in FORTRAN and is approximately 8000 statements long.

The algorithms and data structures used in SPLICE are described and a number of example simulations are included.

Acknowledgements

The author wishes to express his sincere appreciation to Prof. D. O. Pederson for his continuing guidance and encouragement. He also gratefully acknowledges numerous helpful discussions with, and assistance in examples and programming by, G. R. Boyle, E. Cohen, J. Crawford, M-Y. Hsueh, L. Scheffer and the members of the Integrated Circuits Group at the University of California, Berkeley.

Support received from Corporate Engineering, Hewlett Packard, and in particular the encouragement provided by M. Brooksby and R. Smith, is gratefully acknowledged.

The author wishes to thank his parents, Bette and Rodger, for their support and encouragement throughout the course of this work.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

A number of simulation techniques are available for the analysis of electronic circuits. For small circuits where analog voltage levels are critical to circuit performance, or where tightly coupled feedback loops exist, a circuit simulator such as SPICE2 [1] can accurately predict circuit performance. As the size of the circuit increases, the cost and memory requirements of such an analysis become prohibitive.

Large-scale integrated (LSI) circuits can contain over 10000 transistors. Consider the analysis of a circuit containing 10000 Metal-Oxide-Semiconductor (MOS) transistors, for 1000ns of simulation time on an IBM 370/168 computer. If the circuit simulation program SPICE2 were used and computer time cost $1000/hour, the cost of such a simulation would exceed $30000 (App. 9). For most circuits, a number of simulations are required before the design is completed.

For circuits where verification of the logical operation of the circuit and only first-order timing information is sufficient, a logic simulator [2]-[6] may be used. Logic simulators provide a discrete "on/off" analysis for digital circuits and, because of the simplifications made during the simulation, can analyze circuits containing over 10000 logic gates. If dynamic charge-storage effects or bilateral circuit elements are important, or if a waveform analysis is required and the expense of a circuit simulation is not justified, a timing simulator can be used [7], (App. 8). Timing simulation is a simplified form of circuit simulation which takes advantage of the properties of digital circuits to reduce the simulation time. Fortunately, a large portion of a typical large scale integrated circuit is digital in nature. For this reason, simplifications can be made in the analysis which greatly increase execution speed yet provide adequate information about circuit performance.

A comparison of circuit, timing and logic analysis programs for the analysis of the same problem on the same computer [9] has shown the timing simulator MOTIS-C (App. 8) to be typically two orders of magnitude faster than SPICE2, and the logic simulator SALOGS-3 [2] to be three orders of magnitude faster than SPICE2.

It is evident that for the analysis of large digital systems which contain tightly coupled circuit blocks or critical paths, a simulator is required which will combine the accuracy of circuit simulation (for critical parts of the network) with the speed and memory-saving advantages of timing and logic simulation for the remainder of the circuit [10]. At the same time, techniques must be used which can take advantage of the relatively low circuit activity of LSI circuits. If only those portions of the circuit which are active at any time are analyzed by the simulator, substantial time savings can be achieved.

This dissertation describes the *hybrid analysis program* SPLICE (Simulation Program with Large-scale Integrated Circuit Emphasis), which allows concurrent circuit, timing and logic analyses of various parts of the same integrated circuit. Each part of the circuit is partitioned by the user from the rest of the network and hence need only be simulated when it is active.

SPLICE has achieved speed advantages of from one to three orders of magnitude over circuit simulator SPICE2 and requires from one to ten percent of the memory used by SPICE2. The circuit designer may choose the form of analysis (circuit, timing or logic) suitable for each part of the circuit to be simulated.

Chapter 2 introduces the LSI circuit design problem and describes briefly the techniques used today for the analysis of electronic circuits. These techniques include circuit, timing and logic analysis. The hybrid analysis program DIANA, developed by Arnout and DeMan [10] is also described.

The algorithms used in circuit, timing and logic analysis programs are described in Chapter 3 and Chapter 4 describes the algorithms used in program SPLICE. In particular, the techniques used for exploiting the low circuit activity of typical LSI circuits and the interface between the various forms of analysis are presented. The use of an event-scheduler, similar to that used in a logic simulator, for the control of circuit, timing and logic analyses is also described.

Chapter 5 describes the program structure of SPLICE and the data structures used during the analysis. These data structures are critical for the efficient operation of the program.

A number of example simulations are included in Chapter 6. These examples include a 256-by-1 bit dynamic RAM circuit, which combines circuit, timing and logic analyses in the same simulation, and a 700 MOS transistor timing analysis performed on an integrated digital filter which has subsequently been fabricated for use in an electronic instrument. The latter example illustrates how relatively low circuit activity is exploited to enhance the speed of the simulation.

In the final chapter, a summary of program performance is presented and areas for future work are described.

There are eleven appendices. The first two appendices contain some details of timing analysis, appendices three to six contain a description of the data structures used in program SPLICE and appendix seven contains the input to SPLICE for the examples of Chap. 6. Copies of three papers which describe earlier work in this area are included in appendices eight to ten and appendix eleven contains a listing of program SPLICE.

CHAPTER 2

THE LSI CIRCUIT SIMULATION PROBLEM

2.1. Introduction

Circuit simulators, such as SPICE2 [1],[11], SCEPTRE [12] and ASTAP [13], have proved effective for the analysis of small circuit blocks (less than 100 active devices) by providing accurate voltage and current waveforms. Today's large-scale integrated circuits contain over 10000 active devices. The application of such simulators to circuits of more than 1000 active devices is often not cost-effective or is beyond available computer resources, as brought out in a later example.

Logic simulators provide a discrete "on/off" analysis of the circuit under test. By the use of simple gate-level models and Boolean arithmetic, logic simulators such as SALOGS-4 [3], F-LOGIC [4], CC-TEGAS3 [5], and LOGCAP [6] are capable of economically analyzing systems containing the equivalent of over 100000 active devices. The simplicity of the models used in logic simulation and the relatively small number of discrete signal levels available in logic simulators, only a first-order timing analysis for design verification can be provided. Due to the simplicity of the signal representation, however, logic simulators can generate and validate the test patterns used in digital circuit testers and simulate the effect of a variety of circuit faults.

A simplified form of circuit analysis called *timing simulation* [7] has been developed recently and its performance lies between circuit and logic analyses. Timing simulators [7], (App. 8), [14] take advantage of the properties of digital networks to simplify both the active device models and the arithmetic required for the analysis. Between one and three orders of magnitude computational speed improvement over circuit simulation have been obtained using timing simulation. The actual speedup depends on both the type of circuit being analyzed and the level of model complexity used by the program, as described later in this chapter.

In the remainder of this chapter the LSI design process is introduced and the importance of modularity and regularity in both the circuit design and the circuit analysis are explained. The concepts behind circuit, timing and logic analysis are also presented while the details of specific algorithms are included in Chap. 3.

A speed comparison of different simulators, running on the same computer, analyzing the binary-to-octal decoder circuit of Fig. 2.1, is presented in Table 2.1. The circuit simulation was performed using SPICE2, timing simulation with program MOTIS-C (App. 8) and SALOGS-3 [2] was used for the logic simulation. This table illustrates that timing simulation can be two orders of magnitude faster than circuit simulation and logic analysis can be as much as three orders of magnitude faster than circuit analysis on the same computer.

There are many LSI circuits where a logic simulation alone cannot accurately predict circuit performance. For the design of a Random Access Memory (RAM), an accurate circuit level analysis of each sense amplifier and associated storage transistors is required to predict its performance satisfactorily. Integrated circuits which combine digital logic with analog functions such as active filters or analog-to-digital converters also require a circuit-level analysis of the analog circuit blocks. For this reason *hybrid* analysis programs have been developed. These programs combine circuit, timing and/or logic analysis in a single program and allow the designer to analyze some parts of the circuit with detailed device-level analysis and

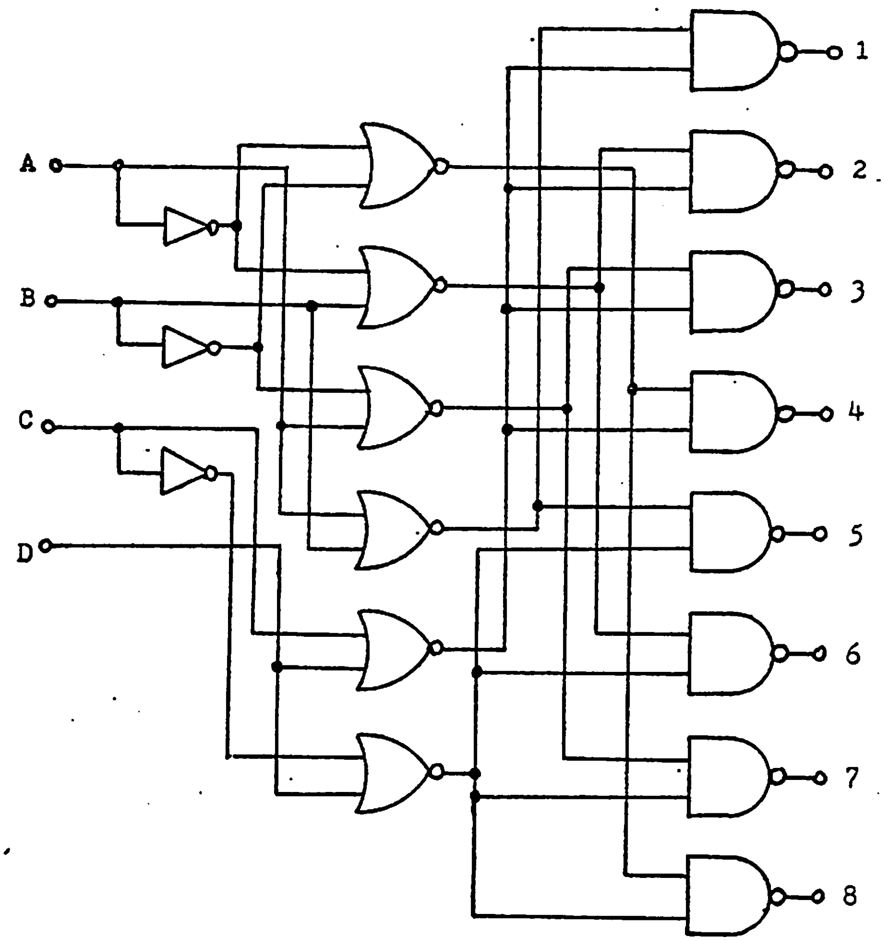


Fig. 2.1 Schematic Diagram of the Binary-to-Octal Decoder

	Central Processor Time per Print Point (2ns) (seconds)	Normalized
CIRCUIT (SPICE2)	1.3	3000
TIMING (MOTIS-C)	0.0037	6
LOGIC (SALOGS-3)	0.0006	1

Table 2.1 Analysis Times for the Binary-to-Octal Decoder

sophisticated device models while less critical digital parts of the circuit may be analyzed using timing or logic analysis.

The partitioning of the circuit into analog and digital blocks allows the designer to choose the level of modeling appropriate to each portion of the circuit and hence reduce the total cost of the analysis. The circuit partitioning is also used by the program to take advantage of the relative inactivity of large digital circuits and reduce analysis time even further. The algorithms used to perform these tasks are described in Chap. 3. Programs of this type include DIANA [10] and SPLICE described in Chap. 4 and Chap. 5.

2.2. The LSI Design Process

When an integrated circuit contains less than 100 active devices or is of a regular structure, such as a register or Programmable Logic Array (PLA), it is often possible for a single engineer to design the entire circuit. With large circuits containing large blocks of random logic this is no longer the case. The circuit must be partitioned into functionally-independent blocks, each of which may be partitioned further, until each piece of the circuit is small enough to be designed by a single engineer. A block diagram of such a partitioning process is shown in Fig. 2.2. The partitioning process is hierarchical and the circuit often contains regular structures such as RAMs, Read-Only Memories (ROMs), PLAs or shift registers.

As the design progresses the circuit may be verified at each level of complexity. The set of design verification aids corresponding to each level of design complexity is shown in Fig. 2.3.

It is often sufficient to perform the detailed and relatively expensive circuit analysis on the separate modules at the lowest level of Fig. 2.2, where the circuit is

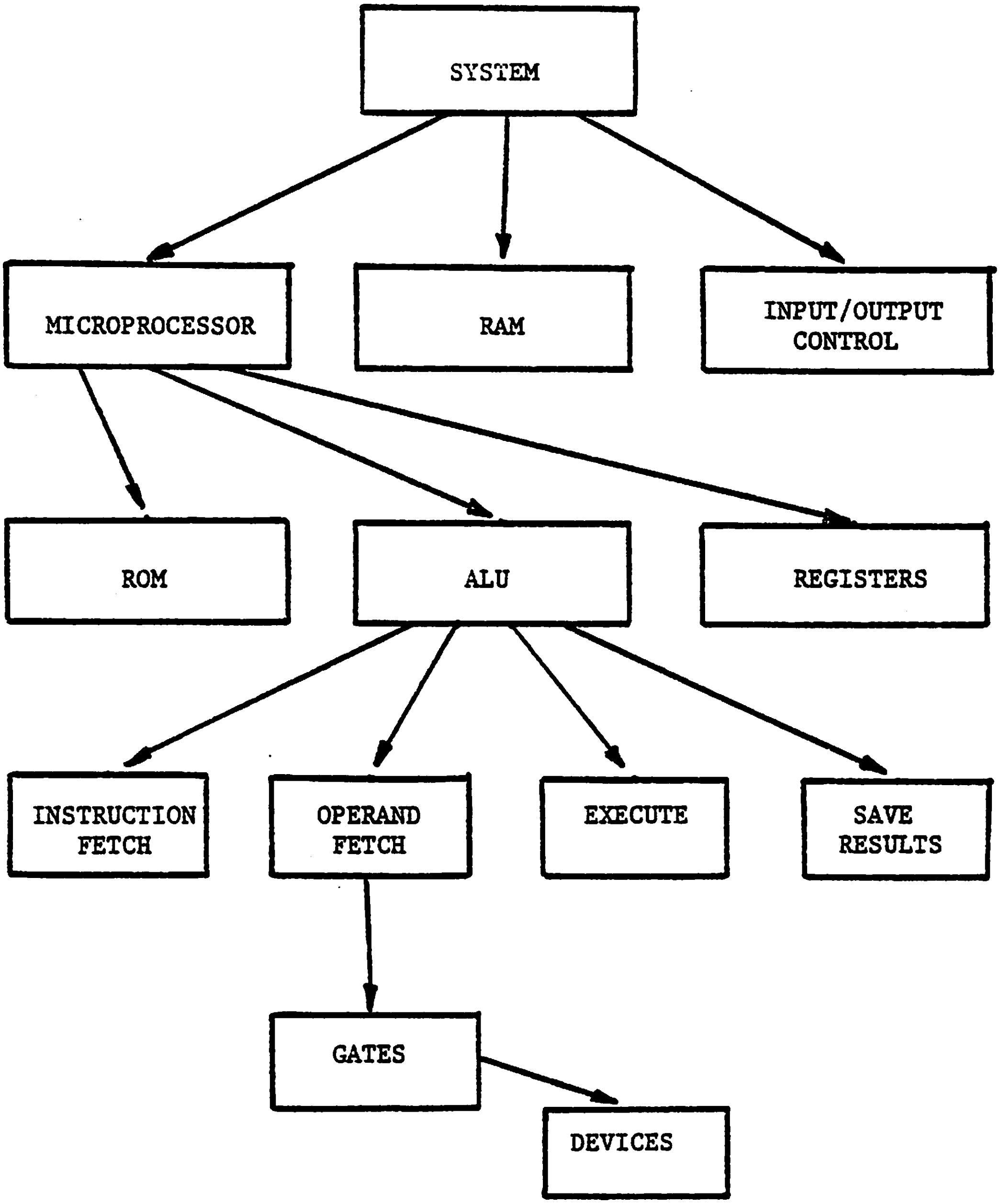


Fig.2.2 The Partitioning of a System Design.

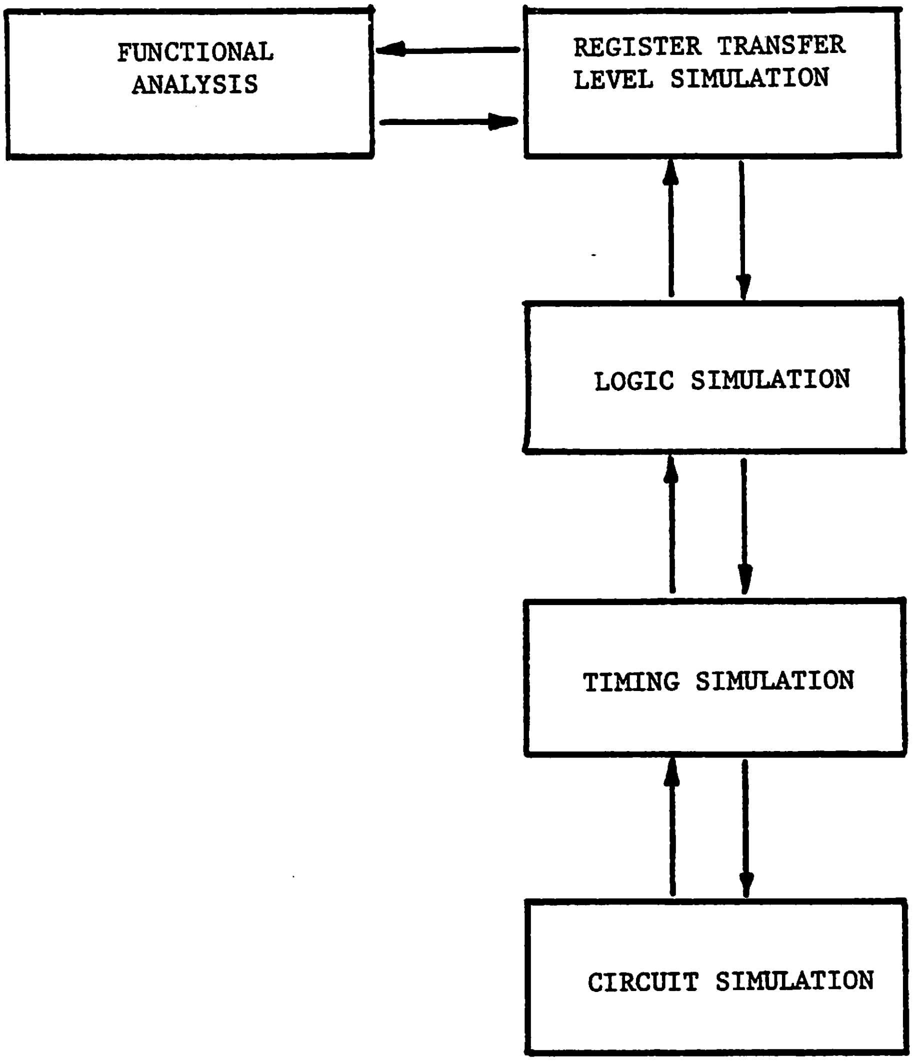


Fig. 2.3 The Hierarchy of Design Verification Tools.

described in terms of devices such as MOS transistors. The circuit analysis data is used by the designer to choose an appropriate topology for a logic gate-level description of the circuit block and to determine its parameters. This task may be performed interactively on a minicomputer or suitable intelligent terminal [15]. In this way, as the implementation of the circuit moves up the hierarchical tree of Fig. 2.2, the integrity of the circuit (its connectivity) and accuracy of the signal timing information for the various blocks in the design may be preserved.

The highest level of simulation shown in Fig. 2.3 is the Register Transfer Level (RTL) simulation. An RTL simulator uses the same form of signal description used in a logic simulator but provides higher-level logic models, such as registers, PLAs and Arithmetic Logic Units (ALUs), and higher-level structures such as parallel data buses, where a number of data lines are described as a single, parallel data path, both in the input description and during the analysis.

Note that *functional simulation* is not included directly in the verification process. Functional simulation is simulation at the algorithmic level and does not depend on the particular design implementation. It is the comparison of the RTL simulation and the original functional-level description of the circuit which finally verifies that the circuit design meets the specifications required by the original algorithmic description of the circuit function. This dissertation is not concerned with functional simulation, as mentioned earlier.

2.3. Analysis Techniques

2.3.1. Introduction The detailed analysis of integrated circuits in the time domain requires the solution of a set of first-order, nonlinear ordinary differential equations which describe the circuit and its associated signal sources. In the case of

Nodal Analysis [16], the node voltages may be expressed in compact vector form as

$$\dot{v} = f(v,t) \tag{2-1}$$

where v_i, $i=1,\ldots,n$ are the node voltages. For MOS circuits, the nonlinear devices include driver transistors, loads and transmission gates and the principle energy storage element is the capacitance at a node. A detailed analysis of circuit performance requires the accurate solution of this set of differential equations.

2.3.2. Circuit Analysis A nodal or modified-nodal [17] circuit analysis program such as SPICE2 solves the initial value problem of Eqn. 2-1 until the successive difference in the computed node voltages between analysis iterations is less than 0.1%. The device models are relatively complex and describe accurately the terminal characteristics of the nonlinear devices which make up the integrated circuit.

In a circuit analysis program the numerical solution of Eqn. 2-1 is performed in two steps. First, the solution time interval T is divided into small time steps where each increment h is called the *stepsize*. This is shown for a single node in Fig. 2.4. The set of nonlinear algebraic difference equations, derived below, is then solved numerically.

At any time t_n where the node voltages are known, the node voltages at time t_{n+1} may be obtained *explicitly* from those already computed at t_n by using a Taylor-series expansion at t_n. If only the first term of the expansion is used the method is called the Forward-Euler algorithm where

$$v_{n+1} = v_n + hf(v_n, t_n). \tag{2-2}$$

Explicit methods such as this require very small values of the timestep h to maintain accuracy and stability [16]. For this reason they are seldom used in circuit

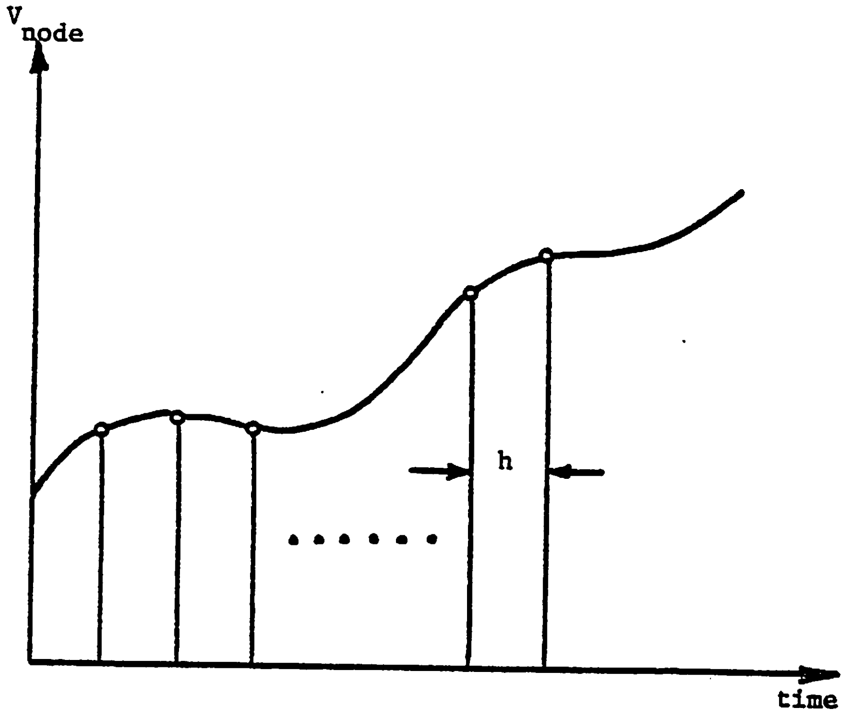


Fig. 2.4 The solution period is partitioned into steps of width h.

simulation programs.

Another way of obtaining the value of the node voltages at t_{n+1} is to use a polynomial approximation to the voltage waveform at each node. In this case the values of the node voltages at previous timepoints are used to predict the value of the voltages at t_{n+1}. These predicted voltages at t_{n+1} may then be used to obtain a better approximation for the node voltages at t_{n+1} in a similar manner. This results in an *implicit* solution method. The two implicit methods of interest here are the first-order Backward-Euler method where

$$v_{n+1} = v_n + hf(v_{n+1}, t_{n+1}) \tag{2-3}$$

and the second-order Trapezoidal method:

$$v_{n+1} = v_n + \frac{h}{2}[f(v_{n+1}, t_{n+1}) + f(v_n, t_n)]. \tag{2-4}$$

The second part of the circuit analysis concerns the solution of the set of non-linear algebraic equations which result from the application of the difference methods described above. Eqns. 2-2, 2-3 and 2-4 may each be written in the standard form:

$$v = V(v). \tag{2-5}$$

where v is the vector of node voltages at t_{n+1}. The subscript n+1 has been dropped for clarity and is assumed below. The method most commonly used for the solution of Eqn. 2-5 is the *Newton-Raphson* algorithm [16]. Eqn. 2-5 may now be written in the form

$$j(v) = 0 \tag{2-6}$$

and assuming an initial choice for the voltages at t_{n+1} of v^0 then the mth iteration of the Newton-Raphson algorithm may be written as

$$v^{m+1} = v^m - [J(v^m)]^{-1} j(v^m) \qquad (2\text{-}7)$$

where $J(v^m)$ is the Jacobian matrix of $j(v^m)$. Eqn. 2-7 then becomes:

$$J(v^m)v^{m+1} = J(v^m)v^m - j(v^m) \equiv i(v^m). \qquad (2\text{-}8)$$

This is the set of linear algebraic equations solved during the circuit analysis.

The flow diagram of a typical circuit analysis program is shown in Fig. 2.5. After an initial choice for the node voltages v_0^0 is made at time $t=0$(A), the nonlinear device models are evaluated to obtain the matrix entries for both the Jacobian matrix $J(v^m)$ and the right-hand-side equivalent current vector $i(v^m)$ in Eqn. 2-8 above (B). The contributions from linear elements, such as time-invariant capacitors, resistors and voltage and current sources, are also loaded into the matrix at this time. The set of linear equations are then solved using an efficient sparse matrix algorithm [1] (C) and the new voltages are compared to the previous estimate (D). If the process has not converged this loop is repeated until convergence is obtained. Typically 3 to 5 iterations are required per timepoint once the initial solution (at $t=0$) has been obtained.

Once convergence is obtained, the error introduced by the Trapezoidal rule approximation is estimated (E). This estimation may be done directly, using a Local Truncation Error (LTE) scheme [1], [16], or indirectly by counting the number of iterations required for convergence in the Newton-Raphson loop. For the analysis of linear or weakly-nonlinear circuits the former method must be used. For highly nonlinear circuits (e.g. digital circuits), LTE estimation algorithms are generally too conservative and the *iteration count* method is far more effective (see App. 9). If the solution at a timepoint is not satisfactory, the timestep is reduced and the loop is repeated until an acceptably small error or number of

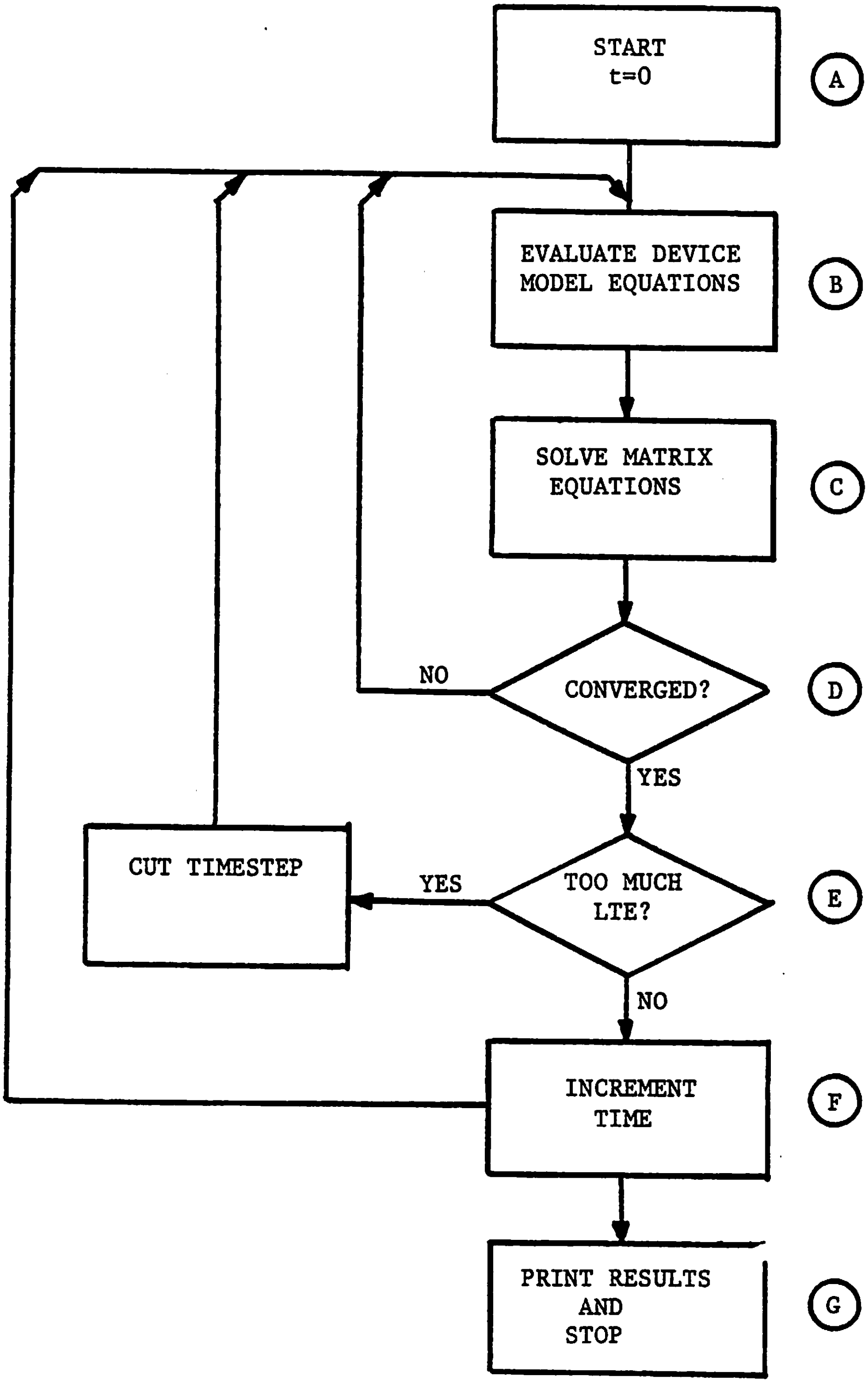


Fig. 2.5 Flow Diagram for Circuit Analysis

iterations is obtained. Time is then incremented (F) and the entire process is repeated until the requested simulation period is over (G).

Fig. 2.6 shows the percentage of time spent in the major subroutines of program SPICE2 for the analysis of the binary-to-octal decoder circuit of Fig. 2.1. It is evident from this analysis that almost 80% of the total analysis time is spent in the evaluation of the device model entries for the Jacobian matrix and the right-hand-side current vector. A large fraction of the remaining time was spent in the evaluation of LTE and the integration of the capacitor currents using the Trapezoidal Rule. Techniques which can be used to reduce these times are described below and in Chap. 3.

2.3.3. Timing Analysis Timing analysis [7] is a simplified form of circuit analysis which takes advantage of the properties of digital circuits to reduce the simulation time and memory requirements of the analysis. Timing simulators are less accurate than circuit simulators. Node voltages may be as much as 5%-10% in error but the timing information provided by the simulator is within a few percent which is sufficient for most digital design problems.

The various simplifications made in a timing analysis are described in terms of the circuit simulation algorithms presented above. Details of the algorithms used for timing simulation are included in Chap. 3.

As mentioned above the evaluation of device model equations for each device in the circuit at each iteration in the analysis can account for a large percentage of the total analysis time. In a timing simulator the model equations are replaced by a table of values and model evaluation consists of looking up the values of the matrix entries in the table. A detailed description of the table models used in

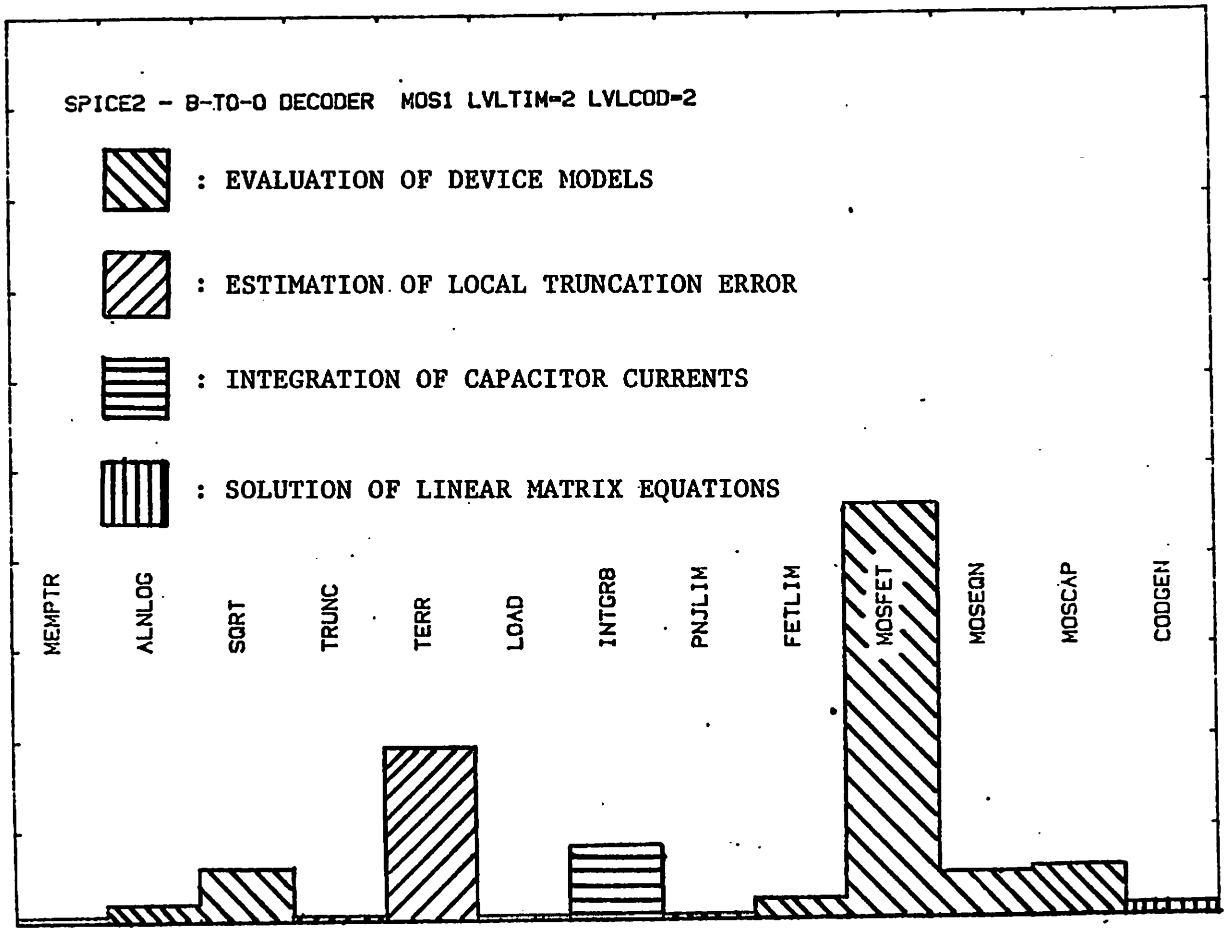


Fig. 2.6 Time Spent in the Major Subroutines of SPICE2 for the Analysis of the Binary-to-Octal Decoder Circuit.

program MOTIS-C and SPLICE is included in App. 1. For example, the drain-to-source current I_{DS} of an MOS transistor may be obtained from

$$I_{DS}(V_{DS}, V_{GS}, V_{BS}) = T_D(V_{DS}, V_{GS}, V_{BS}) \tag{2-9}$$

where V_{DS}, V_{GS}, and V_{BS} are the controlling branch voltages and T_D is a table containing values of I_{DS} spanning the expected range of the branch voltages. For most computers the table look-up scheme is much faster than the equivalent equation evaluation and the model accuracy is still consistent with the accuracy of the overall analysis. The table model requires more storage than an equivalent equation model would; however, the memory requirements of the table may be reduced substantially if a number of simple transformations are used (App. 1). Table 2.2 shows a comparison of model evaluation time and Newton-Raphson iterations required for convergence for the three models available in SPICE2 [18], [19] and a table look-up model of the type described in detail in App. 1. This table model used 100 steps for each controlling voltage to span a range of voltage of ± twice the maximum power supply voltage. These results are based on the analysis of the circuit of Fig. 2.1 on a CDC 6400 computer.

For circuit analysis the table values must be interpolated to avoid convergence problems due to table step discontinuities. In timing analysis interpolation is generally not used because only a single Newton-Raphson step is taken at each timepoint and hence dc convergence is not a problem.

The next simplification used in timing analysis is to replace the circuit matrix solution of Eqn. 2-8 with a simple vector product. This is accomplished by *decoupling* the circuit equations for the evaluation of the node voltages at t_{n+1} by using previously computed values of voltage for the evaluation of node coupling terms.

model	cp/d/it (ms)	%cp load	it/tp averg	cp/pp (ms)	cp/pp norm
GP	2.9	81	4.5	2.6*	12*
EM1	2.3	77	4.7	2.3*	10*
MOS3	9.8	88	3.7	2.2	10
MOS2	4.3	79	3.1	1.3	6
MOS1	3.7	78	4.1	0.9	4
MOS0	0.24	21	4.6	0.22	1

*normalized by device count (48/88)

cp : central processor time for analysis

d : active device

it : Newton-Raphson iteration

tp : time point

pp : user requested print point

Table 2.2 Results for the Analysis of the Binary-to-Octal Decoder Circuit using a variety of Device Models. MOS1, MOS2, and MOS3 are the Models Available in SPICE2. MOS0 is the Table Look-Up Model.

This results in an explicit analysis of the coupling while the solution at the node may still be implicit in form. The process may be clarified by considering Eqn. 2-8 evaluated using voltages at the previous timepoint t_n rather than the previous Newton-Raphson iteration m:

$$J(v_n)v_{n+1} = i(v_n). \tag{2-10}$$

If $J(v_n)$ is partitioned into two parts

$$J(v_n) = D(v_n) + O(v_n) \tag{2-11}$$

where $D(v_n)$ contains the diagonal entries and $O(v_n)$ the off-diagonal entries of $J(v_n)$:

$$D(v_n) = \begin{bmatrix} j_{11} & 0 & \cdots & 0 \\ 0 & j_{22} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & j_{NN} \end{bmatrix} \tag{2-12}$$

$$O(v_n) = \begin{bmatrix} 0 & j_{12} & \cdots & j_{1N} \\ j_{21} & 0 & \cdots & j_{2N} \\ \cdots & \cdots & \cdots & \cdots \\ j_{N1} & j_{N2} & \cdots & 0 \end{bmatrix} \tag{2-13}$$

where N is the number of circuit nodes or equations. Eqn. 2-10 may be written:

$$[D(v_n) + O(v_n)]v_{n+1} = i(v_n) \tag{2-14}$$

$$D(v_n)v_{n+1} + O(v_n)v_{n+1} = i(v_n). \tag{2-15}$$

If the coupling terms $O(v_n)$ are evaluated using the voltages at the previous timepoint t_n then Eqn. 2-15 becomes

$$D(v_n)v_{n+1} + O(v_n)v_n = i(v_n) \tag{2-16}$$

$$D(v_n)v_{n+1} = i(v_n) - O(v_n)v_n \qquad (2\text{-}17)$$

since $O(v_n)v_n$ is known at time t_{n+1}. The left-hand-side of Eqn. 2-17 may now be replaced by a vector product to give

$$g(v_n)v_{n+1} = k(v_n) \qquad (2\text{-}18)$$

where $g(v_n) = [j_{11}, j_{22}, j_{33}, \cdots, j_{NN}]$ and $k(v_n) = i(v_n) - O(v_n)v_n$.

Eqn. 2-18 is in the form used in timing simulators MOTIS, MOTIS-C and SIMPIL [14]. Although explicit analysis is less stable than an implicit scheme of the same order [16], the unilateral nature of most digital circuits and the voltage limiting of logic levels 1 and 0 in MOS and I^2L circuits make this approach practical. The use of explicit coupling techniques and their effect on the stability and accuracy of the analysis are described in Chap. 3.

Single Iteration: As mentioned earlier, only a single Newton-Raphson step is used to approximate the solution of the nonlinear difference equations. This is satisfactory because the device model equations are relatively smooth and slowly-varying functions. Accuracy is maintained in MOTIS by using a global fixed timestep h, typically 1ns, which is small enough to keep the change in device model operating-point between successive timepoints within acceptable limits. MOTIS-C and SPLICE use variable timestep algorithms for timing analysis. MOTIS-C selects a timestep which keeps the change in node voltage between any two successive timepoints small (less than $\frac{1}{64}$ of the supply voltage range) while SPLICE monitors the change in device currents between timepoints. The latter scheme is described in detail in Chap. 4.

The difference equation used for the integration of capacitor currents varies between simulation programs. MOTIS and SIMPIL use the Backward-Euler scheme of Eqn. 2-3, MOTIS-C uses the Trapezoidal algorithm of Eqn. 2-4 and the timing analysis part of SPLICE uses a modified form of the Forward-Euler method, Eqn. 2-2, for the integration of grounded capacitor currents. Non-grounded (floating) capacitors cannot be treated implicitly if the node decoupling scheme is used. Techniques for integrating floating capacitors are described in Chap. 3 and App. 2.

Flow Diagram: The combination of the simplifications described above constitutes timing analysis and the flow diagram of a typical timing simulator is shown in Fig. 2.7. After an initial guess for the node voltages is made at time $t=0$ (A), the contributions to the node conductance vector $g(v_n)$ and the right-hand-side equivalent current vector $k(v_n)$ in Eqn. 2-18 are looked-up in the model tables (B). The contributions from linear elements such as time-invariant capacitors, resistors and voltage and current sources are also loaded into the vectors at this time. The vector division is then performed to obtain a new set of node voltages (C), time is incremented and the loop is repeated until the requested simulation period is over. The simplicity of this scheme is apparent when Fig. 2.7 is compared with Fig. 2.5, the flow diagram for circuit simulation.

If table models are generated for individual transistors, timing simulators may be between one and two orders of magnitude faster than circuit analysis programs, with less than 10% error in the node voltages. The actual speed improvement depends on the type of circuit under analysis. If the circuit contains a node or nodes which may change voltage levels very rapidly, a very small timestep must be used. In this case the program is relatively slow. Four-phase MOS circuits often fall into this category. This problem is alleviated when a variable timestep scheme is used. In MOTIS-C, the timestep is chosen to limit the maximum voltage change at all circuit nodes between timepoints. As circuit size increases, the probability

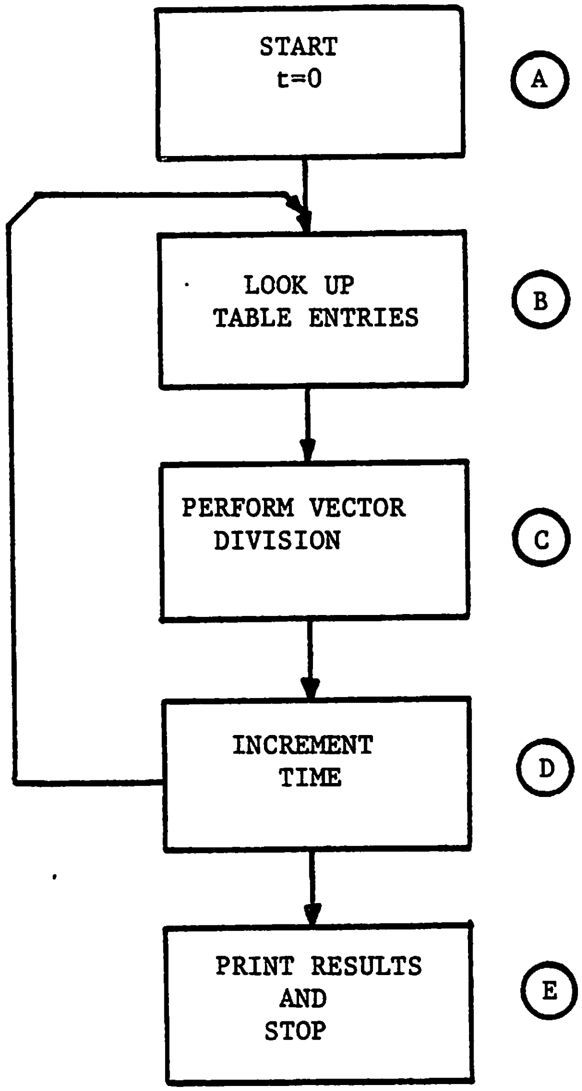


Fig. 2.7 Flow Diagram for Timing Analysis

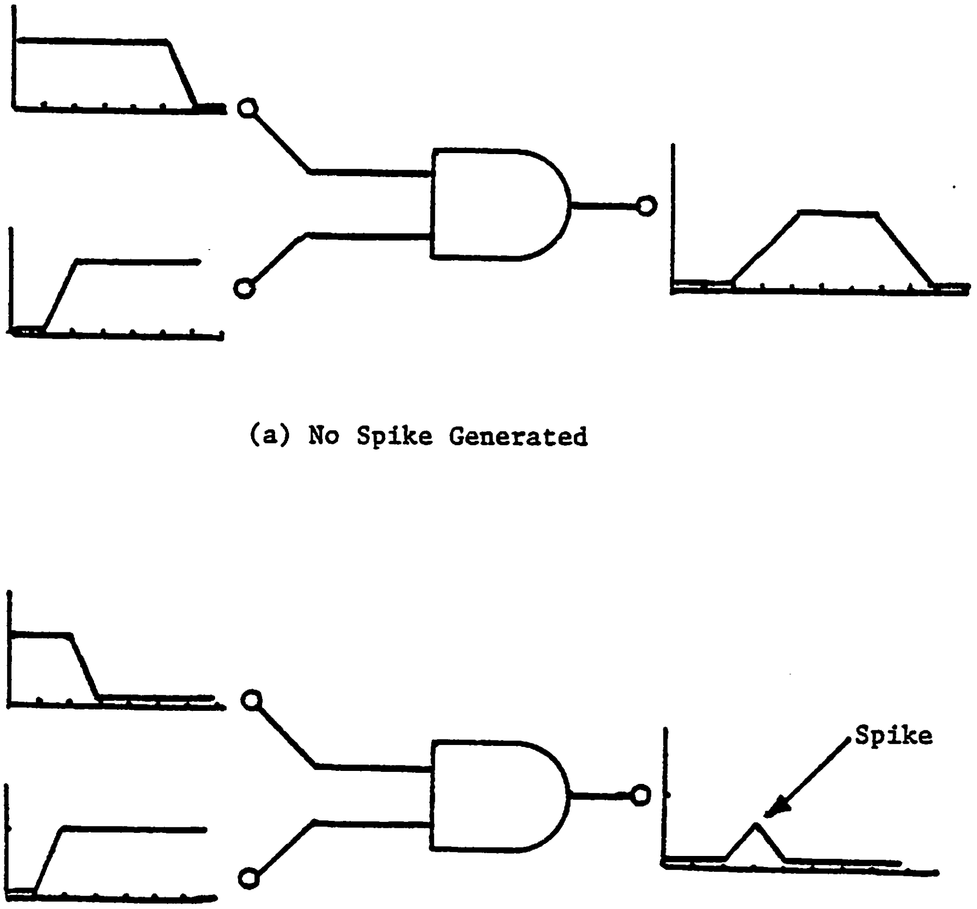
that at least one node in the circuit is changing rapidly at any time increases. Thus if the timestep is the same for every node in the analysis it is often maintained at a small value for most of the analysis. A technique which overcomes this problem is used in SPLICE and is described in Chap. 4.

Macromodels: LSI circuits often contain many groups of transistors which perform the same function (e.g. logic gates in a digital circuit). If the designer or computer program can identify these blocks before the analysis, further speed improvements can be obtained by exploiting known characteristics of the group of devices. The simplified model of a group of transistors which perform a specific circuit function is called a *macromodel* [20]. For example MOTIS-C contains a variety of macromodels for logic gates and a data latch (D-type flip-flop). The macromodels may use the existing device look-up tables or generate their own, as in the case of the CMOS inverter macromodel in MOTIS-C [8]. SIMPIL uses a macromodel for each multi-collector I^2L gate. With the use of such macromodels up to three orders of magnitude of speed improvement over conventional circuit analysis has been obtained, with comparable waveform accuracy (App. 8), [14].

2.3.4. Logic Analysis A logic analysis may be viewed as a simplified timing analysis where a number of discrete voltage levels are used rather than a continuous voltage range. The level denotes the logic condition at a node and in the simplest logic simulator these conditions are logic-1 ('1'), logic-0 ('0') and all other conditions are denoted by the *unknown* logic state ('*'). For MOS circuits an additional state is often used to simplify the analysis of tri-state gates and logic buses. This is called the *high impedence* state ('H'). For worst-case and other forms of logic analysis other states may be added to denote such conditions as signal *rising,* *falling, about-to-rise* and so on [3].

Rather than model individual transistors, groups of devices which perform a logic function are modeled as a single block. This is similar to the timing macromodels described in the previous section. These models may include simple gates such as NAND, NOR and inverter, or more complex functions such as flip-flops and registers.

Some logic simulators can only analyze combinatorial circuits. That is, time delays through the signal paths are not included in the gate models. Other logic simulators allow *unit delays* where all logic gates have the same delay and still others have *assignable delays* where the designer can assign specific delays to any of the gates used in the simulation. For MOS circuits where rise and fall times of gates may be quite different an assignable delay simulator with the ability to assign different rise and fall delays to each gate is required. In general only one logic state change may propagate through a gate at any one time. Not until it has reached the output of the gate can a second change begin. This type of delay is sometimes called an *inertial* delay. Should the input change again before the gate output has reached its new value a logic *spike* is produced and may or may not be propagated depending on the simulator being used. The generation of a spike is illustrated in Fig. 2.8. Most simulators print a warning message when spikes occur should the user request it. If more than one logic state may propagate through the device at the same time, a transmission line delay is required. This requires a variable-length queue for each delay and is generally not used except in special cases, such as for long signal path delays. Transmission line delays may be generated artificially in many simulators by connecting a number of buffer gates in series.



(a) No Spike Generated

(b) Spike Generated at Gate Output

Fig. 2.8 An Example of Spike Generation In Logic Analysis.

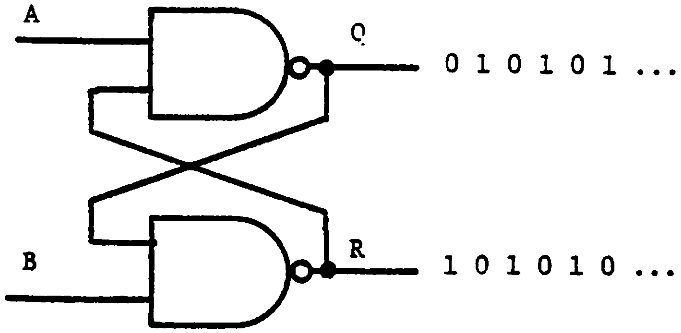
Even in assignable delay logic simulators the delays may only be integer multiples of a fundamental time quantum. This quantum may represent 0.5ns, for example, in which case a gate of delay 10 units would have an effective delay of 5ns.

The unilateral nature of logic gates is fundamental to the operation of logic simulators. The inputs of gates sample the logical values of the nodes to which they are connected and then the gate determines the logical values of its own outputs. Inherently bidirectional elements such as MOS transmission gates are difficult to implement in a logic simulator other than by using a unidirectional approximation to the gate.

Just as with timing simulation, it is the unidirectional nature of the gates which maintains the stability of the analysis. Tightly-coupled gates can still cause problems. Consider the NAND latch of Fig. 2.9. If the initial conditions are as shown in the table at t=0, and each gate has a unit delay, the logic outputs will oscillate. Many logic simulators can detect such oscillations during the analysis and attempt to correct the problem by *holding* a node until the oscillation settles, or halting the analysis and advising the user of the problem.

In many logic simulators tri-state gates or *wired-or* circuits also require special attention.

Due to the many simplifications described above, logic simulators can achieve speeds of more than three orders of magnitude faster than circuit-level simulators. The decoupling of logic nodes by the logic gates also permits the use of algorithms which detect the logic gates that may change at any given time and ignore the remainder of the circuit (gates where no input changes have taken place at this



INITIAL CONDITIONS:

A : 1
B : 1
Q : 0
R : 1

Fig. 2.9 An Example of a Critical Race in Logic Analysis

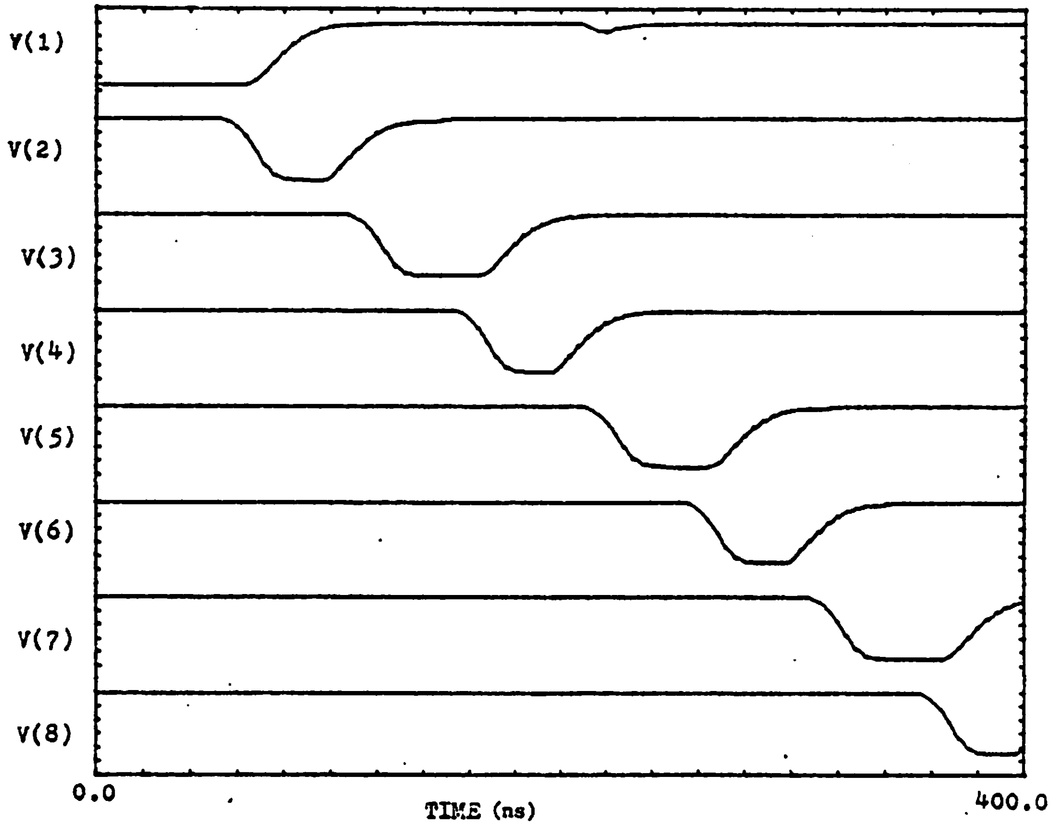
time). This process is called *selective trace* or *event-driven* analysis and is described further in Chap. 4. Since in most large digital networks less than 20% of the nodes are changing at any one time selective trace algorithms can enhance execution speed greatly.

The simplifications made in logic simulation also reduce the accuracy of the analysis. The output waveforms for the analysis of the circuit of Fig. 2.1 for circuit and logic analysis are shown in Fig. 2.10(a) and Fig. 2.10(b). Note that the logic levels have been scaled to match those of the circuit analysis and the logic unknown states are shown as rising or falling lines as appropriate. The relative speeds of the analysis are also included. Both analyses were performed with program SPLICE.

2.4. Hybrid Analysis

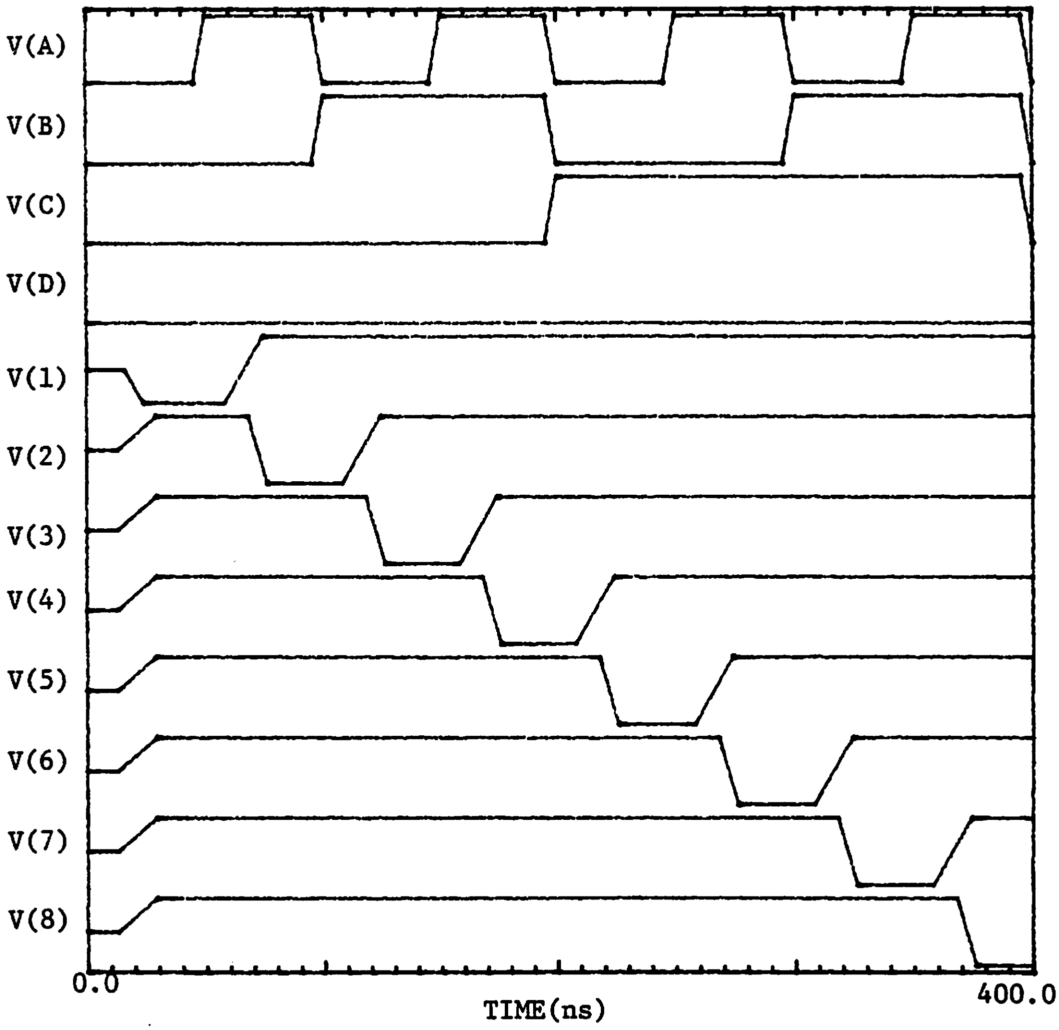
Modern circuit analysis programs require a great deal of computer time for the analysis of LSI circuits. In many of these circuits the detailed accuracy provided by a circuit simulation program is not required for the entire circuit under investigation but only in some areas of the circuit. This is particularly true of digital circuits where often a gate-level logic analysis provides sufficient information about the performance of much of the circuit while other parts, such as transfer gate clusters in MOS circuits [15], require more detailed modeling and analysis.

By providing a range of models, from highly accurate circuit-level device models for critical parts of the network to less accurate models which describe larger pieces of the circuit, the designer can reduce the simulation time significantly by choosing the computationally less expensive models wherever it is appropriate.



Simulation time: 237seconds.

Fig. 2.10(a) Circuit analysis of the Binary-to-Octal Decoder.



Simulation time: 0.3seconds

Fig. 2.10 (b) Logic Analysis of the Binary-to-Octal Decoder

For example, consider the dynamic RAM circuit shown in block form in Fig. 2.11. To predict the circuit performance accurately it is necessary to analyze each sense amplifier with a detailed circuit-level analysis. The high loop gain of the amplifier makes a decoupled timing analysis unsatisfactory due to the equation decoupling scheme used in the timing analysis. For storage transistors and data input/output circuits a timing analysis provides the voltage waveform information required, while the row and column decoding functions are modeled adequately by a pure logic analysis.

Another property of LSI circuits which may be exploited in the analysis is their relative inactivity. Typically less than 20% of an LSI circuit is changing state at any one time. In a circuit simulation program, where a single matrix is used to describe the network, the entire circuit must be re-computed at each analysis point even when only a small percentage of the entries are changing in the matrix describing the LSI circuit. Just as sparse matrix techniques reduce the memory requirements and analysis time for circuit analyses so algorithms must be used which take advantage of this relative inactivity of LSI circuits to reduce the simulation time.

Hybrid analysis programs allow the designer to use a combination of analysis techniques and models, from circuit-level device models to logic-level gate models in the same simulation program. Such programs have provided up to three orders-of-magnitude reduction in simulation time and substantially lower memory requirements than conventional circuit simulation, while still providing a detailed circuit-level analysis where necessary [10] (App. 8).

There are many ways that circuit, logic and timing analysis may be combined in a single analysis program. The simplest approach is to combine existing

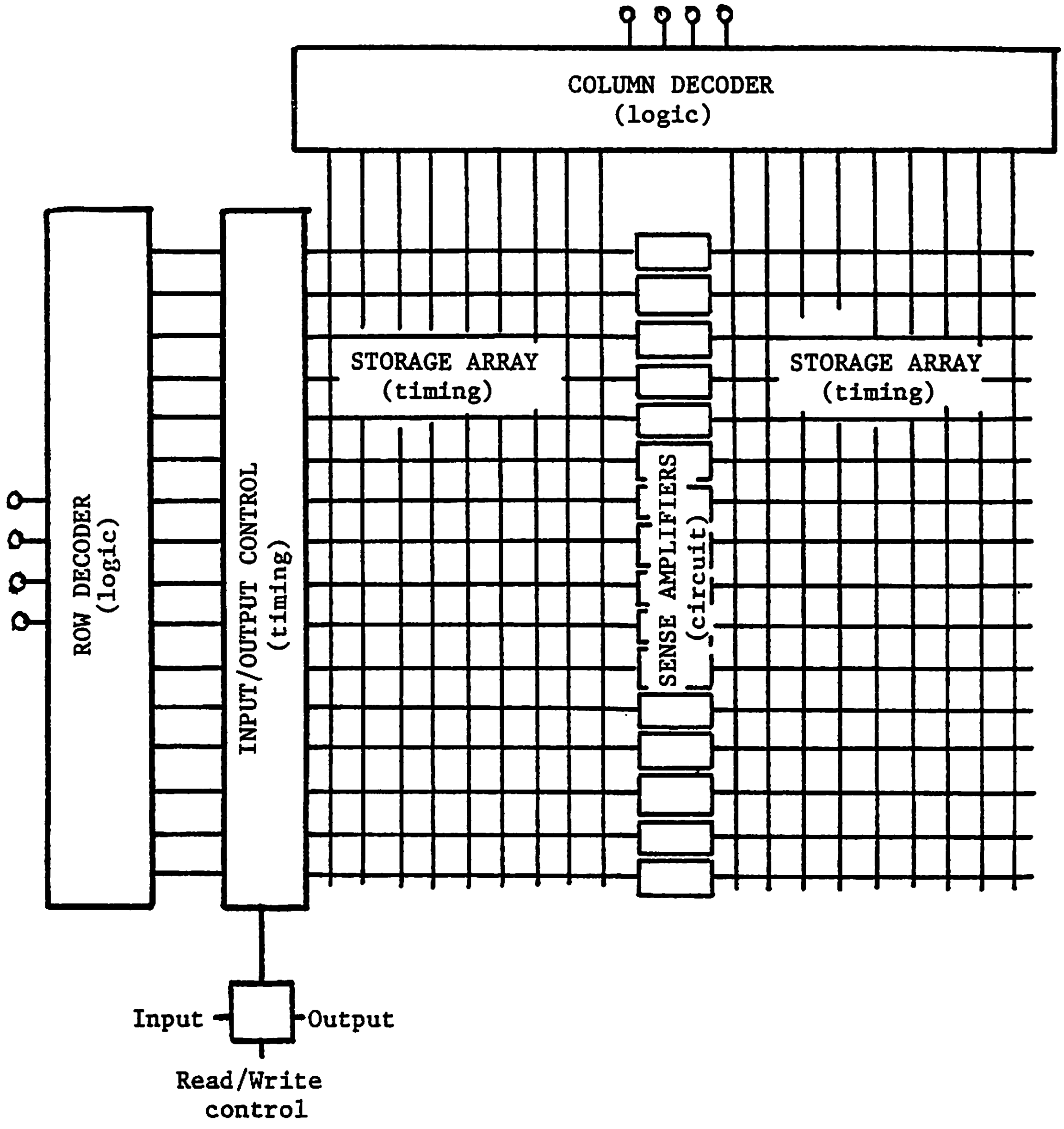


Fig. 2.11 Block Diagram of 256-by-1 bit Dynamic RAM

simulators via a data interface which transforms the circuit or logic variables into a form suitable for use by the other program(s). The block diagram of such a program, which combines circuit and logic analysis, is shown in Fig. 2-12. After the circuit has been analyzed for a short period of time the circuit node voltages are converted to equivalent logic levels with a *thresholding* process. A node voltage or branch current below a prescribed level is converted to a logic 0, above another prescribed value to a logic 1, for positive logic. Voltages or currents between these levels are propagated as unknown logic states ('*'). A logic analysis is then performed for those parts of the circuit which were described in terms of logic gates. After a short period of time, the logic nodes which are connected to the circuit-level devices are processed and used to control voltage sources, current sources or switches in the circuit analysis. This process is repeated for the duration of the simulation. Note that the circuit-level part of the analysis is included as a single block and thus the entire circuit-level part of the analysis must be performed at each analysis iteration. The program DIANA uses an analysis similar to this.

Another approach is to integrate the analysis algorithms in such a way that the circuit analysis may be partitioned into many small blocks, each of which may be processed independently. In the case of the RAM circuit above only one sense amplifier would be selected at any time. By partitioning each sense amplifier into a separate block for circuit-level analysis, only the selected sense amplifier need be processed. Program SPLICE permits such decoupling of circuit blocks whereas programs where the circuit analysis is performed as a single block must analyze all sense amplifiers at every analysis point. The algorithms used in SPLICE are described in detail in Chap. 4.

CHAPTER 3

ALGORITHMS FOR CIRCUIT, TIMING AND LOGIC ANALYSIS

3.1. Introduction

A number of approaches may be used in the design of circuit, timing and logic analysis programs. In each case tradeoffs are made between memory requirements, execution speed, model accuracy and simulation accuracy. In a hybrid analysis program, in which two or more forms of analysis may be performed concurrently, a number of additional constraints are applied. The degree to which these constraints influence the architecture of the hybrid program depends largely on the way in which the various components of the program communicate with one another.

This chapter describes certain of the critical algorithms used in circuit, timing and logic analysis programs as an introduction to the description of hybrid simulation presented in Chap. 4. Table look-up models for circuit analysis and the application of diakoptics or tearing to nonlinear circuit analysis are described. The use of equation decoupling in timing simulation and its effects on the stability of the analysis are included and time queue techniques for logic simulation are introduced.

3.2. Circuit Analysis

3.2.1. Introduction Circuit analysis algorithms for the time-domain transient analysis of medium and small scale integrated circuits have received a great deal of attention over the past decade. Most of the algorithms used in modern circuit simulation programs have been compared and described in detail by Nagel [1].

Recent work has focused on increasing the speed of circuit analysis, at the expense of some accuracy, and the application of circuit *tearing* techniques [27] to the solution of nonlinear networks. Tearing algorithms allow the circuit analysis to take advantage of the known inactivity of certain parts of the circuit at any time and, by using previously computed solutions for these inactive blocks, the simulation time can often be reduced substantially. These algorithms are described later in this section.

3.2.2. Table Models As shown in the previous chapter the evaluation of the nonlinear device model equations may account for over 80% of the total circuit simulation time. A table look-up model for MOS transistors similar to that described earlier for timing analysis was used in a version of circuit simulator SPICE2 for the analysis of large digital circuits. The tables were generated using the techniques described in App. 1. The two timestep control algorithms used in program SPICE2 (the LTE method and the iteration count method) were also compared and the effectiveness of the device *bypass* algorithm [1] was investigated. The bypass scheme enhances execution speed by monitoring the operating point (node voltages and branch currents) of each active device. If the operating point does not change significantly between Newton-Raphson iterations the device models are not re-evaluated but rather the matrix entries computed at the previous iteration are used again. All devices must still be checked at each iteration to determine whether the model equation evaluation may be bypassed.

The details of these investigations are included as App. 9. The results show that with the table model, analysis speed could be increased by as much as a factor of four over an analysis using the equivalent conventional analytic model. The iteration count timestep control scheme is far more effective than the LTE scheme for the digital circuits investigated and it is to be also noted that the error criteria used by the program in the Newton-Raphson iteration could be relaxed significantly before errors were observed in the voltage waveforms. The bypass scheme also proved very effective for digital circuits. Approximately 50% of all model evaluations were bypassed during the analyses. This is a far greater percentage than that observed for typical linear integrated circuits [1].

With all of the above techniques incorporated in the program the execution speed of SPICE2 could be increased by approximately a factor of twenty. This improvement is not sufficient for the economic analysis of LSI circuits.

3.2.3. Nonlinear Circuit Tearing A number of recent publications [21]-[23], [25] have extended and generalized Kron's method [27] of *diakoptics* or *tearing*. These extensions include the application of this approach to nonlinear networks and some strategies for choosing appropriate tearing interfaces [24]. Rather than repeat these general results, a description of Kron's technique for nodal analysis is presented and its application in the circuit analysis program MACRO [26] is described. Modifications of the diakoptic approach, such as co-diakoptics [22] and node tearing [23] are not described here.

Kron's method greatly reduces the dimensions of the matrices to be inverted during the analysis by dividing the network into a number of smaller subnetworks.

A more important outcome for the analysis of LSI circuits is that the approach partitions the subnetworks in such a way that only those that are active at any time need be analyzed.

The division of a given network is performed by detaching suitable tie branches so as to create a number of unconnected subnetworks. To compensate for the branches removed by this process, currents equal to those that were flowing through the branches are injected at the nodes from which they were disconnected. This method, based on the nodal admittance approach, is called diakoptics.

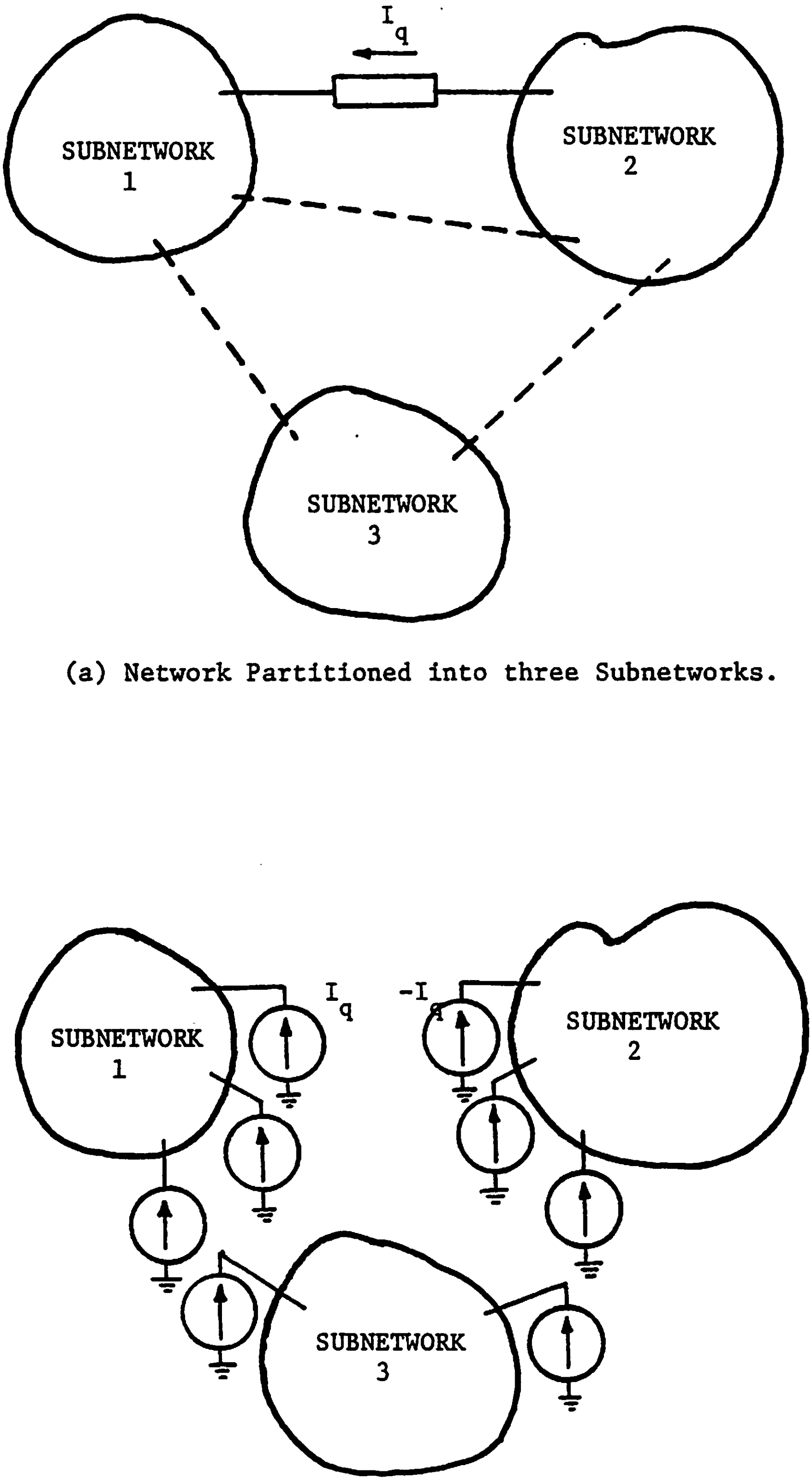
Consider the network shown in Fig. 3.1(a). Its corresponding admittance matrix may be written in the form:

$$Y = \begin{bmatrix} Y_{11} & Y_{12} & Y_{13} \\ Y_{21} & Y_{22} & Y_{23} \\ Y_{31} & Y_{32} & Y_{33} \end{bmatrix} \tag{3-1}$$

All the submatrices along the main diagonal are square and their off-diagonal elements depend only on the corresponding subnetwork. The elements of the off-diagonal submatrices Y_{ij}, $i \neq j$ consist of the tie or transfer admittances connecting nodes of the subnetworks i and j. The nodal equations for the network may be written in the form

$$Yv = i \tag{3-2}$$

where v and i are the vectors of node voltages and node currents, both of order N, respectively, where N is the total number of nodes in the circuit. By adding additional equations to the system, the tie admittances Y_{ij} can be removed from the Y matrix and the resulting block-diagonal matrix structure obtained is



(a) Network Partitioned into three Subnetworks.

(b) The Network After Tearing

Fig. 3.1 A Partitioned Network and its Torn Representation

$$\begin{bmatrix} Y' & -K \\ K^t & Z \end{bmatrix} \begin{bmatrix} v \\ b \end{bmatrix} = \begin{bmatrix} i \\ 0 \end{bmatrix} \tag{3-3}$$

where p is the number of branches removed, $Y'(N \times N)$ is the block-diagonal admittance matrix, $Z(p \times p)$ is the tie-branch impedance matrix, $K(N \times p)$ represents the tie connections, $b(p)$ is the vector of compensating currents to be injected and v and i are the unknown node voltages and node currents respectively. The number of torn branches p is much smaller than the total number of nodes N. For the example of Fig. 3.1(a) $p = 4$.

K has N rows, corresponding to each node of the original network, and p columns, corresponding to each of the tie branches. In column q of K, $+1$ will appear in the row corresponding to the node where $+I_q$ is injected and -1 in the row of the node at which $-I_q$ is injected, as shown in Fig. 3.1(b).

The form of the matrix on the left side of Eqn. 3-3 is bordered block diagonal and is shown schematically in Fig. 3.2. It can be shown that for a relatively sparse network and using efficient sparse matrix algorithms the number of arithmetic operations required to solve Eqn. 3-3 can be greater *or* fewer than those required for the solution of the original nodal system depending on the form of the network being simulated and the re-ordering scheme used to reduce fillin terms generated during the matrix decomposition process [22].

The major saving of such a scheme for circuit analysis comes from the fact that the LU factors for the individual Y_{ii} blocks need only be re-computed if the node voltages within the subnetwork change [29]. Since the evaluation of the admittance matrix entries consumes a substantial portion of the total circuit analysis time, and large networks are often relatively inactive, the potential savings can be quite significant.

On the other hand, if a bypass scheme of the form described in the previous section is used, the only time savings of a diakoptic analysis of this type is the evaluation of the LU factors. Unless the model evaluation time is reduced to the point where it is comparable to the time required for LU factorization and forward/backward elimination, diakoptic analysis on computer which can only perform one arithmetic operation at a time cannot provide a significant speed improvement over nodal analysis with bypass.

Another approach based on diakoptics, which is used in the circuit analysis program MACRO [26], [25] is to partition the analysis into two separate Newton-Raphson iterations which are coupled by a functional iteration as described in the previous chapter. In this case the block-diagonal Y' is inverted to give $Z' = (Y')^{-1}$. This inversion is performed symbolically to reduce computation time. Z' is then used to solve for the tie branch currents b in an independent Newton-Raphson loop. Once these have converged a new estimate for the node voltages is computed and the Y' matrix entries are re-computed for any subcircuits where the voltages have changed. The algorithm proceeds as follows: partition Eqn. 3-3 into two parts

$$Y'v = i + Kb \tag{3-4}$$

$$K^t v + Zb = 0 \tag{3-5}$$

Substitute Eqn. 3-4 into Eqn. 3-5 to obtain:

$$Zb = -K^t[Z'(i + Kb)]. \tag{3-6}$$

An initial estimate is made for b, i and Z' then Eqn. 3-6 is solved using a Newton-Raphson algorithm until b converges. Z' is fixed during this iteration and this is the major difference between this approach and conventional diakoptic analysis. Once the b values have been obtained, Eqn. 3-4 is used to solve for a new estimate of v

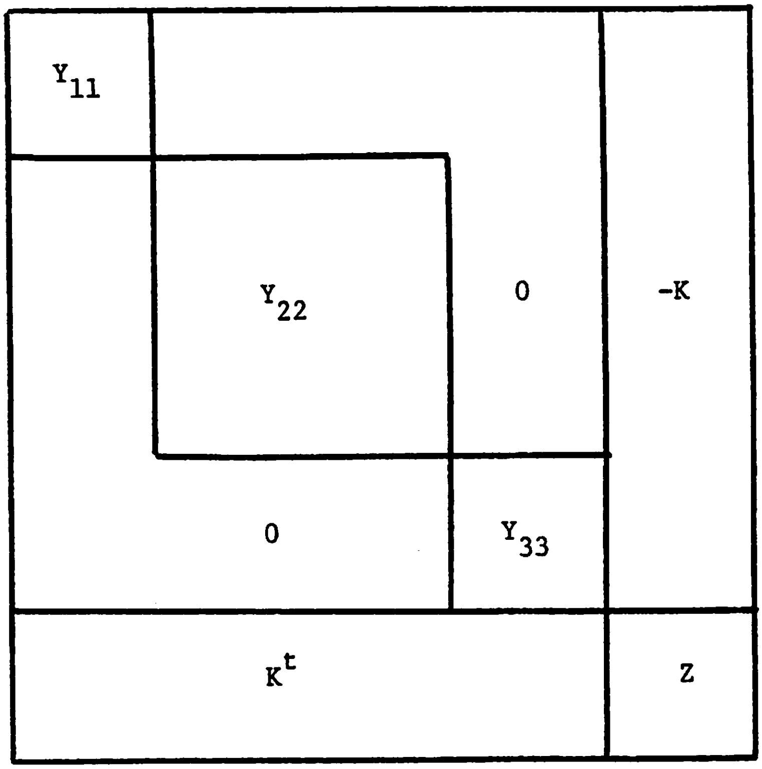


Fig. 3.2 Bordered Block-Diagonal Form of the Augmented Circuit Matrix.

and then the necessary entries of Y′ are re-computed. This outer Newton-Raphson loop is repeated until convergence is obtained for the node voltages.

Rabbat and Hsieh [28], [25] have also presented a scheme for detecting inactive subnetworks *a priori* by noting that only if a block connected to an already-inactive block changes state, can the inactive block become active. Since this technique is very similar to the selective trace approach used in logic simulation, it will not be described further here.

Many of the above approaches have the potential of reducing the numerical operation count for the solution of the linearized equations at a Newton-Raphson iteration. While device model evaluation is still the most time-consuming part of the analysis, diakoptic approaches alone cannot improve the speed of circuit analysis so that the simulation of LSI circuits becomes economical.

3.3. Timing Analysis

3.3.1. Introduction Many simplifications made to the circuit simulation algorithms introduced in the previous chapter to obtain a timing simulator. The algorithms described here apply to MOS timing simulation but most of the techniques are similar to those used in other timing simulators, such as SIMPIL [14] for I^2L circuits.

Since device model evaluation is generally the most expensive part of circuit analysis, timing simulators use table look-up models to reduce model evaluation time. This technique has been described earlier and a detailed description of the algorithms used in MOTIS-C and SPLICE are included in App. 1. Another important simplification in timing simulators is the reduction of the Newton-Raphson iteration loop and matrix solution to an explicit, single-iteration vector product as shown in Chap. 2. By using decoupling the circuit equations the program can take advantage of the low circuit activity of many LSI circuits and enhance execution speed. The penalty for this simplification is reduced stability of the analysis. Algorithms which allow the program to take advantage of this low circuit activity and techniques for overcoming some stability problems in timing simulators are described in this section.

3.3.2. Equation Decoupling As shown in Chap. 2, the use of node voltages from previous timepoints for the evaluation of coupling terms allows the use of the explicit, single-iteration-per-timepoint analysis of Eqn. 2-18.

$$g(v_n)v_{n+1} = k(v_n) \tag{3-7}$$

where $g(v_n) = [j_{11}, j_{22}, \cdots, j_{NN}]$ and $k(v_n) = i(v_n) - O(v_n)v_n$ as described in Sec. 2.3.3. This equation is in the form used in programs MOTIS, MOTIS-C and SIMPIL. Program SPLICE uses a variation of Eqn. 3-7 as described in Chap. 4. Eqn. 3-7 shows that the voltage at node m in the circuit at timepoint t_{n+1} may be computed directly from voltages evaluated at t_n:

$$g^m(v_n)v^m_{n+1} = k^m(v_n). \tag{3-8}$$

Hence the nodes are decoupled at time t_{n+1}. Eqn. 3-8 may be expanded to include the coupling terms in the form:

$$g^m(v_n)v^m_{n+1} = i^m_{n+1} - \sum_{j=1}^{N} O^{mj} f(v^j_k,\ k = n, n-1, \cdots). \tag{3-9}$$

O^{mj} contains the off-diagonal Jacobian entries as described in the previous chapter. Note that v^{jn} has been replaced by $f(v^j_k,\ k = n, n-1, \cdots)$ above. In MOTIS, MOTIS-C and SIMPIL, extrapolation is not used and hence:

$$f(v_k^j,\ k = n, n-1, \cdots) = v_n^j. \tag{3-10}$$

Accuracy is improved by evaluating the slope of the current characteristic at t_n and using the slope to estimate the current at t_{n+1}. If this were not used the algorithm would reduce to the Forward-Euler form of Eqn. 3-10. As it is, the programs use one iteration of the Backward-Euler method for the node voltages. Note that this technique is similar to the Jacobi method [30]. If $k = n+1$ is permitted in Eqn. 3-9 for nodes which have already been solved at t_{n+1} then the method is of the Gauss-Seidel form [30].

Since the equations have now been decoupled and each node voltage may be solved independently of the other nodes at t_{n+1}, techniques which exploit the inactivity of the circuit can be used. One approach is to monitor node voltages for a number of timepoints and if any node has not changed by a significant amount the solution for that node voltage may be *bypassed* at the next timepoint and its old value used instead. To do this effectively the program must have a facility for determining which other nodes are affected by the change in voltage at a particular node. For example in the circuit of Fig. 3.3, if the voltage at node 2 changes then node 3 must be checked to see if its voltage has changed. MOTIS performs this task by having a flag associated with each element. Should the operating point of any element change significantly at a timepoint, the flags associated with all elements to which it is connected are set and these elements are processed at the *next* timepoint. Note that this implies a numerical delay of at least one timestep but if the timestep is small these errors will not be significant. SIMPIL performs a fast logic simulation of each gate to determine whether it need be processed in detail at the current timepoint and MOTIS-C does not use a bypass scheme.

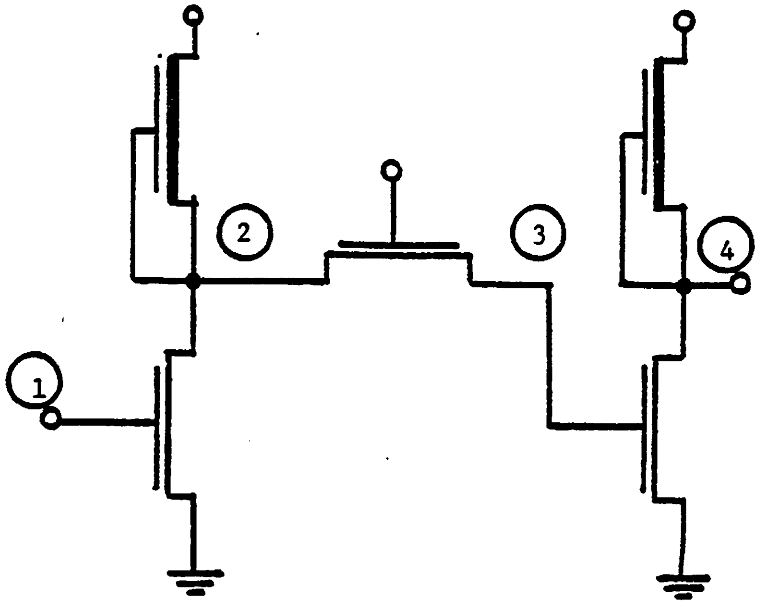


Fig. 3.3 Simple MOS Circuit

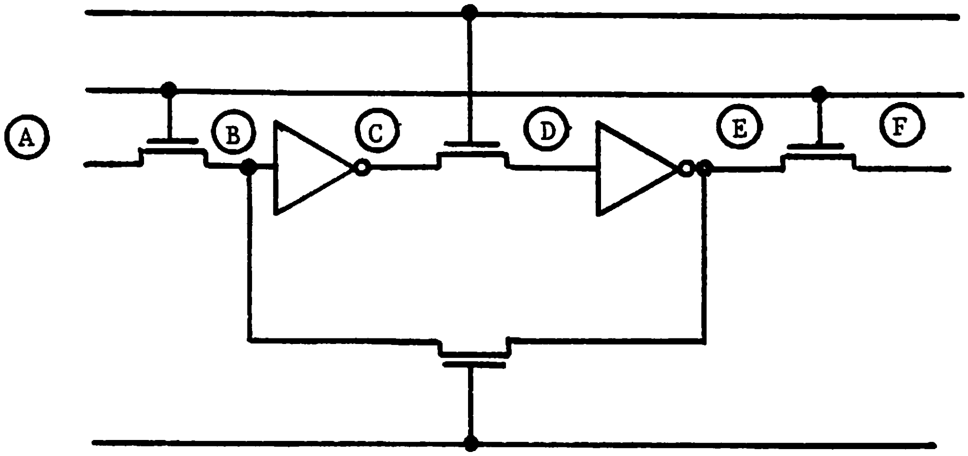
3.3.3. Single Iteration Circuit simulators use the Newton-Raphson procedure to solve the nonlinear algebraic difference equations at each timepoint (typically 2-5 iterations are required). If the equations are decoupled using the scheme described above, convergence could only be obtained using a functional iteration approach and would therefore be much slower in most cases. Rather than perform a number of iterations at each timepoint, a single iteration is used in timing analysis. Accuracy is maintained by reducing the timestep so that the linearized device models do not change significantly between timepoints. In MOTIS and SIMPIL this timestep is chosen prior to the analysis and must therefore be a conservative value. MOTIS-C uses a variable timestep scheme in which the initial guess for the timestep is based on the properties of the circuit (App.8), but at any time during the analysis the timestep may be adjusted by the program to limit the voltage change at all nodes in the circuit. This approach is acceptable for most digital circuits where accumulated voltage errors at a node are removed when the node voltage reaches logic "1" or logic "0" value. A variable timestep scheme also enhances the stability of the analysis as described in the following section.

With a global timestep and without a bypass scheme, the value of the timestep chosen by the program will be small for large circuits. As the size of the circuit increases the probability that the voltage at at least one node in the circuit will be changing rapidly increases as well. Since the equations are decoupled, it is possible to have different timesteps at each circuit node. Those nodes where there is little activity or where the voltages are changing slowly may use a large timestep while nodes switching rapidly can use a smaller stepsize. The voltage at inactive nodes used in the analysis of a node with a small timestep may then be extrapolated to estimate their value at the new time, for the evaluation of device models.

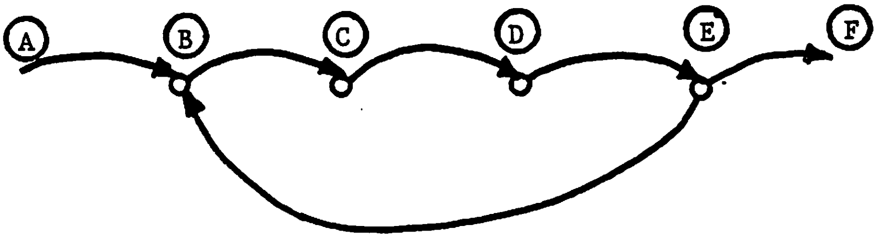
SPLICE uses such an approach and it is described in Chap. 4.

3.3.4. Stability Matrix iterative schemes such as the Jacobi and Gauss-Seidel methods are inherently unstable when the matrix has a weak diagonal [30]. In circuit terms, this corresponds to a network in which there is a strong bilateral coupling between nodes. For MOS digital circuits most of the node coupling may be represented by voltage-controlled current sources and their almost unilateral properties ensure local stability. It is possible, however, for loops to exist within the network where the circuit delay around the loop may be comparable to the analysis timestep. A simple example of such a circuit and its associated directed graph is shown in Fig. 3.4. A loop exists which could cause numerical instability if the analysis timestep were comparable to the loop delay. In this case the instability can be removed by reducing the timestep and hence increasing the self-admittance of each node in the analysis until the system is diagonally dominant and stable. This approach relies on the fact that their are grounded capacitors at nodes around the loop.

There are two cases which arise in MOS circuits where this approach is not sufficient. The first is the case of an MOS transmission gate where the drain and source nodes are coupled by the channel conductance. This conductance can be quite large when the device is in the conducting state. Techniques for the analysis of transmission gates in this situation are presented in [15] and App. 8. The second case is that of a floating capacitor. A technique which has been used successfully for the analysis of floating capacitors is included in App. 2. Both of these solutions require the use of a much smaller timestep than would otherwise be necessary.



(a) Section of an MOS Shift Register



(b) Directed Graph showing the signal loop

Fig. 3.4 Circuit with Potential Timing Instability if Clock Signals Overlap.

3.4. Logic Analysis

Most gate-level logic simulators belong to one of two general types. The first is based on the Huffman model [31] shown in Fig. 3.5. In this case, the gate description of the network supplied by the user is read by the program and any signal delays are factored out. The resulting combinatorial network is then ordered in terms of signal dependence. This ordering process includes the detection of certain pathological conditions in the network such as zero-delay loops. The analysis then consists of applying the input excitations to the network and following any signal path state changes through the network to the outputs. The delays are then applied to any secondary outputs and the analysis of the combinatorial block begins once again. The process is repeated until the requested input sequence has been completed.

This approach is very efficient for circuits where relatively few delays are significant in the operation of the circuit. When all gates have associated delays, performance will be degraded but the severity of the degradation depends greatly on the way the algorithm is implemented. SALOGS [2], [3] uses an algorithm of this form.

The second and more common approach is based on the use of a Time Queue (TQ) [32] as shown in Fig. 3.6. Each entry in the queue represents a discrete point in simulation time. Time moves ahead in fixed increments, determined by the user, which correspond to consecutive entries in the time queue. Each entry in the queue contains a pointer to a list of *events* which are to occur at that instant of time. An event is defined as the change of logical state of an output node of an element. The element may be a logic gate or an input signal source. The new state may or may not be the same as the state already held by the output line. If the

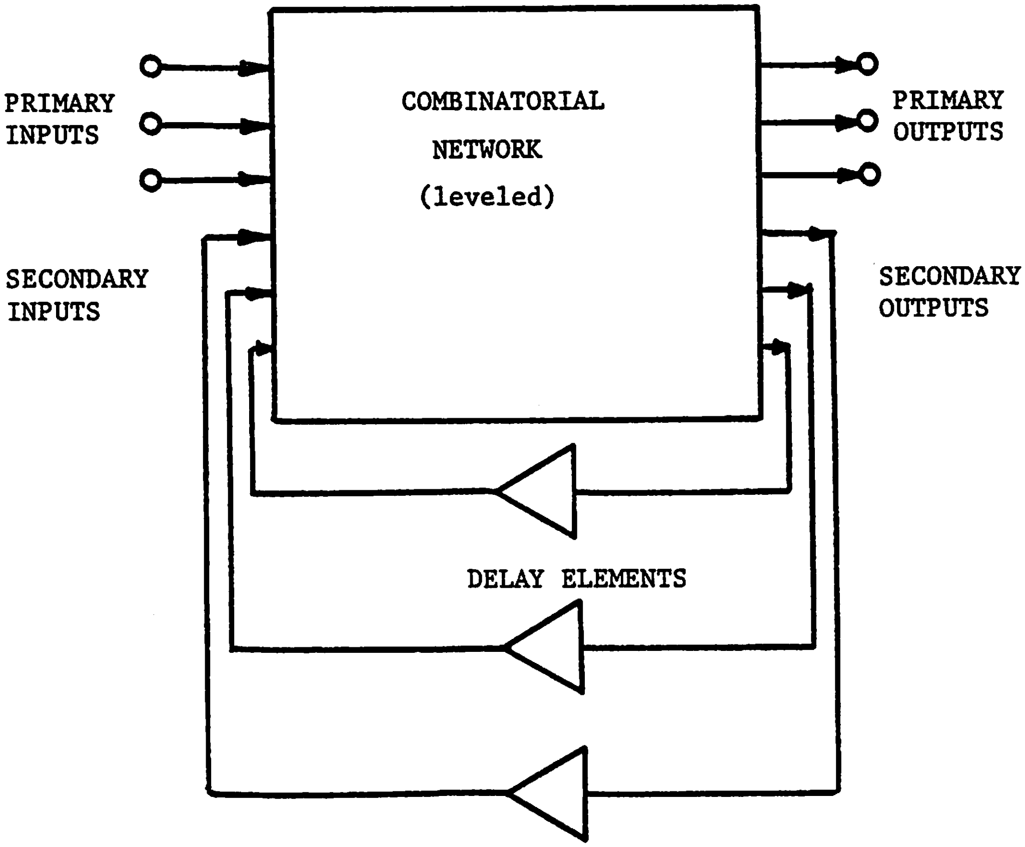


Fig. 3.5 Huffman Logic Model for Logic Analysis

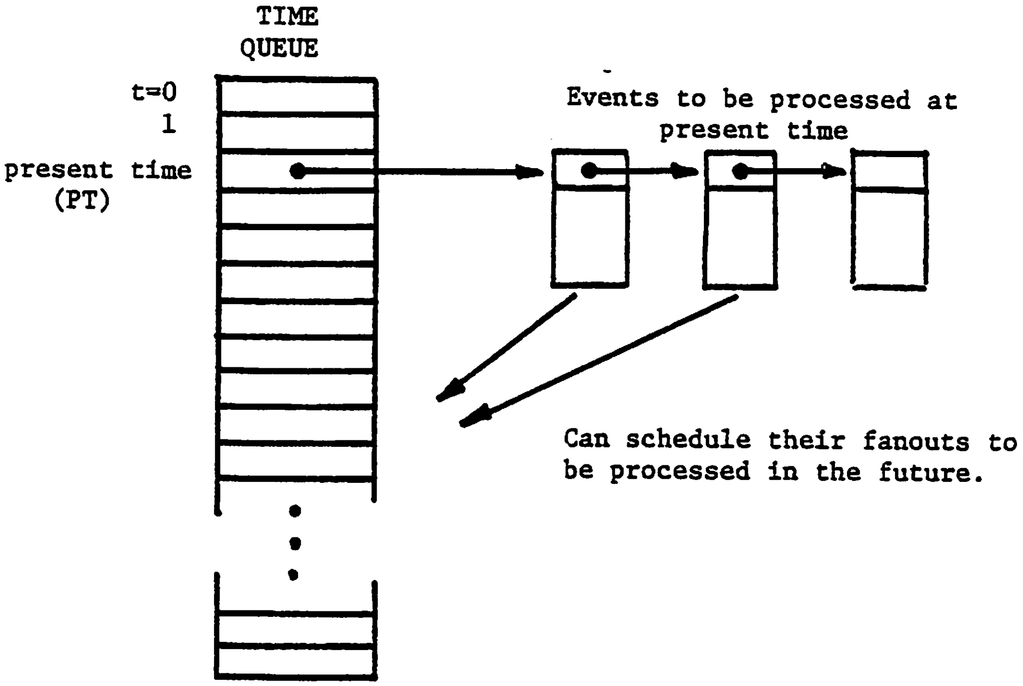

TIME
QUEUE
t=0
1
present time
(PT)
Events to be processed at
present time
Can schedule their fanouts to
be processed in the future.

Fig. 3.6 Principle of the Time Queue Simulator

new state is different from the old one, all elements whose input lines are connected to this output line must be processed to see if the change affects their outputs. These elements are called the *fanouts* of the output node and if the gate has only one output they constitute all the fanouts of the gate. Fig. 3.7 shows a simple circuit in which the NAND gate with output at node 1 has 3 fanouts. If the new state at the output line is the same as the old state then the fanouts need not be processed at this time. The algorithm used to determine whether the fanouts need processing at any time is called a *selective trace* algorithm as mentioned in the previous chapter. It is also referred to as event-driven analysis or dynamic leveling. For logic simulation no penalty in accuracy or stability is incurred with the use of selective trace.

When an output is evaluated and the new value is the same as the value already held by the node, the event is cancelled and the fanouts are not added to the time queue. If the new value is different, the event is executed by adding a list of all the elements which fan out from the node to the time queue. Each of these elements is then checked in turn to see if its outputs may have changed due to the change of state at its input and the process is repeated. Cyclic races can occur in this simulation. They are detected and the simulation is halted using the approach mentioned in the previous chapter.

The program module responsible for adding elements to the time queue is often called the *scheduler* and elements added to the time queue at time t are *scheduled* to be processed at time t. The scheduler is the heart of a time queue simulator and will be described in detail in Chap. 4 and Chap. 5.

In the time queue algorithm delays are included as part of each gate element and hence are not treated separately as in the first approach. Since for an accurate

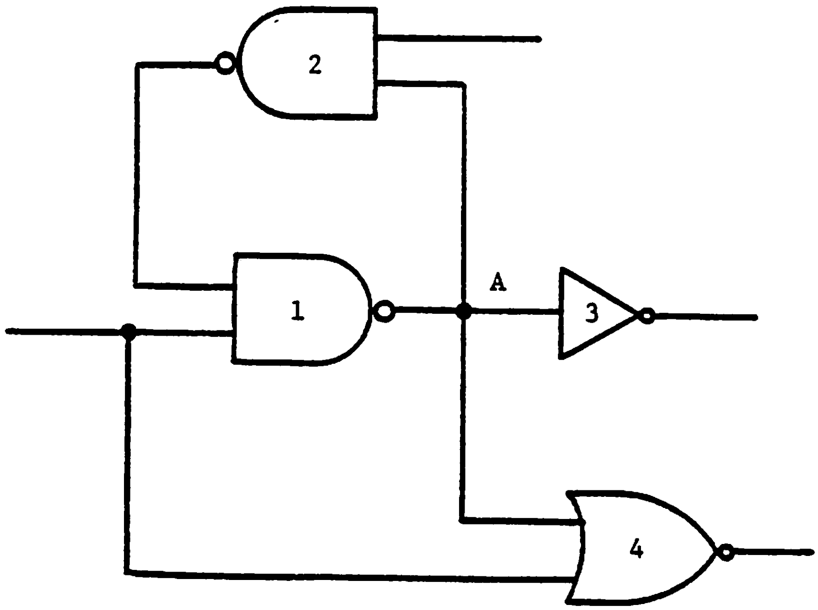


Fig. 3.7 NAND gate 1 has 3 fanouts at node A .

simulation most gates will have a delay associated with them this approach lends itself to a more efficient implementation.

Many logic simulators use the time queue approach (e.g. [4]-[6]) and this approach also has advantages for hybrid simulation as described in Chap. 4.

CHAPTER 4

HYBRID ANALYSIS AND SPLICE

4.1. Introduction

This chapter describes the algorithms used in the hybrid simulation program SPLICE. Program SPLICE can perform circuit, timing and/or logic analysis for MOS circuits. Parts of the circuit where simple logic functions are performed and the voltage levels are not critical can be described in terms of logic gates. Other parts of the circuit where MOS transfer gates are present, dynamic loading effects are critical to the circuit operation or where voltage levels are required, may be analyzed using a timing simulation. If the circuit contains blocks where a timing analysis is not satisfactory a circuit simulation may be performed. These blocks include circuit networks where strong feedback is present, such as sense amplifiers in a RAM or closed-loop operational amplifiers in an analog filter circuit. If floating capacitors are connected in series a circuit analysis may also be required. The circuit analysis is performed locally on a small group of devices and hence many separate blocks which require circuit analysis may be included in the input to the program.

Each portion of the circuit, whether it is described using logic gates, transistors used in a timing analysis or circuit elements, can communicate with the other parts of the circuit via the *hybrid interface* This interface converts logic levels to voltages and currents, or voltages to logic levels, according to the user's specifications. The hybrid interface is described in Sec. 4.6.

Consider the 256-by-1 bit dynamic RAM circuit shown in Fig. 4.1 and mentioned in Ch. 2. If this circuit were simulated using a complete circuit analysis, a great deal of time would be spent providing detailed waveform information about the input decoders and input/output circuits. At the same time, when only one sense amplifier is selected in a read or write operation, all sixteen amplifiers and the entire circuit array are processed. A bypass scheme or diakoptic approach as described in Chap. 3 may reduce the total analysis time but the cost of the analysis would still be prohibitive. A logic analysis would be much faster but would not provide any information about the operation of the RAM (access time, refresh time, etc.).

The input decoders can be simulated using a simple logic analysis to select the addressed row and column of the memory array. The logic outputs of the decoders may then be converted to voltages which control the input/output transistors and storage transistors. These transistors can be analyzed using a timing analysis since the voltage waveforms, charge stored on the bit storage capacitors and the circuit delays are important. The sense amplifiers are regenerative and hence require a circuit analysis for accurate prediction of their performance and to avoid numerical instabilities during the simulation. For this example the program would consider each sense amplifier separately and only analyze the active or selected one(s). The analysis results for this RAM example are included in Ch. 6.

With a hybrid analysis program the designer can select the form of analysis suitable for each part of the circuit. Each block of the circuit is processed separately (each gate in the logic analysis, each node in the timing analysis and individual circuit block for the circuit analysis). Blocks which are not active at any time are not simulated.

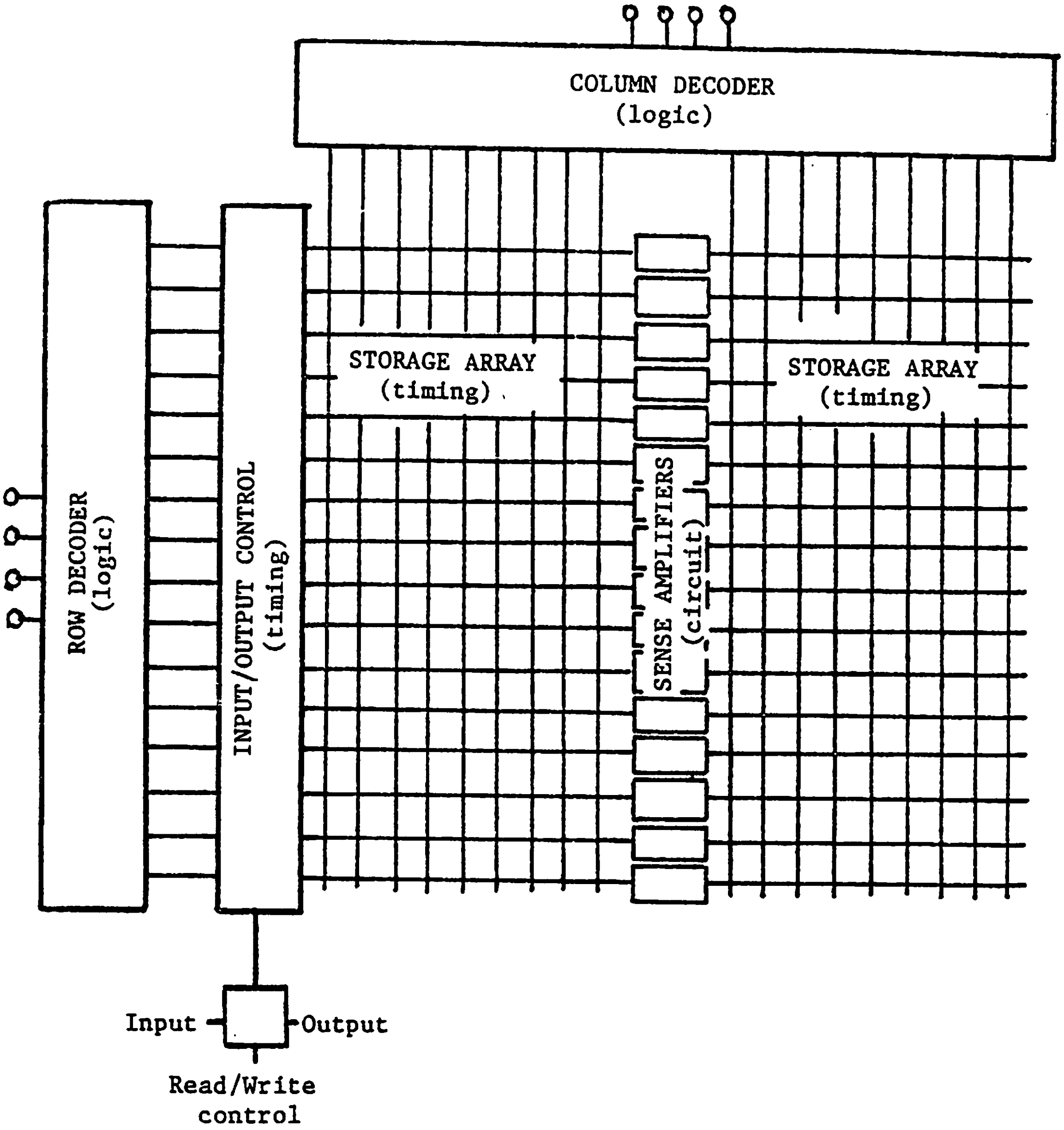


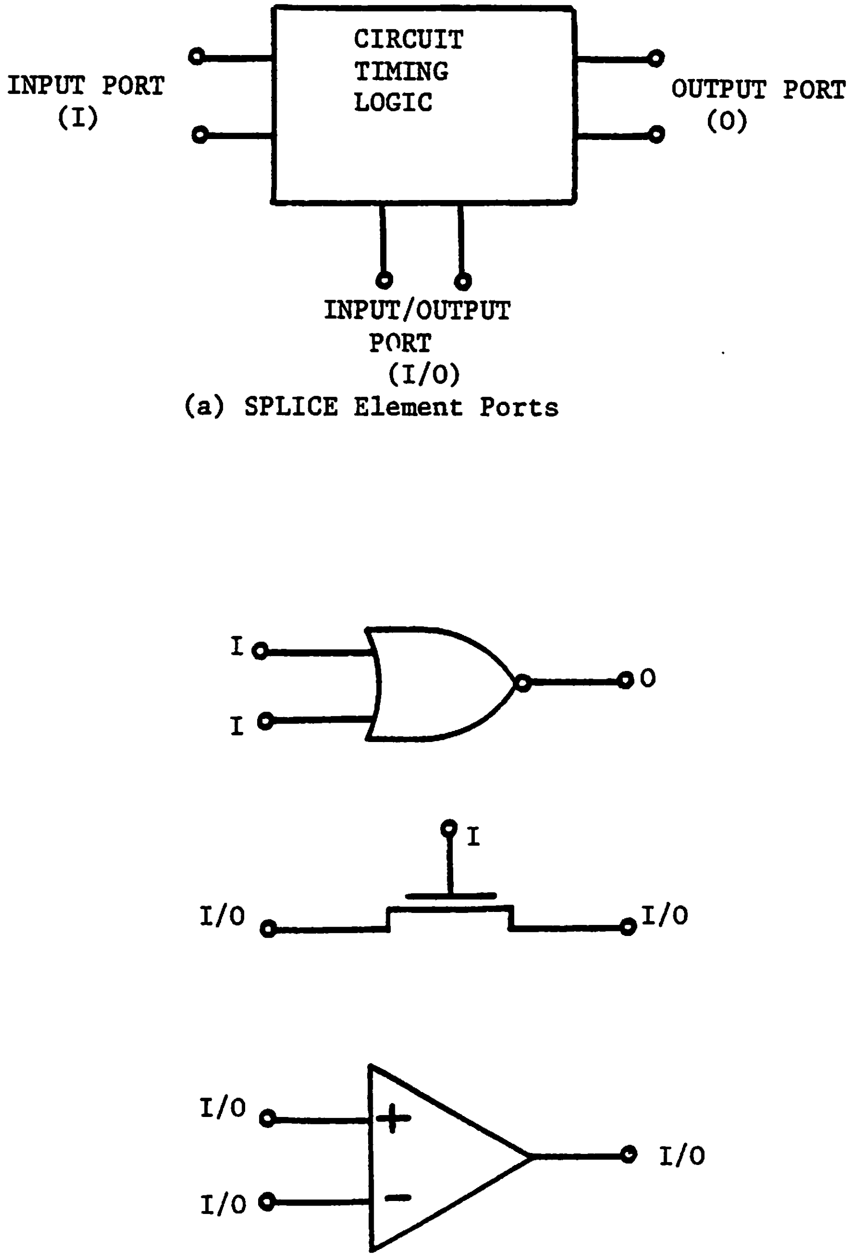
Fig. 4.1 Block Diagram of 256-by-1 bit Dynamic RAM

The analysis algorithms of SPLICE are controlled by an *event scheduler* of the type used in a time queue logic simulator. In SPLICE however, the events scheduled can be gates, as in the logic simulator, *and/or* timing elements or entire blocks of transistors which require a circuit analysis. In the remainder of this chapter the algorithms used by the event scheduler, the logic, timing and circuit analysis modules of SPLICE, and the hybrid interface are described.

4.2. The Event Scheduler

4.2.1. Basic Concepts The event scheduler used in SPLICE is similar to that used in a time queue logic simulator. As mentioned above, the elements the scheduler deals with are not only logic gates but timing transistors and circuit blocks as well. An *element* of the analysis is defined to be a logic gate, timing elements or group of connected transistors which constitute a circuit analysis block. Elements have three types of ports. *Input* ports (I), *output* ports (O) or *input/output* ports (I/O). Input ports sample the signal at the node to which they are connected but play no part in determining its value, as shown in Fig. 4.2. Output ports simply drive a node and the value of the signal at that node plays no part in determining the output of the element at that time, and input/output ports both sample the signal and then may change its value. Some examples of logic, timing and circuit elements and their ports are shown in Fig. 4.2.

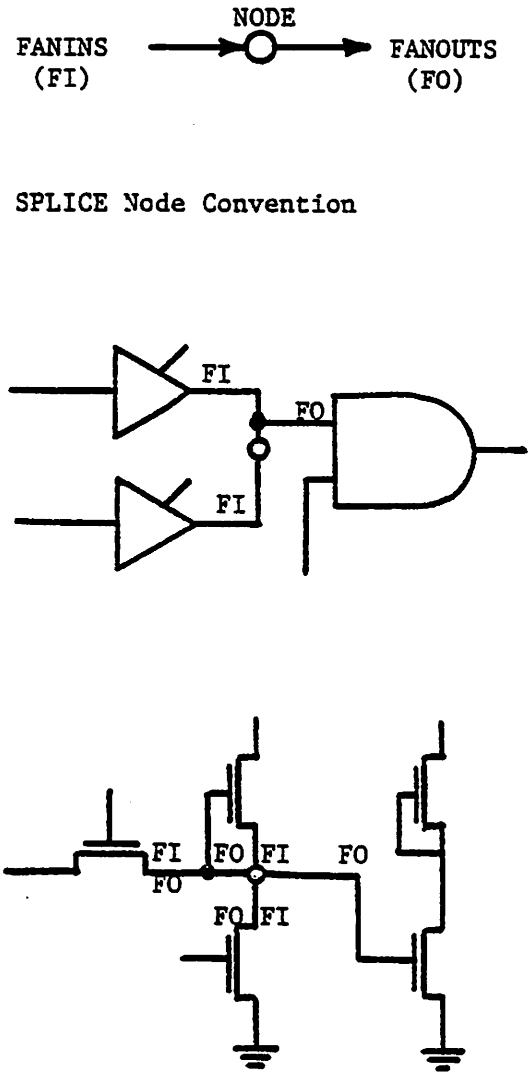
In SPLICE, the circuit nodes also have distinct properties, as shown in Fig. 4.3. A node may have *fanouts,* which are the connections to the input ports of elements, and *fanins* which are the connections to output ports of elements. A connection to an input/output port constitutes both a fanin and a fanout. Fanouts sample the signal level at the node and fanins can change the value of the signal at



(a) SPLICE Element Ports

(b) Examples of Element Ports.

Fig. 4.2 The Port Convention for SPLICE Elements.



(a) SPLICE Node Convention

(b) Examples of Fanins and Fanouts

Fig. 4.3 Node Convention used in SPLICE.

a node. Note that nodes internal to a circuit-analysis block are not the concern of the scheduler and hence not included here.

The efficiency of a time queue simulator depends on the particular data structures it uses. As events are scheduled to occur, they must be ordered in such a way that the scheduler can determine which event to process next. To make this process efficient the simulation time is broken down into small, uniform timesteps. The size of the timestep is the smallest non-zero delay which a logic gate may have and the delay of any gate must be an integral multiple of this timestep. This technique is used in SPLICE and the size of the timestep is called the *Minimum Resolvable Time* (MRT). For MOS circuits one unit of MRT is typically 1ns. In SPLICE, one unit of MRT is the minimum non-zero delay of a logic gate and the minimum time for which a circuit or timing element may be analyzed before its output change is propagated to the remainder of the network. The timestep may be reduced below one unit of MRT within the circuit or timing analysis, as described later in this chapter. One unit of MRT is also the smallest timestep between output events, such as print or plot requests made by the user.

4.2.2. Data Structure of Elements and Nodes Prior to the analysis, SPLICE generates a list of data for each element and node in the circuit. Fig. 4.4 shows the general structure of an element block. The first word of the data list contains a pointer to another data list which describes the element model (whether it is an MOS transistor, NAND gate, etc. and the parameters of the model such as delays, threshold voltage, etc.). Rather than store the element parameters in the element list itself, a separate model list is used. Since many elements in a large circuit are of the same type, by keeping all device parameters separate form the elements the

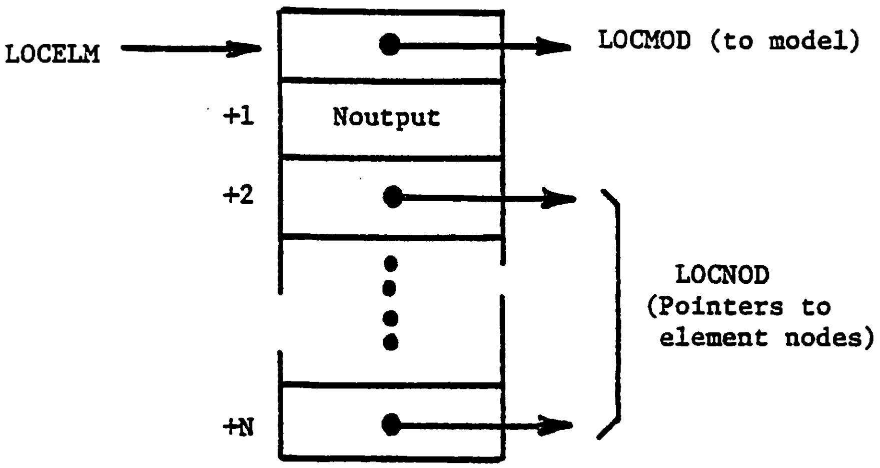


Fig. 4.4 General Structure of SPLICE element.

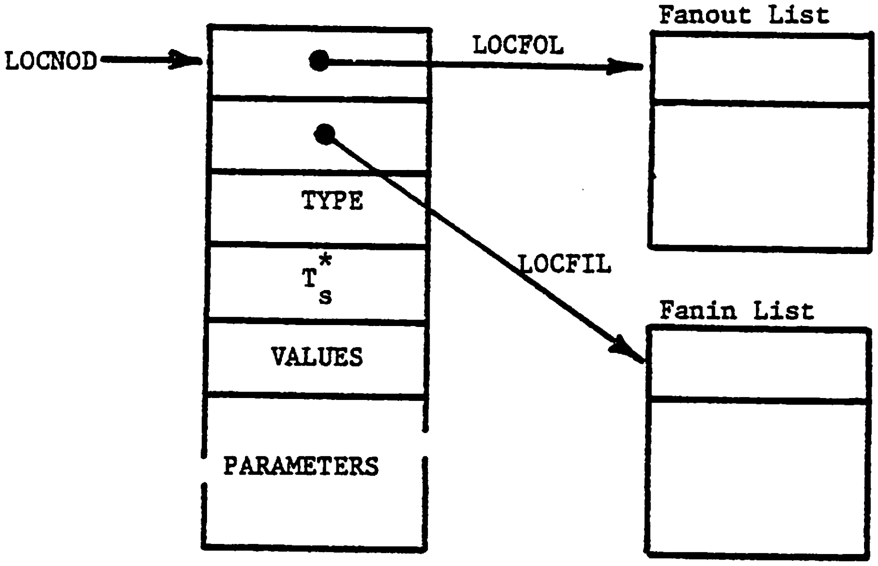


Fig. 4.5 General Structure of SPLICE node.

likelihood of storing redundant information is reduced.

The second word in the list contains the number of output plus input/output nodes the element has. This value is stored with the element for reasons of efficiency during the scheduling process.

The list then contains pointers to each of the node lists which correspond to the outputs of the element, followed by input/output node pointers and finally input node pointers. The data structure for a circuit block is more complex and is described in Chap. 5.

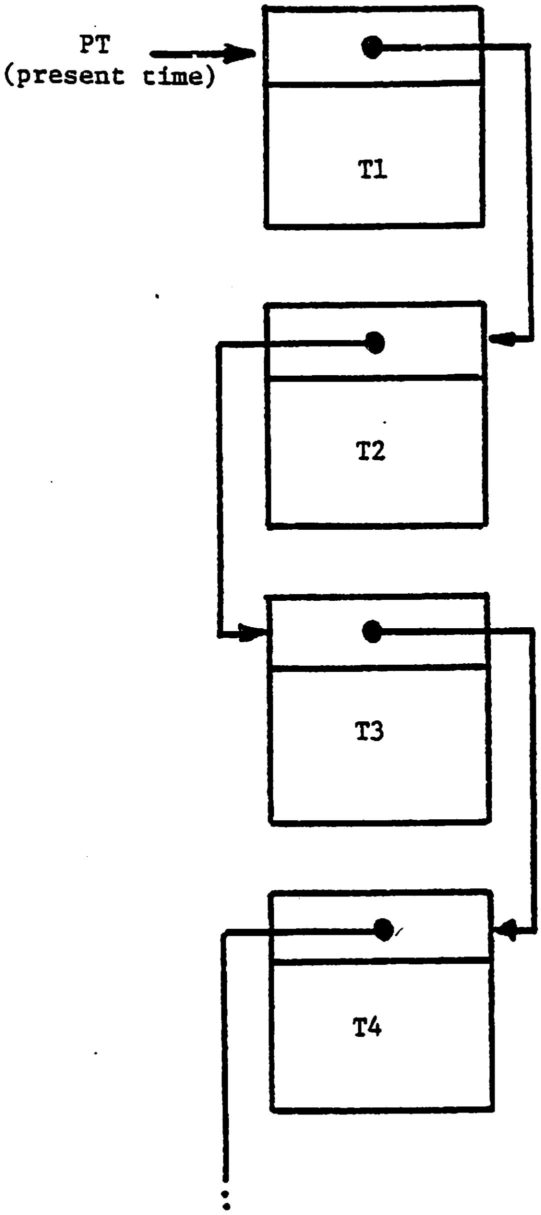
The general structure of a node is shown in Fig. 4.5. The first word of the list contains a pointer to a list of elements which fan out from the node. This list is called the fanout list and contains pointers to each element whose input or input/output port is connected to the node. The second word contains a pointer to a fanin list. This list contains pointers to elements whose output or input/output ports are connected to the node. The third word contains the node *type*. Nodes may be of type logic, timing, external-circuit and internal-circuit. External circuit nodes are those connected to timing elements. The fourth word contains T_s^*, the last time a which the nodes fanout list was scheduled to be processed. The remaining words contain the node voltages or logic levels at the previous two analysis points as well as a variety of parameters such as node capacitance, for timing nodes, and pointers to matrix entries for circuit nodes. A detailed description of these data structures is included in App. 5.

When an element is processed, its outputs are evaluated and if any of them have changed, the element node pointer is used to find the corresponding node list and the location of the fanout list is obtained from the first word of the node list.

The fanout list origin is then used by the scheduler to schedule the fanouts to be processed at the appropriate time in the future.

Thus the lists for an element contains the element connection information and a pointer to the model information. The node list contains information about the state of the signal level at the node, pointers to fanout and fanin lists, and some parameter information.

4.2.3. Data Structure of the Time Queue Efficient processing of the time queue is critical to the overall performance of the program. The time queue contains the fanout lists of all nodes scheduled to be processed and the time at which they are scheduled to be processed. A simple way of storing these entries is the linked list structure of Fig. 4.6. The scheduler moves along this time-ordered linked list where each entry in the list contains the time the fanouts of a node are to be processed and a pointer to the fanout list of the node. As each event is executed, the scheduler processes each element on the fanout list in an arbitrary order. Once the elements on th list have been processed, the simulator moves to the next entry in the time queue. It may contain a list to be processed at the same time as the last list or a list to be processed some time in the future. Any events generated as each list is processed are inserted in the time queue at the time they are due to occur. Unfortunately, if an event is scheduled to occur more than one unit of MRT into the future, the process of inserting it in the time queue may involve searching many entries already in the queue before its place can be determined and hence this scheme is relatively inefficient. By observing that most events occur within a few units of MRT from the present time (PT), a more efficient scheme may be used [5], [33].



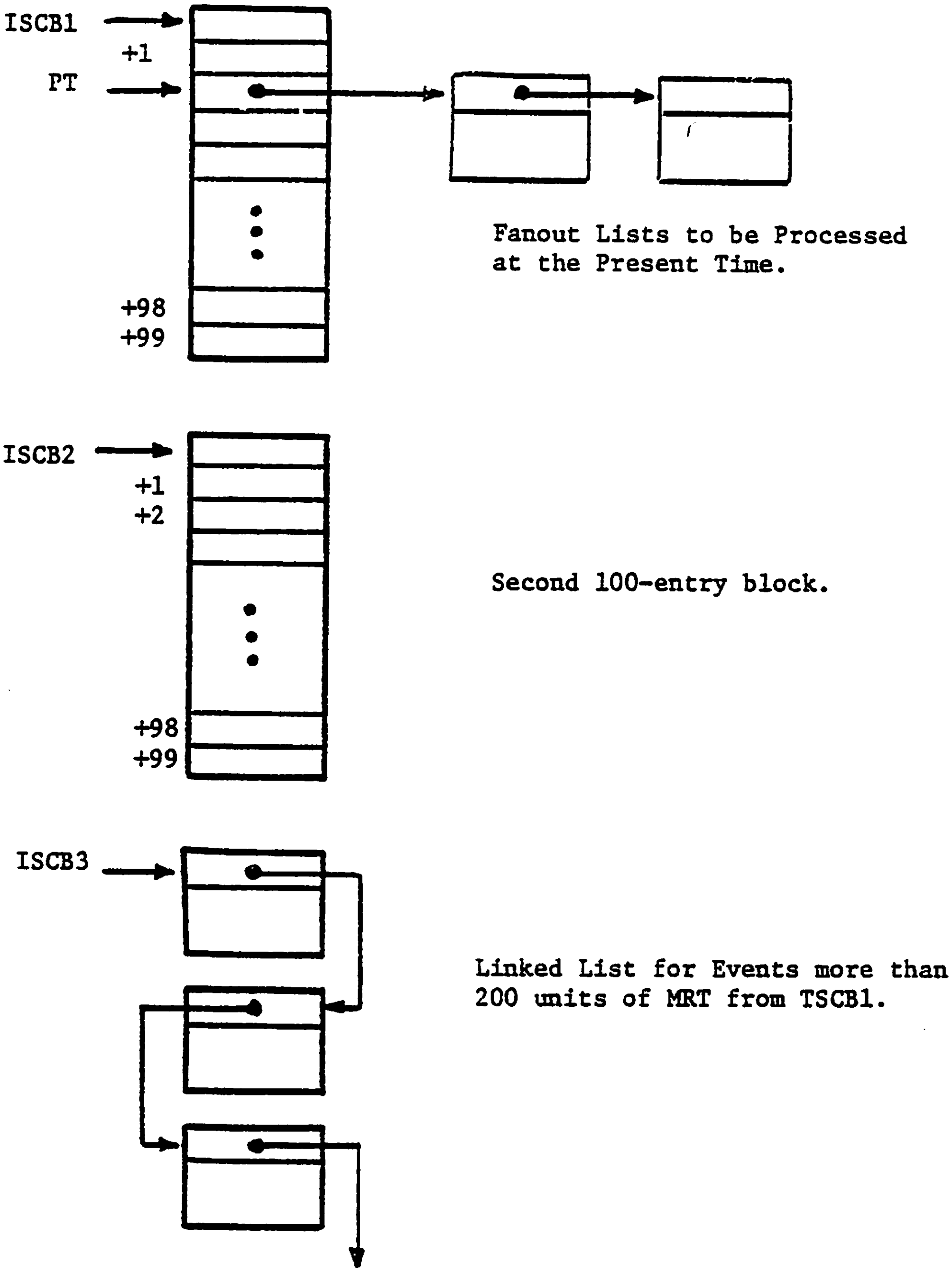
T4 > T3 > T2 > T1 = PT

Fig. 4.6 Simple Linked-List Time Queue

Rather than a simple linked list as described above, a contiguous block of data is set aside where each consecutive position in the block represents the next unit of MRT. If no events are scheduled at a particular time, the entry in the block is null (−1). If the first entry in the block is the present time, PT, then any event to occur s units in the future can be added to the list simply by adding s to the PT pointer and inserting the fanout list pointer in the scheduler block. If the block is 100 units of MRT long, most events will occur within the time-span of the block. A linked list is still used for events outside this range.

Fig. 4.7 shows the structure of the scheduler used in program SPLICE. The program uses two 100-word blocks as well as the linked lists. As the PT pointer moves down one block, entries may be added to the second block. When PT reaches the end of the first block, it jumps directly to the second block and a *swap* occurs. When a swap occurs, the first block is cleared and the linked list is searched to find any events which may occur within the next 200 units of MRT. If any are found they are added to the blocks at the appropriate point. Note that if more than one event (fanout list) is scheduled to occur at the same time, the fanout lists are linked as shown in Fig. 4.7.

4.2.4. Schedular Operation A simple example of the way in which the scheduler operates is illustrated in Fig. 4.8. When processing begins at PT, the address of FOL1 is obtained from the time queue. Each entry on FOL1 is then processed in turn. In this case the first entry in the list is a pointer to element EL1. The scheduler determines how many output nodes the element has and then proceeds to check each one. If the first output node is ON1, the scheduler finds the node list, checks the node type and depending upon the node type passes the fanin list



Fanout Lists to be Processed at the Present Time.

Second 100-entry block.

Linked List for Events more than 200 units of MRT from TSCB1.

Fig. 4.7 Structure of SPLICE Scheduler Tables.

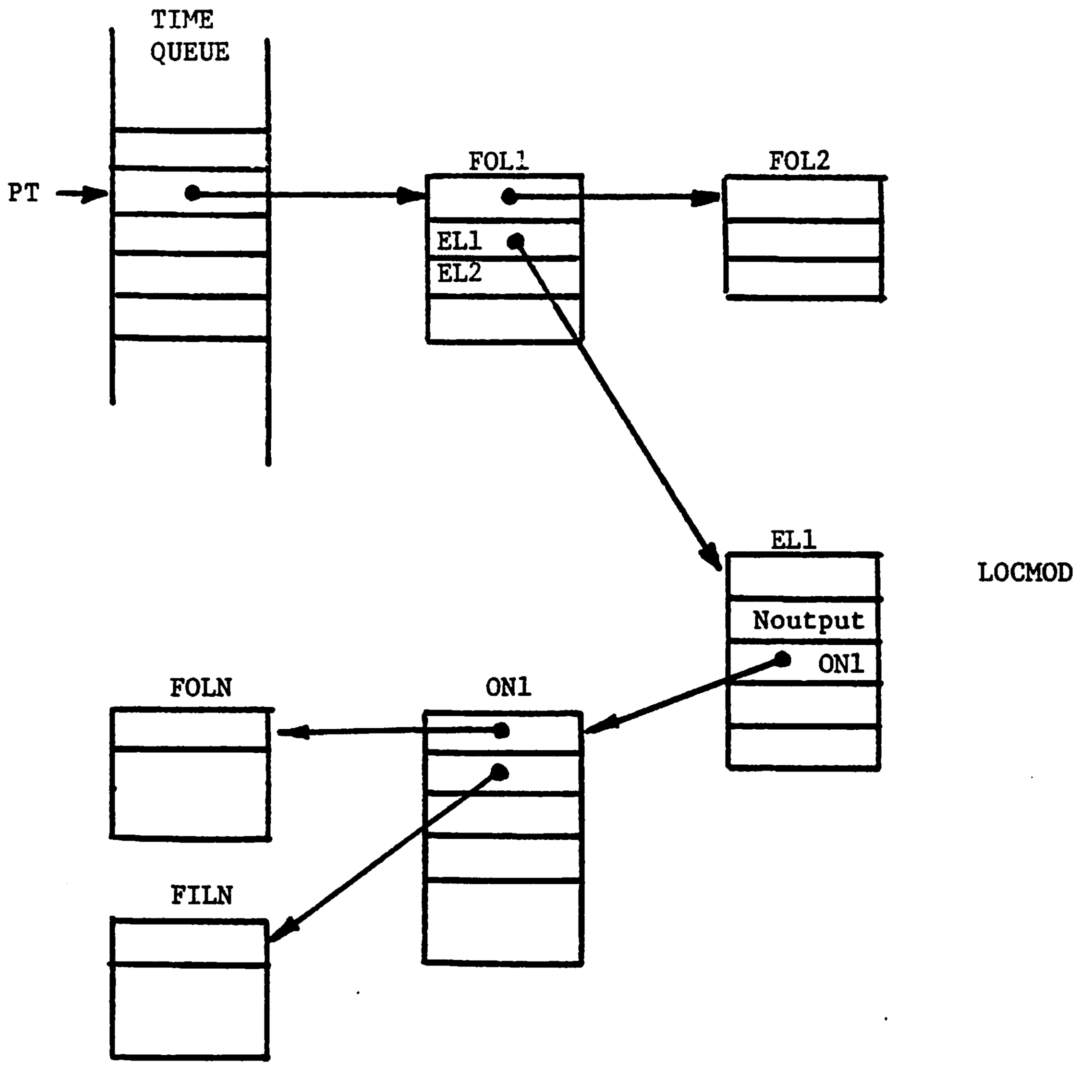


Fig. 4.8 An Example of the Processing of an Event in SPLICE.

address to either the circuit, timing or logic analysis module of the program. The analysis module then performs the analysis at the node in question, updates the signal values and if a significant change has occured sets a flag, notes the time at which the fanouts should be scheduled and returns control to the scheduler. The scheduler checks the flag and, if it is set, adds the pointer to FOLN at the end of the list at the appropriate point in the time queue. The scheduler then moves on to the next output node of EL1 and continues the processing until no output nodes remain. The scheduler begins processing the next element on list FOL1 and so on until all fanouts from the node have been processed. It then moves on to the next list of node fanouts, FOL2. When all events at PT have been processed, the PT pointer is incremented and the process begins again.

4.3. Logic Analysis

With the event scheduler described above, logic analysis is straightforward. When a gate is scheduled, its inputs are evaluated and the logic function of the gate determines the value at the output. If there is a change in a gate output, the appropriate gate delay t_D is obtained from the gate model and the gate fanouts are scheduled t_D units of MRT in the future.

A spike analysis is performed for each gate and should a spike be detected an entry is made in a spike data file. Spikes are not propagated by SPLICE. This data file may then be interrogated by the user after the analysis. A variety of logic gates are defined by SPLICE and described in App. 3.

Most logic nodes have only one fanin (only one gate determines the logic value at the node) and in this case the fanin list contains a single entry. When tri-state gates and logic buses are involved, many gates may fan in to a node. When the logic analysis is performed a node where more than one fanin is present, SPLICE checks the outputs of all other gates connected to the node to see if a *bus contention* exists. If a conflict is present and more than one gate connected to the node is trying to set its value, a diagnostic message is issued to a bus contention file and the new value is forced at the node.

4.4. Timing Analysis

When the scheduler determines that the next node to be processed is a timing node, it calls the timing processor and passes it a pointer to the node fanin list as above. The timing processor then evaluates the nett current charging the grounded node capacitor and computes the change in node voltage for one unit of MRT using a Backward-Euler model for the grounded capacitor. If this change in voltage ΔV is less than a user-defined value ΔV_s, the node fanouts are not scheduled, the node voltages are not updated and control returns to the scheduler. The default value for ΔV_s is 0.1volt. Note that by not updating the node voltages if ΔV is less than ΔV_s, the program avoids the situation where a slowly-varying node may change over a wide range of voltage without ever exceeding ΔV_s between timepoints and hence without ever having its fanouts scheduled. All node voltages in the program are stored as 16bit integers to conserve memory and voltages are scaled to lie within the range of ±32767 units.

SPLICE does not use both a current and an equivalent conductance to model the active devices in the circuit as would be the case in a true Newton-Raphson step. Instead the program predicts the current midway between timepoints and uses a single current source to model each active device. The approach is shown schematically in Fig. 4.9 and results in the use of Eqn. 3-9:

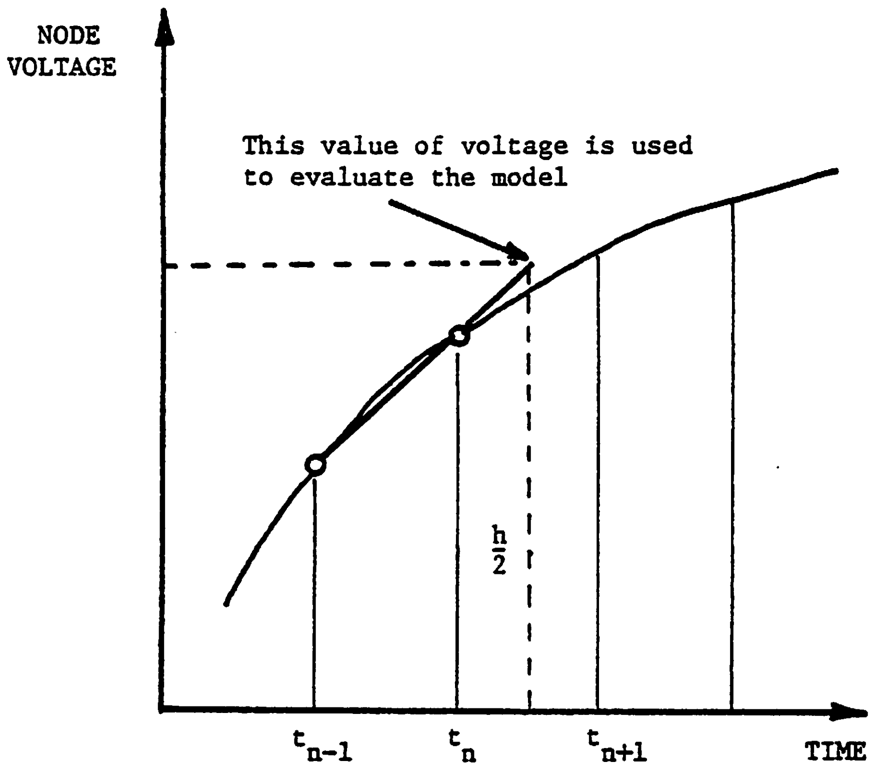


Fig. 4.9 The Extrapolation Scheme Used to Predict Timing Element Current.

$$g^m(v_n)v^m_{n+1} = i^m_{n+1} - \sum_{j=1}^{N} O^{mj} f(v^j_k,\ k = n, n-1, \cdots). \tag{4-1}$$

where now

$$f(v^j_k,\ k = n, n-1, \cdots) = v^j_n + \frac{(v^j_n - v^j_{n-1})}{2} \tag{4-2}$$

This approach reduces the arithmetic required to evaluate the device models and no penalty in either stepsize or accuracy has been observed.

For a node which is switching rapidly from one voltage level to another, the change in the currents flowing through the active devices over a period of one unit of MRT may be larger than can be tolerated for an accurate simulation. For this reason SPLICE monitors the change in I_{ds} for each MOS transistor, ΔI_{ds}, and if ΔI_{ds} exceeds a user-defined value ΔI_c, the timing analysis timestep is reduced bu a factor of 4 and the analysis is repeated. Hence many *internal* timesteps may be used by the timing simulation to obtain a satisfactory solution for the node voltage over the one unit of MRT required by the scheduler. For each internal timestep Eqn. 4-1 is used for the evaluation of the voltage at node m but now the coupling terms are evaluated using

$$f(v^j_k,\ k = n,n-1, \cdots) = v^j_n + \frac{h_i(v^j_n - v^j_{n-1})}{2h_{MRT}} \tag{4-3}$$

where h_i is the internal timestep at the node and h_{MRT} is one unit of MRT.

During the internal solution, if the drain current changes by less than an amount ΔI_D, a provision is made to increase the internal timestep.

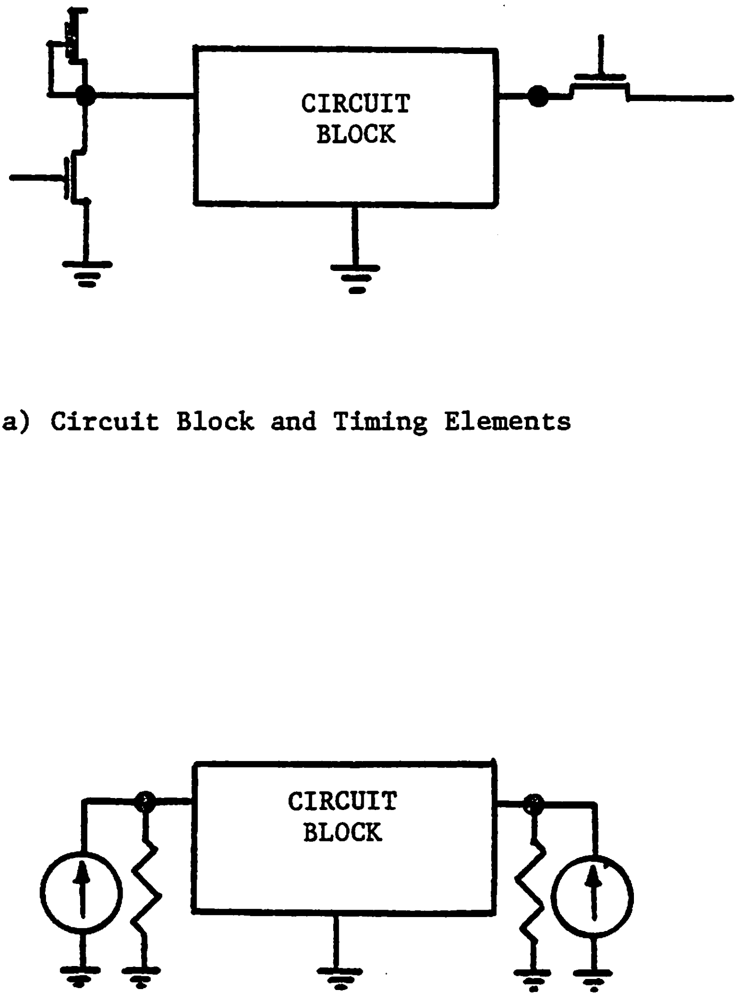
4.5. Circuit Analysis

If the next node to be processed is an external circuit node, the scheduler passes control directly to the circuit analysis module of SPLICE. The data structures used for circuit analysis are more complex than those used for logic and timing analysis and are described in Chap. 5. Once the circuit analysis module has determined which circuit block is to be processed for one unit of MRT it proceeds by evaluating the timing element models for all devices connected to the external circuit nodes, as described in the previous section. This process is illustrated in Fig.4.10(b). The resulting circuit block is then simulated using algorithms similar to those used in program SPICE2. The Trapezoidal method is used for the integration of capacitor currents and a linked-list sparse matrix structure, described in Chap. 5, is used for the solution of the linear algebraic circuit equations at each Newton-Raphson step.

The circuit simulator may use an internal timestep smaller than one unit of MRT and the extrapolation algorithm of Eqn. 4-3 is used to evaluate the timing element contributions at each Newton-Raphson step.

4.6. The Hybrid Interface

The timing analysis and circuit analysis are coupled directly since both simulations use voltages to denote the signal level at a node and hence an additional interface is not required. The logic simulation does require an interface to and from the circuit and timing blocks so that logic levels may be converted to currents and voltages and the voltages of circuit and timing nodes can be converted to logic levels for use in the logic simulation. This interface is included in SPLICE by the use of three types of elements: a *thresholder* (THRESH) for converting voltages to



(a) Circuit Block and Timing Elements

(b) Equivalent Circuit after Evaluation of the Timing Element Contribution.

Fig. 4.10 Circuit Block Processing in SPLICE.

logic levels, a *logic-to-voltage converter* (LTV) and a *logic-to-current converter* (LTC) to convert the logic levels for use in circuit and timing analysis. One of these elements must be included in the simulation whenever a connection is made between a logic gate and circuit or timing transistors.

4.6.1. The Thresholder This element is used to convert the node voltages computed in the logic or timing analysis into one of the four logic levels used in SPLICE. The user can define two threshold levels for each thresholder as shown in Fig. 4.11. For a voltage greater than the logic "1" level, a logic "1" is propagated into the logic network. If the level is below the logic "0" level, a logic "0" is propagated. Any level between these two constitutes a logic unknown state "*".

The high-impedance state is determined by monitoring the equivalent node current and output conductance used in the evaluation of the voltage at the node. If both the current and the conductance values are small an "H" state is propagated.

4.6.2. Logic-to-Circuit Conversion Elements are provided in SPLICE for the conversion of logic levels to voltages and currents. These are the LTV converter and the LTC converter mentioned above. They are identical in form and provide a voltage-source and current-source output respectively for the circuit and timing analysis. The operation of the LTV converter is shown in Fig. 4.12. When a transition occurs from one logic level to another, the converter outputs a ramp. The user can specify the logic "1", logic "0" and logic "*" voltage or current levels as well as rise and fall time. If a logic state becomes high-impedance the converter simply holds the voltage or current at its present level. The converter does not include a provision for voltage decay in the high-impedance state.

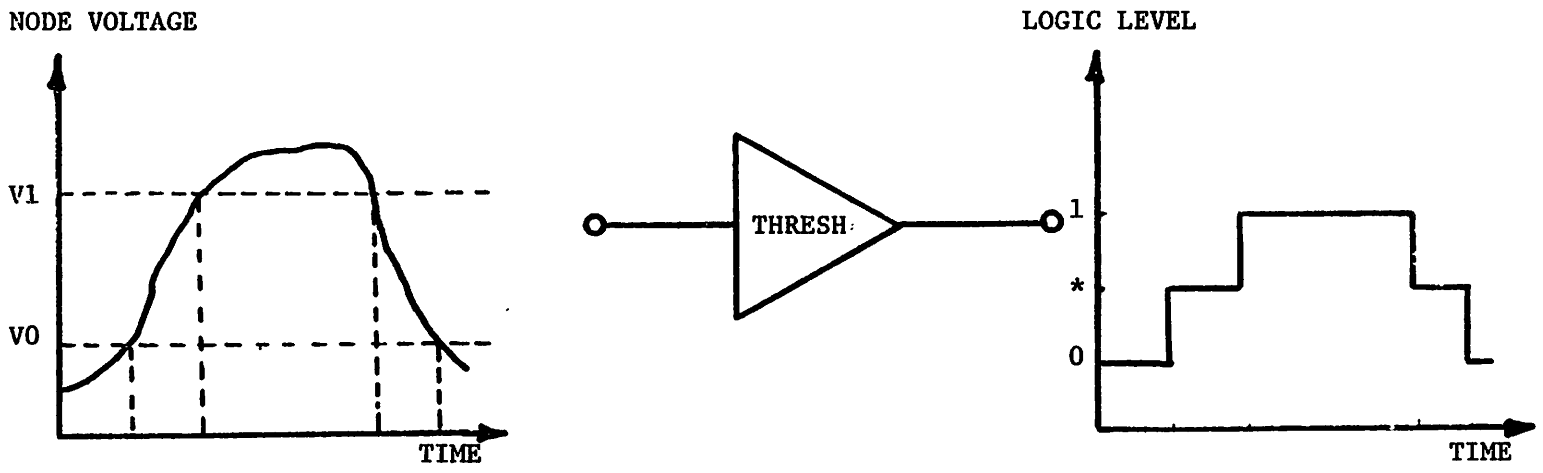


Fig. 4.11 The Operation of the Thresholder in SPLICE.

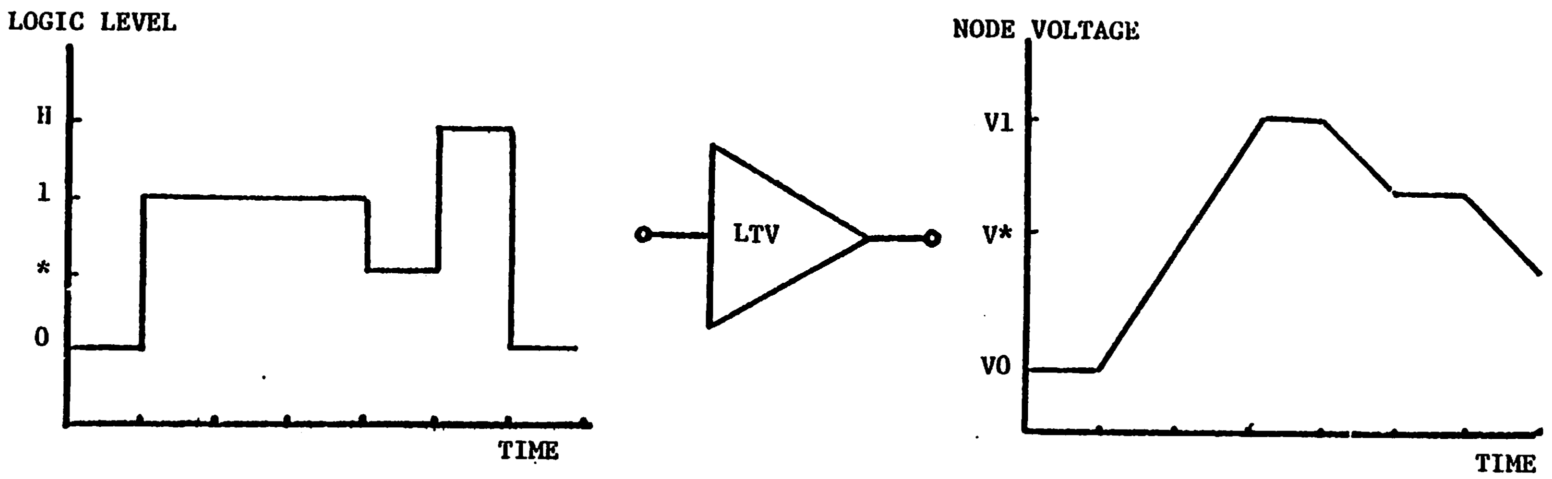


Fig. 4.12 The Operation of the Logic-to-Voltage Converter in SPLICE.

CHAPTER 5

THE SPLICE PROGRAM

5.1. Introduction

The overall structure of SPLICE is shown schematically in Fig. 5.1. Program SPLICE is written for use as a stand-alone batch program or for use with an intelligent terminal for input and output processing. To aid implementation on an intelligent terminal the program is written as three separate modules which communicate via data files.

The input module reads the user's input data and checks for obvious syntax and circuit errors, such as missing device models or a node with only one element connected to it. The input processor produces a binary data file which is then read by the setup and analysis module of SPLICE. The setup and analysis module of the program actually performs the analysis. During the analysis, an output data file is generated for post-processing by the output module. The file contains the voltages or logic levels at all nodes to be plotted by the user for each time the value at the node changed and its fanouts were scheduled to be processed.

The output processor interprets this file and plots the logic or voltage waveforms on an x-y plotter. Examples of the output from SPLICE are included in Chap. 6.

SPLICE is written in FORTRAN and is approximately 8000 statements long. A listing of SPLICE is included in App. 11. This chapter describes the operation of

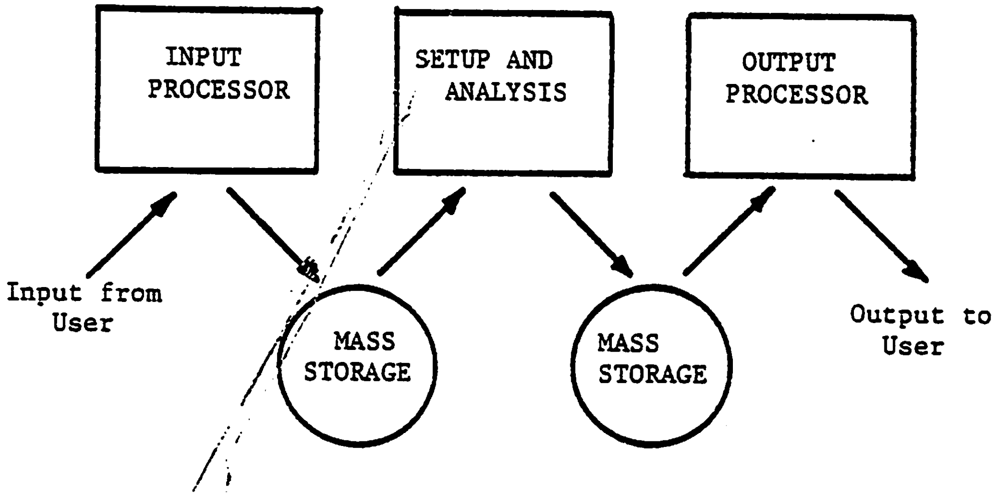


Fig. 5.1 General Structure of SPLICE.

SPLICE and the data structures it uses. The input processor is described with the types of circuit elements available to the user. A description of the setup and analysis phase follows with a simple circuit example, and finally the output post-processor is described briefly.

5.2. The Input Processor

The input processor reads model, element and analysis control statements which are entered by the user. It links elements to the corresponding model and compacts the input data into a file to be read by the setup and analysis module of SPLICE. The program contains a variety of built-in models for logic gates, transistors and other elements. A list of the elements available in SPLICE is included in App. 3. The format of the input data provided by the user is very similar to that of SPICE2. Some examples of inputs for SPLICE are included in App. 7. After decoding the input data, the input processor generates a binary file whose format is described in App. 4.

5.3. Setup and Analysis Module

5.3.1. Introduction A block diagram of the setup and analysis module of SPLICE is shown in Fig. 5.2. The data file generated by the input processor is read by the setup module and the data structures required for analysis are generated. All calculations that can be performed prior to the analysis are also done at this time. After setup is complete, the analysis module is entered and the event scheduler takes over for the duration of the analysis.

SPLICE uses a simple dynamic memory allocation scheme. The memory manager used in SPLICE can allocate fixed blocks of real or integer data and the

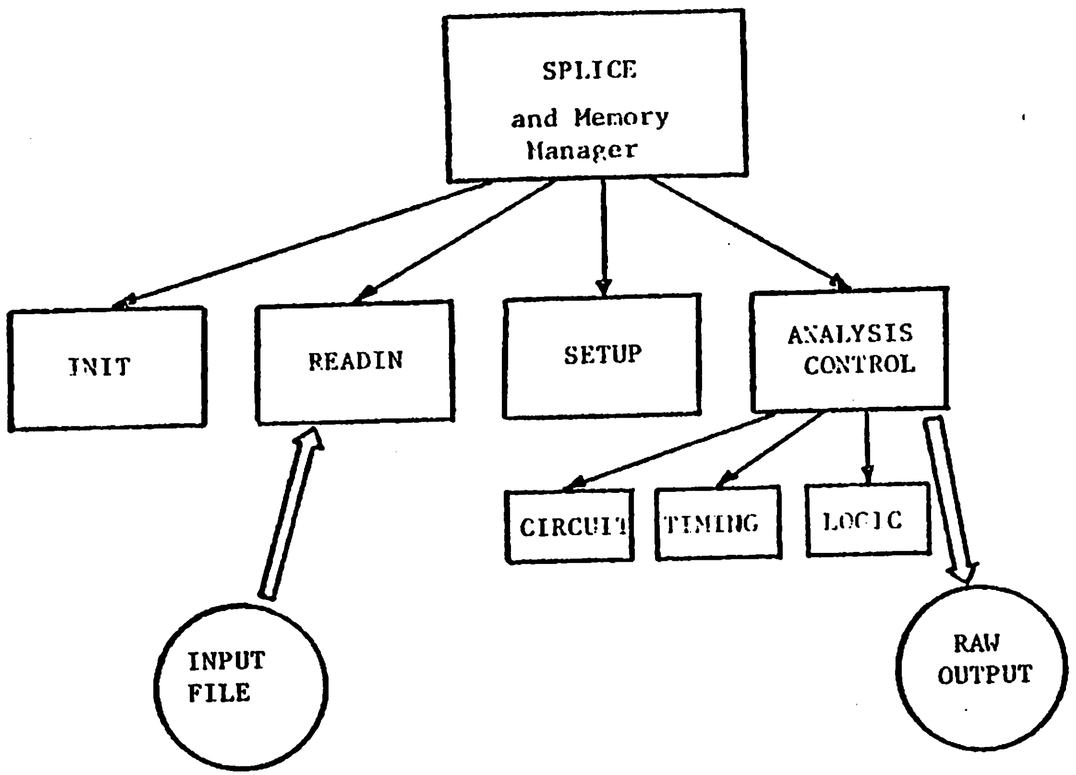


Fig. 5.2 Block Diagram of the Setup and Analysis Block of SPLICE

transportability of the program was considered in its design. The program uses scratch files during the setup phase to reduce the maximum amount of memory required by the program and to avoid the necessity to relocate data blocks which can be an difficult process in FORTRAN. The scratch files are referenced via subroutine calls to aid conversion of the program to computers with a different file structure.

5.3.2. The Setup Phase The operation of the setup phase is illustrated with the simple example shown in Fig. 5.3. SPLICE reads the element models and allocates a data array for the model type and its parameters as shown in Fig. 5.4. In this case, three models are used: one model for the input source, one for the inverter and the third model for both NAND gates. At the same time the program generates a model map to aid the linking of elements and models which is to follow. Next the circuit elements are read and sorted according to type (logic, timing or circuit) and the size of the fanin and fanout tables are computed and stored in a node map. The program proceeds to generate the data structures for all the nodes including the allocation of storage for the fanin and fanout tables. The tables for the four nodes of Fig. 5.3 are shown schematically in Fig. 5.5. The circuit elements are read back into memory in the compact form described in the previous chapter, and the scheduler table storage is allocated. The data storage for the four elements of the example circuit is shown schematically in Fig. 5.6. By allocating the scheduler storage last, should the linked-list of future events grow during the analysis, its size is not limited by any other blocks allocated later in the setup phase.

Once all the data blocks have been allocated, SPLICE proceeds to generate the fanin and fanout tables and link each element to its model. At this point, the

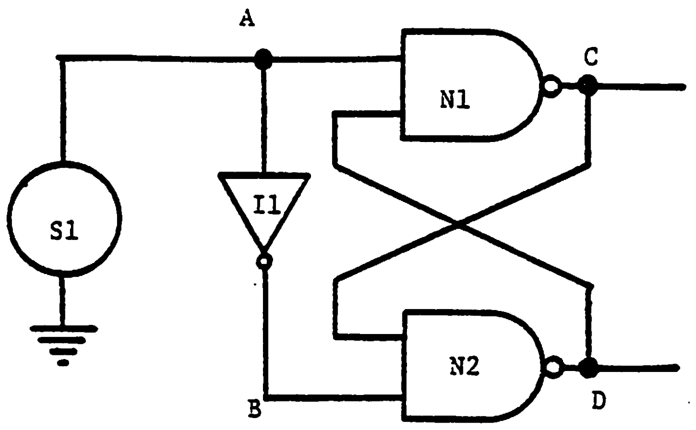
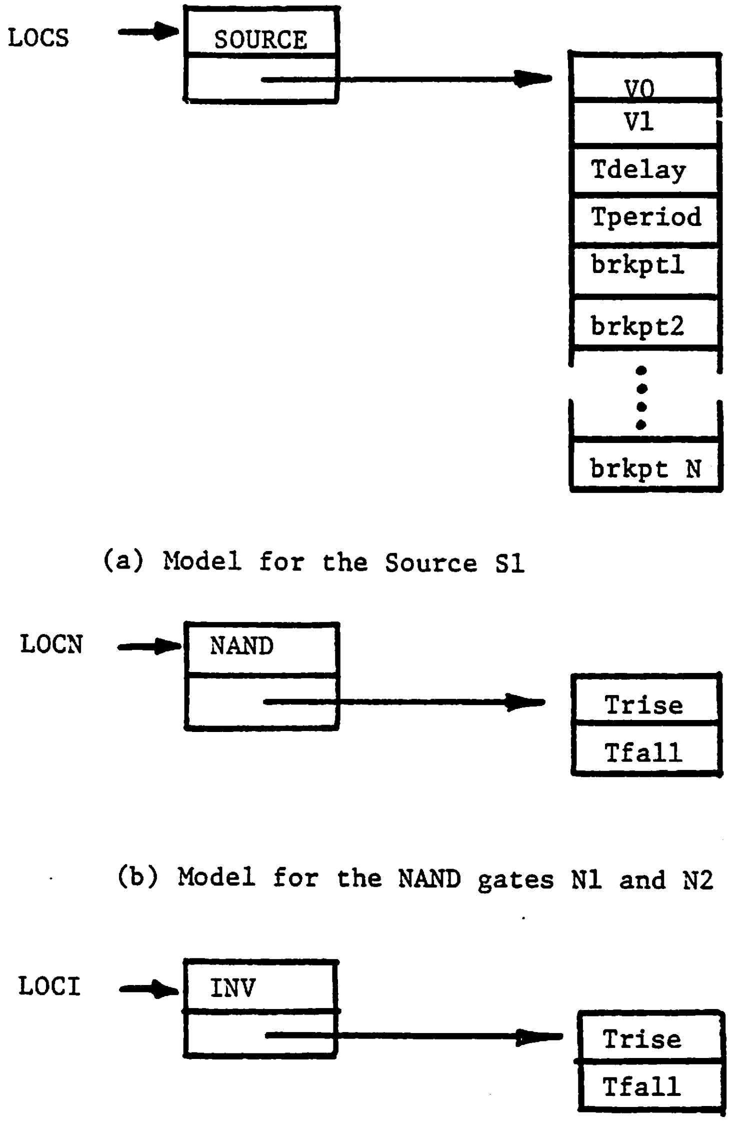


Fig. 5.3 Example Circuit to illustrate the
Data Structures of the Setup Phase.



(a) Model for the Source S1

(b) Model for the NAND gates N1 and N2

(c) Model for Inverter I1

Fig. 5.4 Model Storage for the Example of Fig. 5.3.

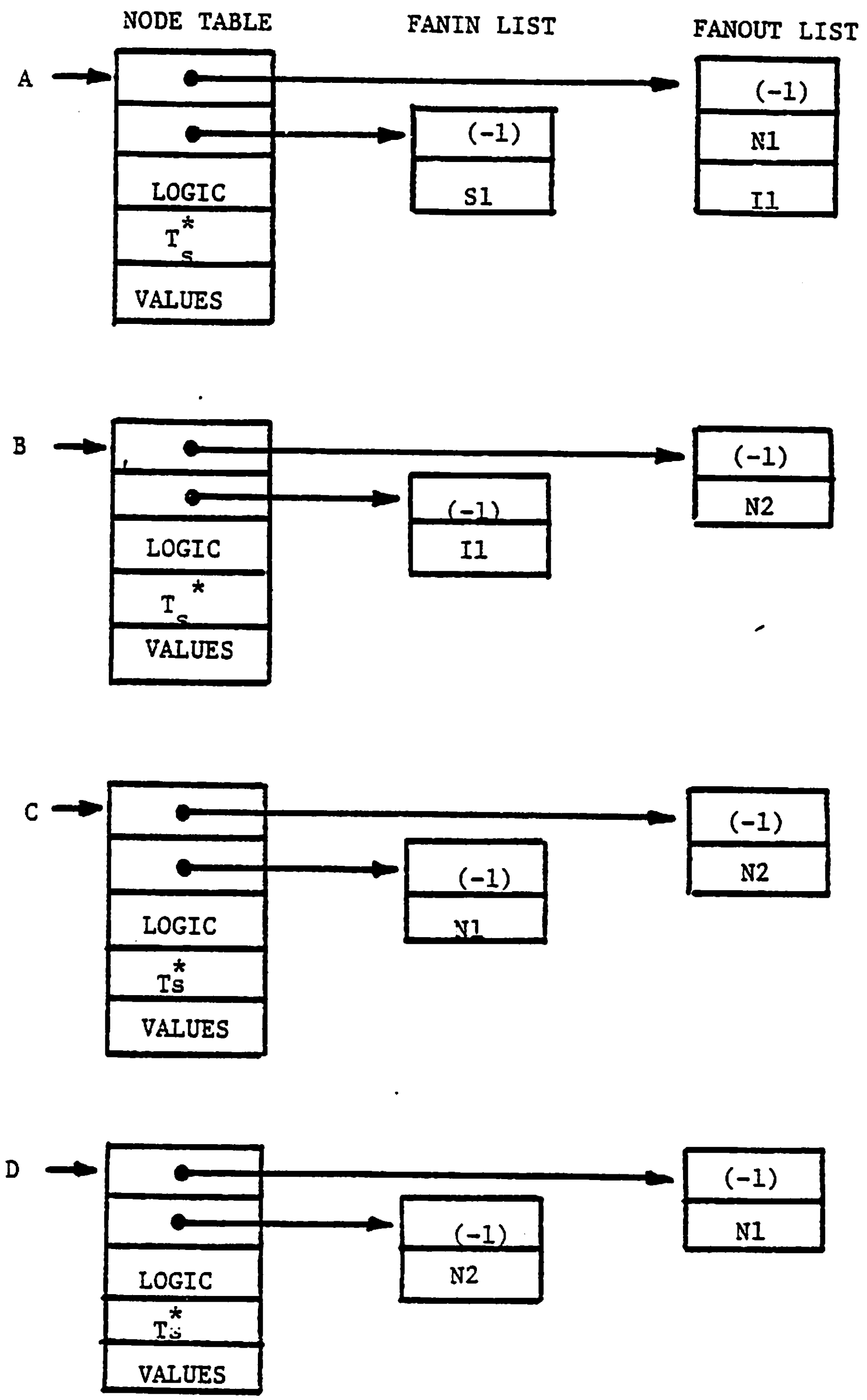


Fig. 5.5 SPLICE tebles for the nodes of Fig. 5.3.

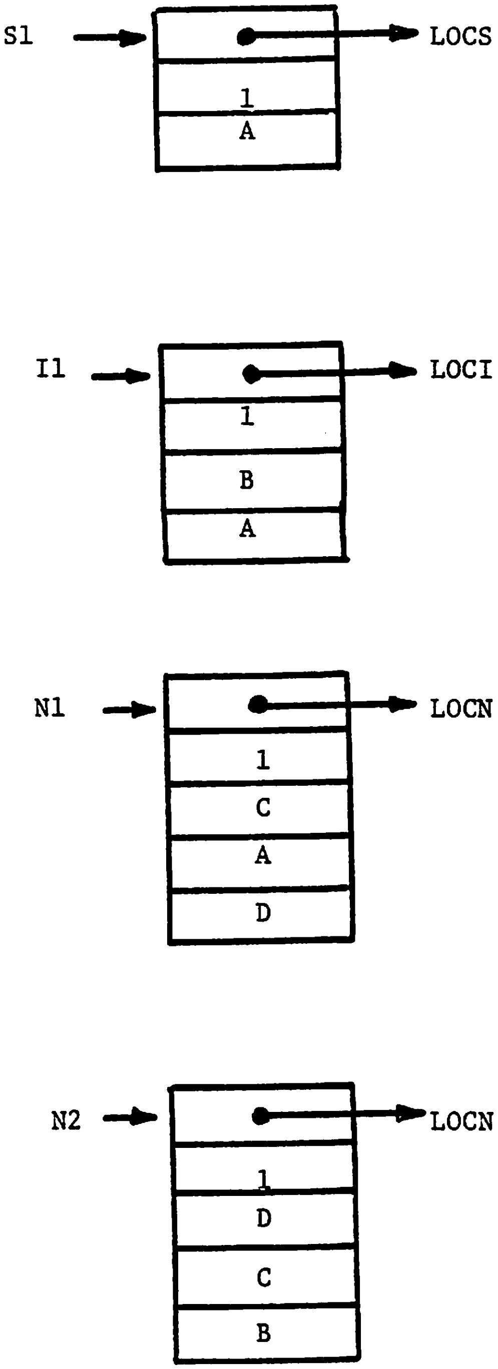


Fig. 5.6 SPLICE tables for the Elements of Fig. 5.3.

elements which comprise circuit blocks must be identified. All elements which require circuit analysis are linked. SPLICE then generates a *Model Control Block* (MCB) for the first circuit block and repeatedly searches the circuit element list to coalesce all connected circuit elements. These elements are linked to the MCB as illustrated in Fig. 5.7. Once all elements of the first circuit block have been identified, a second MCB is generated and the process is repeated until the circuit element list is empty. Once all the MCBs have been generated and the internal and external circuit nodes associated with each block have been added to the MCB, SPLICE begins to allocate storage for the circuit matrix. The matrix is stored with the circuit nodes as shown in Fig. 5.8. A linked list is used to identify the upper and lower triangular entries of the matrix. The internal circuit nodes are then reordered using a Markowitz scheme [34] to minimize matrix fillins generated during the LU factorization process. the external circuit nodes are not re-ordered as they must occur first in the MCB data array of Fig. 5.4 to allow efficient evaluation of the timing element contributions in the analysis. A mock LU factorization is not performed but the matrix fillins are generated during the first iteration of the circuit analysis.

Once the data structures have been set up, SPLICE reads the first set of analysis requests (until a "GO" statement is encountered) and computes the analysis control data, such as MRT and the voltage scaling factor (for a range of ±32767 units). A number of pre-analysis data reductions are performed, such as the conversion of gate delays to units of MRT, and all the elements are scheduled at time t=0.

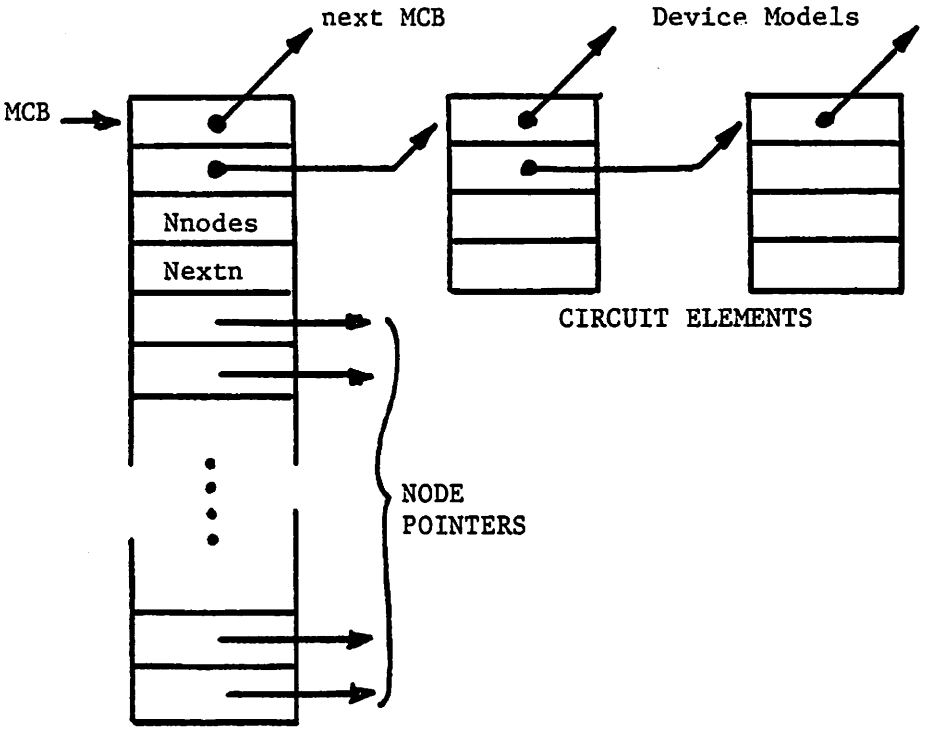


Fig. 5.7 Data Structure of a Circuit Model Control Block and Associated Elements

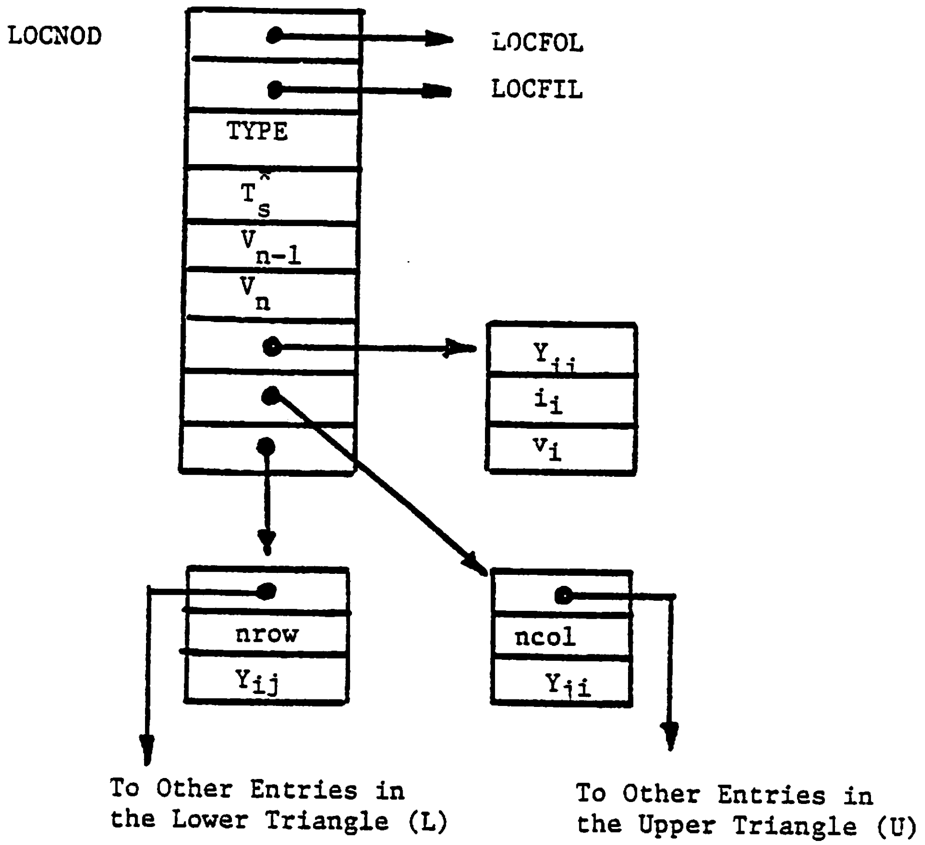


Fig. 5.8 Storage of the Circuit Matrix with the Circuit Node.

5.3.3. The Analysis Phase The analysis phase consists of executing the algorithms describcd in the previous chapter. When an output event is scheduled SPLICE enters the time and signal level at the node in the output file for post processing. After the requested simulation time has elapsed, any remaining analysis control requests are executed and the program terminates. A list of analysis control statements available in SPLICE is included in App. 3

5.4. The Output Processor

The output processor has been implemented on an off-line HP21-MX minicomputer acting as an intelligent terminal. It can produce plots of selected node waveforms on an X-Y plotter. The format of the output file read by this processor is included in App. 6

CHAPTER 6

PROGRAM PERFORMANCE

6.1. Introduction

This chapter presents the results of a number of circuit simulations using SPLICE. The results show that using a hybrid analysis for a large circuit, between one and three orders of magnitude speed improvement and between one and two orders of magnitude reduction in memory requirements can be obtained compared to conventional circuit analysis.

The first example illustrates the use of the hybrid approach to improve analysis speed and the input to program SPLICE is described. The second example, a 256-by-1 bit dynamic RAM circuit, demonstrates the use of concurrent circuit, timing, and logic analysis. The final example, a 700 MOS transistor digital filter circuit, illustrates the speed and stability improvements possible using an event-driven analysis. The description of each of these circuits, as read by SPLICE, is included in App. 7.

6.2. The Binary-to-Hexidecimal Decoder

6.2.1. Timing Analysis The circuit schematic of the Binary-to-Hexidecimal Decoder is shown in Fig. 6.1. It consists of an array of NOR gates and sixteen output inverters to provide an active-low output. The waveforms obtained from program SPLICE using a timing analysis of the circuit are shown in Fig. 6.2. These waveforms agreed with those obtained from a SPICE2 simulation to within 2% using

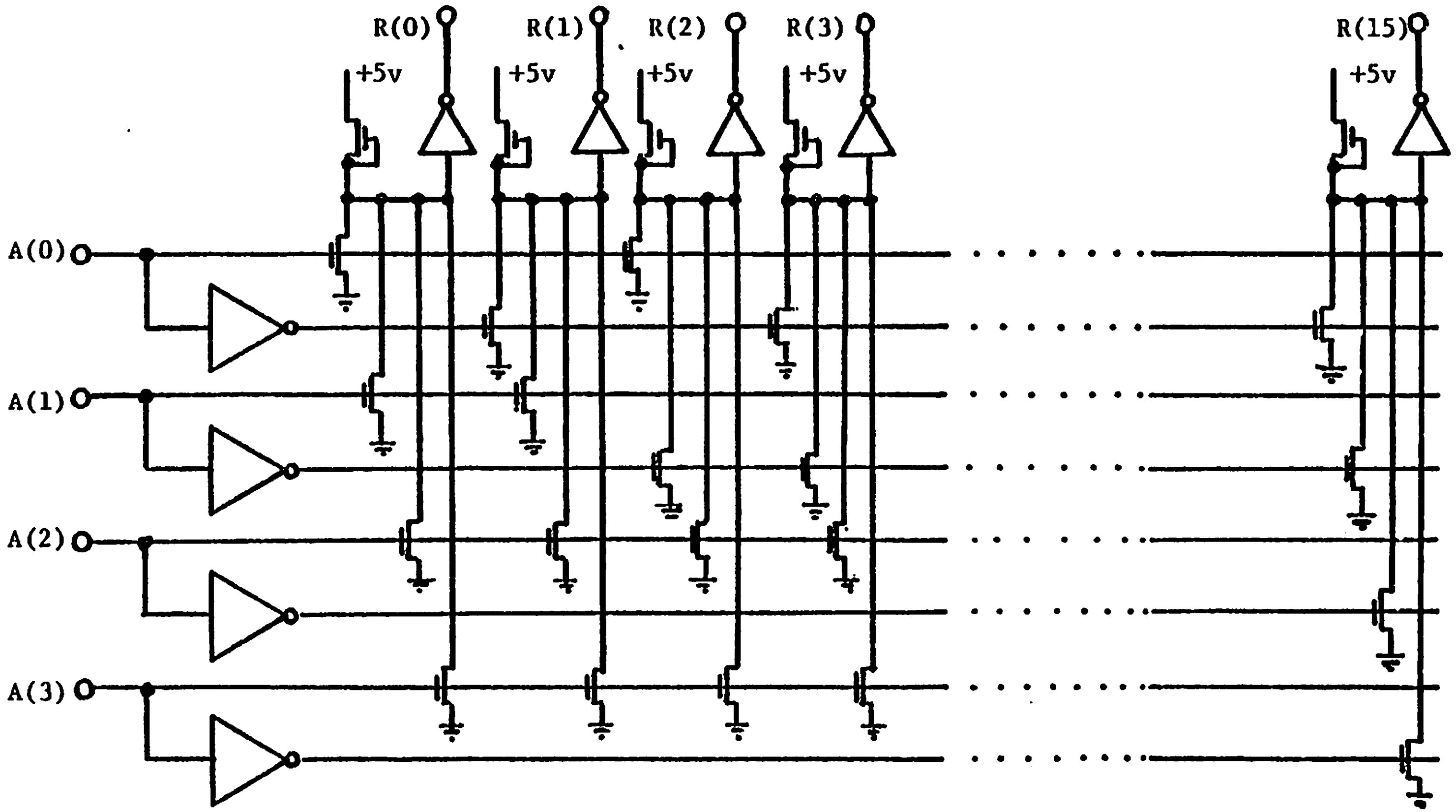


Fig. 6.1 Schematic Diagram of the Binary-to-Hexidecimal Decoder Circuit. Note that all Elements of the Analysis are Timing Elements and the Inverters shown are in fact a load and driver transistor.each.

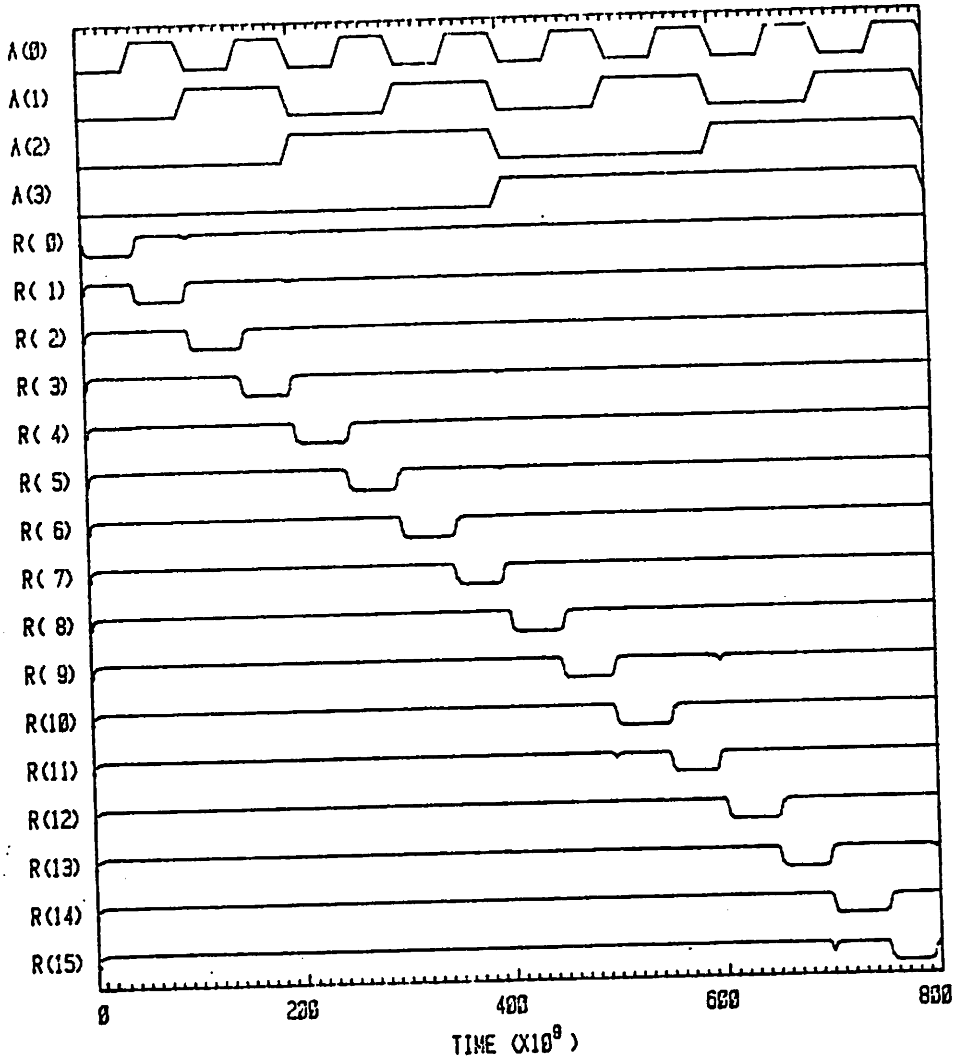


Fig. 6.2 Output waveforms for the Timing Analysis of the Hexidecimal Decoder Circuit.

equivalent device models in both programs. A comparison of analysis times between the SPICE? analysis and the SPLICE timing analysis shows that SPLICF is approximately 80 times faster than SPICE2 and requires 5% of the data storage. Note the spikes in the output waveforms caused by the delay of the input inverters.

Since SPLICE is an event-driven simulator, it only analyzes nodes which are changing at any time during the simulation. This is illustrated in Fig. 6.3 where the "events" used to generate the plot of Fig. 6.2 are shown. Each dot in Fig. 6.3 represents an analysis at that node. Note that while the nodes are not active, they are not processed.

6.2.2. Hybrid Analysis Once the circuit designer is satisfied that the circuit meets the design specifications, a macromodel of the circuit may be constructed using a combination of timing transistors and logic gates. A macromodel for the decoder above is shown in Fig. 6.4. All the internal transistors have been replaced with logic gates and the output inverters are still analyzed using a timing analysis. The input voltages are converted to logic levels using thresholding circuits, and the logic outputs are converted back to voltages to drive the output inverters.

The input file used to describe the circuit is shown in Fig. 6.5(a) and Fig. 6.5(b). The model for each input source (type LSRC) is specified first. The model parameters for the sources are the input levels, followed by a delay time, period, and a set of breakpoints on the piece-wise linear input waveform. The list of inputs is terminated with a −1. The two MOS transistor models follow. They are used for the transistors in the output inverters. The parameters for these models are described in App. 3. The first model is the driver device (NDRIV) and the

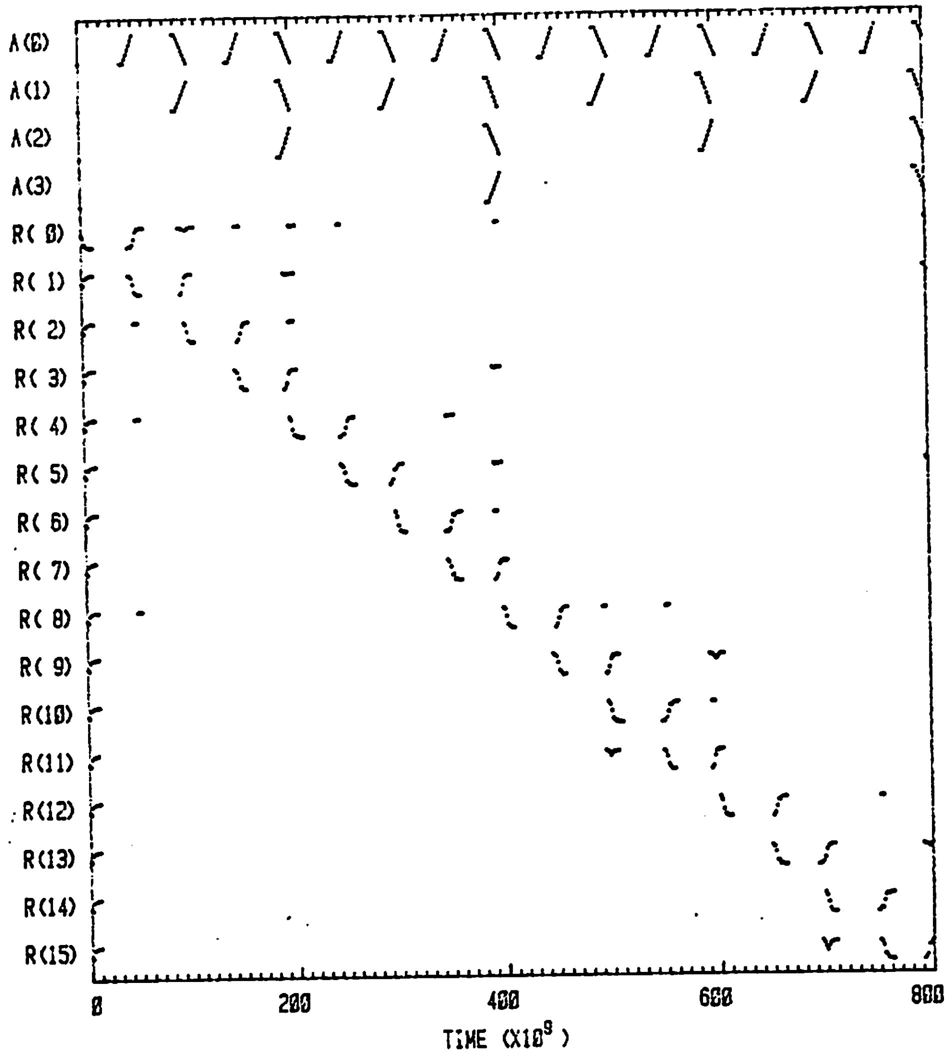


Fig. 6.3 Events processed by SPLICE for the analysis of the Binary-to-Hexidecimal Decoder Circuit.

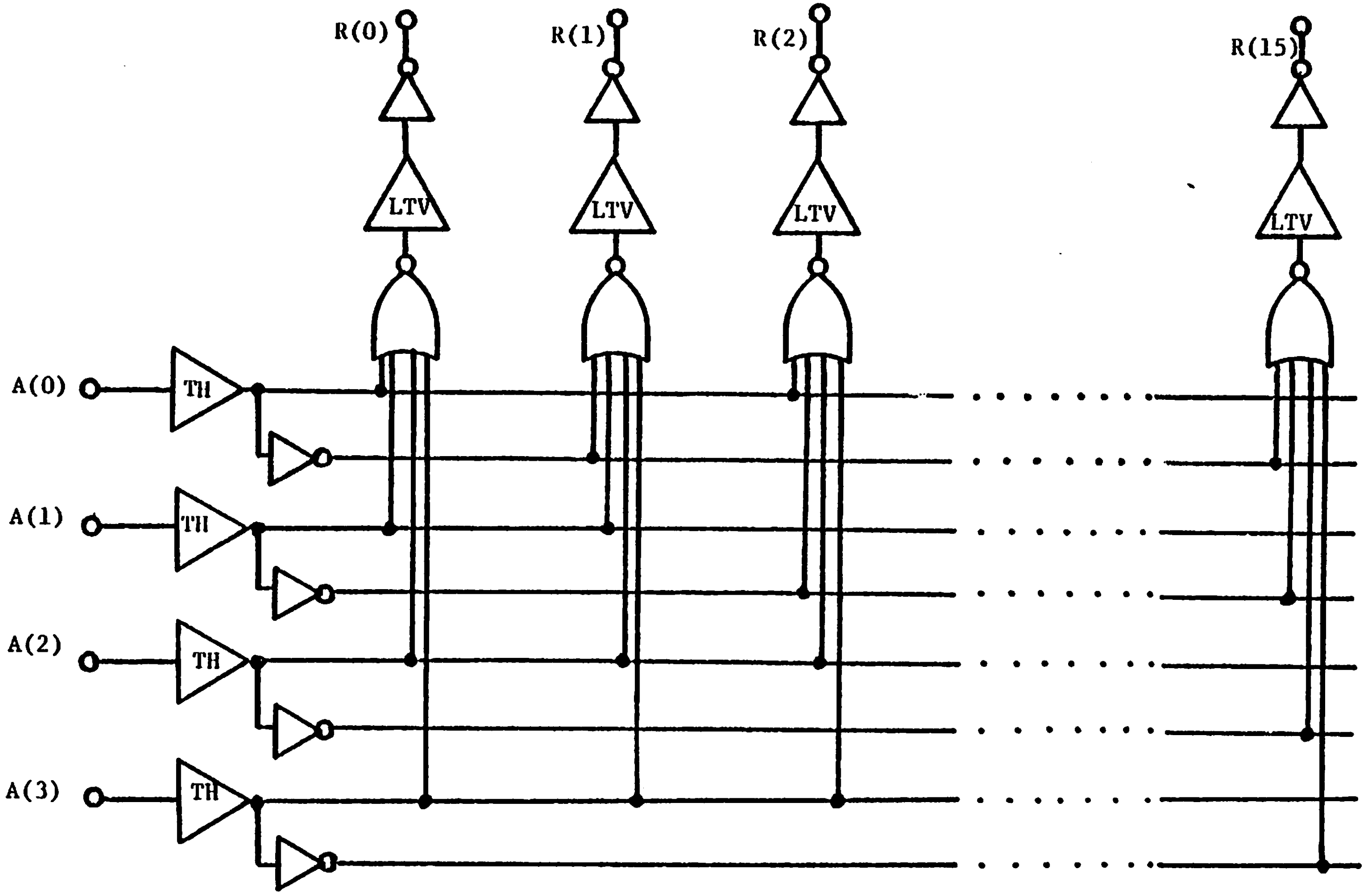
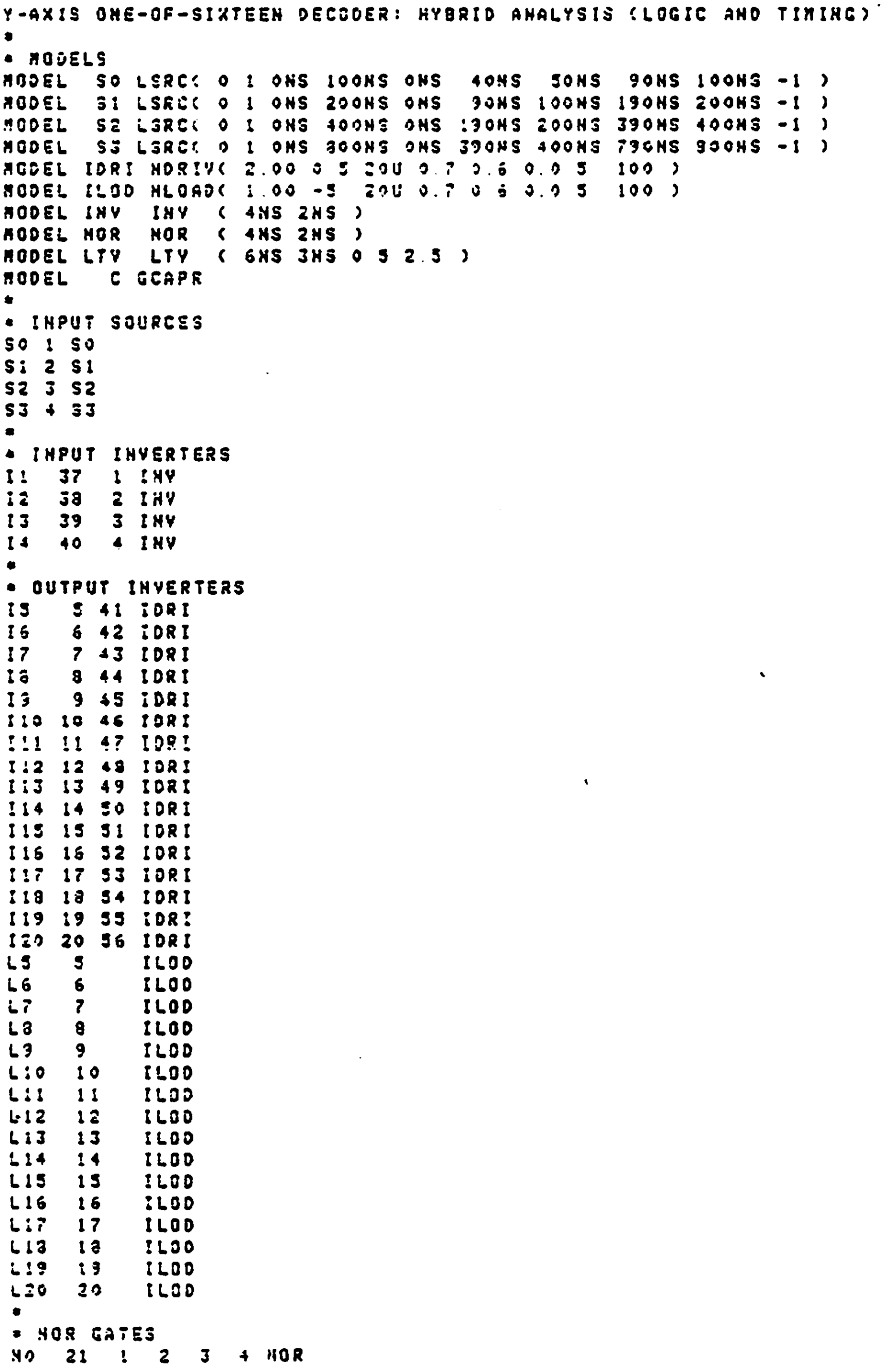


Fig. 6.4 Macromodel of the Binary-to-Hexidecimal Decoder Circuit. Note only the Output Inverter gates are timing elements.

second model is for the depletion load transistor (NLOAD). The supply voltage for the load is set to 5 volts in the model. Models for the logic inverter and NOR gates are included, each with a rise time of 4ns and a fall time of 2ns, derived from the timing analysis results of the previous example. The logic-to-voltage converter model, LTV, defines the rise time ("0"-to-"1" transition time), fall time ("1"-to-"0" transition time) and voltages corresponding to logic "0", logic "1" and logic unknown "*" respectively. The final model defines C as a grounded capacitor.

The element list follows the ten model specifications described above. The input sources, input logic inverter gates and the sixteen output timing inverters, each of which consists of two transistors, a driver and a load, then follow. The driver transistors have two nodes; a gate node and a drain node. The load simply requires that its gate/source node be specified and the substrate node is assumed to be node 0. The NOR gates follow in Fig. 6.5(b). These are all four-input gates and the output node number is the first one specified. The logic-to-voltage converters are specified and the capacitance at each output node is set to 0.08pF, derived from the circuit layout.

The final data required by SPLICE is the list of analysis requests. The OPTS statement sets the time for each output event (plot point) to correspond to each unit of MRT when the node is active. This request also sets the maximum circuit voltage to 7 volts. The TOPTS request sets the timing analysis options. These include the internal step-size control parameters, the ratio of minimum-permitted-stepsize to one unit of MRT and finally the voltage change required at a node before its fanouts are scheduled (0.1volt). The simulation time is set for 800ns with each unit of MRT set to 2ns.



```
Y-AXIS ONE-OF-SIXTEEN DECODER: HYBRID ANALYSIS (LOGIC AND TIMING)
*
* MODELS
MODEL  S0 LSRC( 0 1 0NS 100NS 0NS  40NS  50NS  90NS 100NS -1 )
MODEL  S1 LSRC( 0 1 0NS 200NS 0NS  90NS 100NS 190NS 200NS -1 )
MODEL  S2 LSRC( 0 1 0NS 400NS 0NS 190NS 200NS 390NS 400NS -1 )
MODEL  S3 LSRC( 0 1 0NS 800NS 0NS 390NS 400NS 790NS 800NS -1 )
MODEL IDRI NDRIV( 2.00 0 5 20U 0.7 0.6 0.0 5  100 )
MODEL ILOD NLOAD( 1.00 -5  20U 0.7 0.6 0.0 5  100 )
MODEL INV  INV  ( 4NS 2NS )
MODEL NOR  NOR  ( 4NS 2NS )
MODEL LTV  LTV  ( 6NS 3NS 0 5 2.5 )
MODEL   C GCAPR
*
* INPUT SOURCES
S0 1 S0
S1 2 S1
S2 3 S2
S3 4 S3
*
* INPUT INVERTERS
I1  37  1 INV
I2  38  2 INV
I3  39  3 INV
I4  40  4 INV
*
* OUTPUT INVERTERS
I5   5 41 IDRI
I6   6 42 IDRI
I7   7 43 IDRI
I8   8 44 IDRI
I9   9 45 IDRI
I10 10 46 IDRI
I11 11 47 IDRI
I12 12 48 IDRI
I13 13 49 IDRI
I14 14 50 IDRI
I15 15 51 IDRI
I16 16 52 IDRI
I17 17 53 IDRI
I18 18 54 IDRI
I19 19 55 IDRI
I20 20 56 IDRI
L5   5    ILOD
L6   6    ILOD
L7   7    ILOD
L8   8    ILOD
L9   9    ILOD
L10  10   ILOD
L11  11   ILOD
L12  12   ILOD
L13  13   ILOD
L14  14   ILOD
L15  15   ILOD
L16  16   ILOD
L17  17   ILOD
L18  18   ILOD
L19  19   ILOD
L20  20   ILOD
*
* NOR GATES
N0  21  1  2  3  4 NOR
```

Fig. 6.5 (a) Input Data for the Analysis of the Decoder Circuit.

```
N1  22 37  2  3  4 NOR
N2  23  1 38  3  4 NOR
N3  24 37 38  3  4 NOR
N4  25  1  2 39  4 NOR
N5  26 37  2 39  4 NOR
N6  27  1 38 39  4 NOR
N7  28 37 38 39  4 NOR
N8  29  1  2  3 40 NOR
N9  30 37  2  3 40 NOR
N10 31  1 38  3 40 NOR
N11 32 37 38  3 40 NOR
N12 33  1  2 39 40 NOR
N13 34 37  2 39 40 NOR
N14 35  1 38 39 40 NOR
N15 36 37 38 39 40 NOR
*
* LOGIC-TO-VOLTAGE CONVERTERS
V1  41 21 LTV
V2  42 22 LTV
V3  43 23 LTV
V4  44 24 LTV
V5  45 25 LTV
V6  46 26 LTV
V7  47 27 LTV
V8  48 28 LTV
V9  49 29 LTV
V10 50 30 LTV
V11 51 31 LTV
V12 52 32 LTV
V13 53 33 LTV
V14 54 34 LTV
V15 55 35 LTV
V16 56 36 LTV
*
* NODE CAPACITANCES
C5   5 C 0.08P
C6   6 C 0.08P
C7   7 C 0.08P
C8   8 C 0.08P
C9   9 C 0.08P
C10 10 C 0.08P
C11 11 C 0.08P
C12 12 C 0.08P
C13 13 C 0.08P
C14 14 C 0.08P
C15 15 C 0.08P
C16 16 C 0.08P
C17 17 C 0.08P
C18 18 C 0.08P
C19 19 C 0.08P
C20 20 C 0.08P
*
* ANALYSIS REQUESTS
OPTS 1  7.0
TOPTS 0.1   0.05   1000 0 1
TIME 2.0NS 800NS
PLOT 1, 2, 3, 4
PLOT 5, 6, 7, 8, 9, 10, 11, 12
PLOT 13, 14, 15, 16, 17, 18, 19, 20
*
GO
*
```

Fig. 6.5 (b) Input Data for the Hybrid Analysis of the Decoder Circuit.

The output waveforms for this analysis are shown in Fig. 6.6. Note that they are almost identical to those of Fig. 6.2 except that some of the spikes are missing. This analysis is over an order of magnitude faster than in the previous case. If the circuit simplification process illustrated here is performed carefully, substantial analysis speed improvements can be obtained for a small penalty in accuracy.

6.3. The 256-by-1 bit Dynamic RAM

A block diagram of the RAM circuit is shown in Fig. 6.7. The row and column decoders used in the analysis of this circuit are based on the decoder described above and included both logic gates and timing elements. The input/output circuits and storage transistors of the RAM are analyzed using a timing analysis and each sense amplifier is analyzed as a separate circuit block. The sense amplifiers used in the simulation are a modification of the Intel design [35] and the schematic for the sense amplifier and associated "dummy" cell is shown in Fig. 6.8. The input data used for this analysis is included in App. 7.

Fig. 6.9 contains a summary of the statistics provided by SPLICE for the RAM analysis. It also includes estimates for the memory and time requirements of SPICE2 for the same analysis. Fig. 6.10 shows the waveforms produced in a "write-1-read-1" mode and Fig. 6.11 shows the waveforms for a "write-0-read-0". The voltage at the storage node is indicated as A_{bit}.

The simulation time for this example was estimated to be approximately twenty times faster than for SPICE2. This estimate is based on the MOS model evaluation time and other overhead required in the circuit analysis (App. 9). The simulation could not be performed with SPICE2 as the memory requirement for the SPICE2 analysis exceed 200000 words on a CDC 6400 computer and these resources

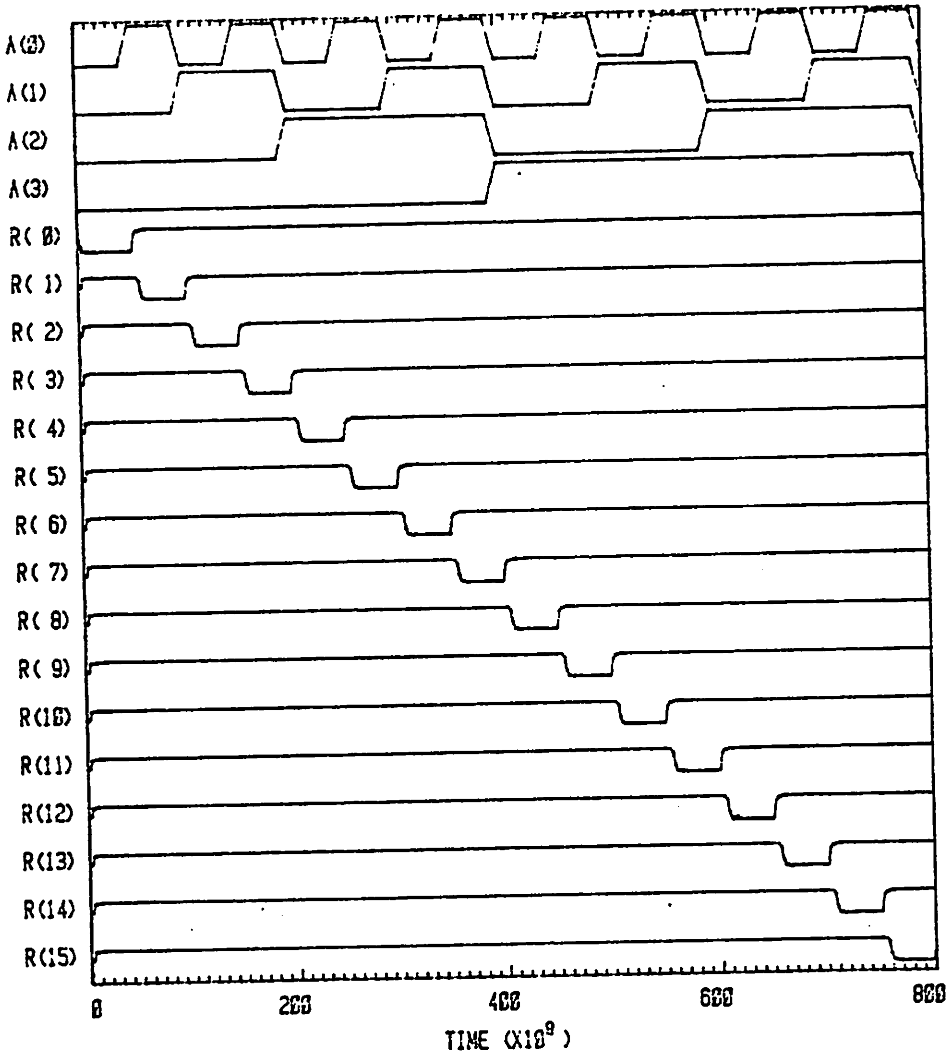


Fig. 6.6 Output Waveforms for the analysis of the hybrid macromodel of the Decoder circuit

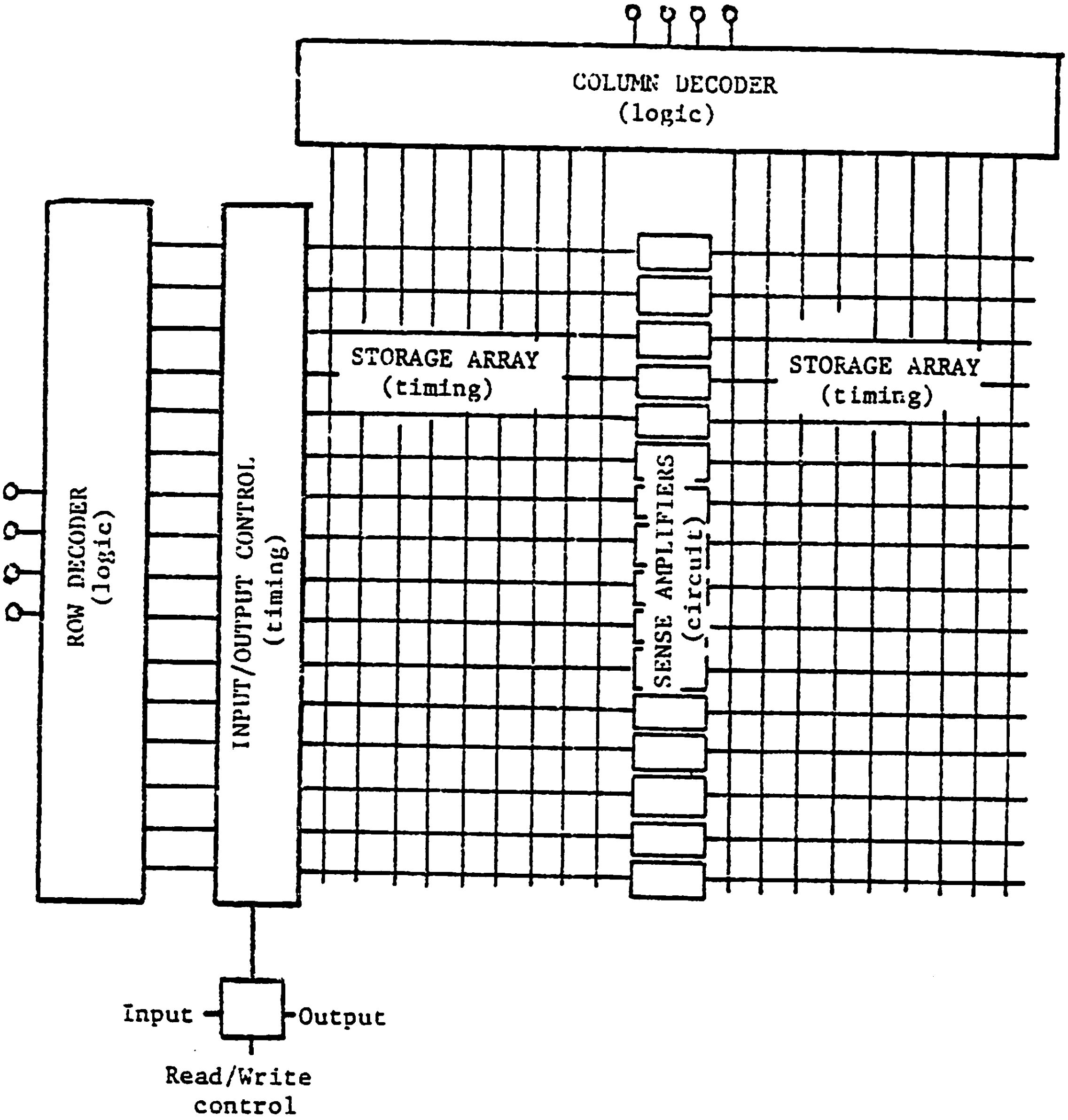


Fig. 6.7 Block Diagram of 256-by-1 bit Dynamic RAM

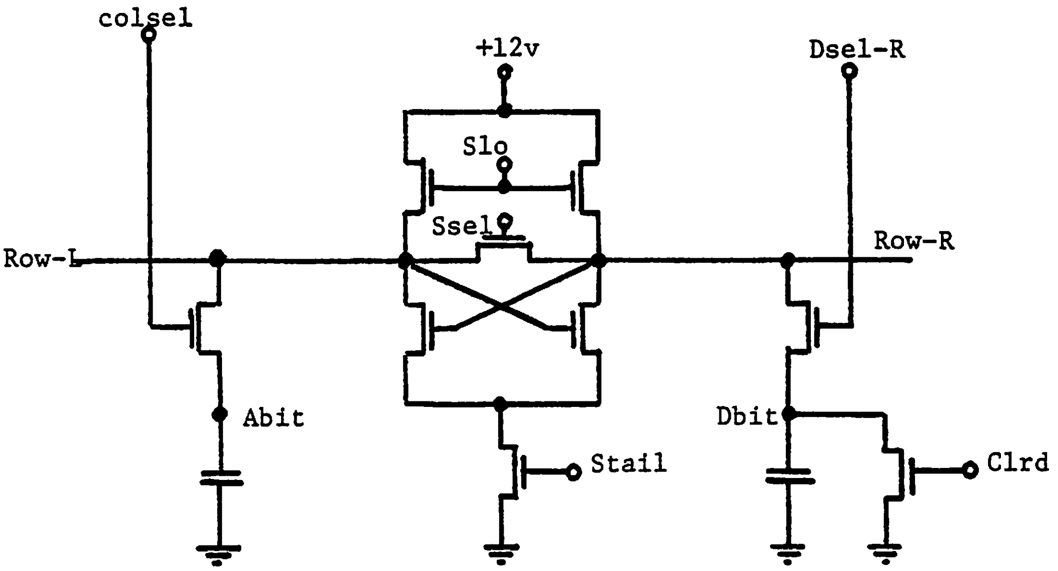


Fig. 6.8 Schematic Diagram of the Sense Amplifier used in the Analysis of the RAM Circuit. Parameter Values are listed in App. 7.

SPLICE ANALYSIS STATISTICS:

```
459 nodes     :  96 circuit, 305 timing,  58 logic
565 elements  :  96 circuit, 411 timing,  58 logic

Setup time    :   14 seconds
Analysis time : 1223 seconds
```

	SPLICE	SPICE2 (estimate)
Total Memory Required for data (words_{10})	8805	220000
Total Central Processor Time (seconds)	1237	21000

Fig. 6.9 Summary of Statistics for the Analysis of the 256-by-1 bit RAM Circuit.

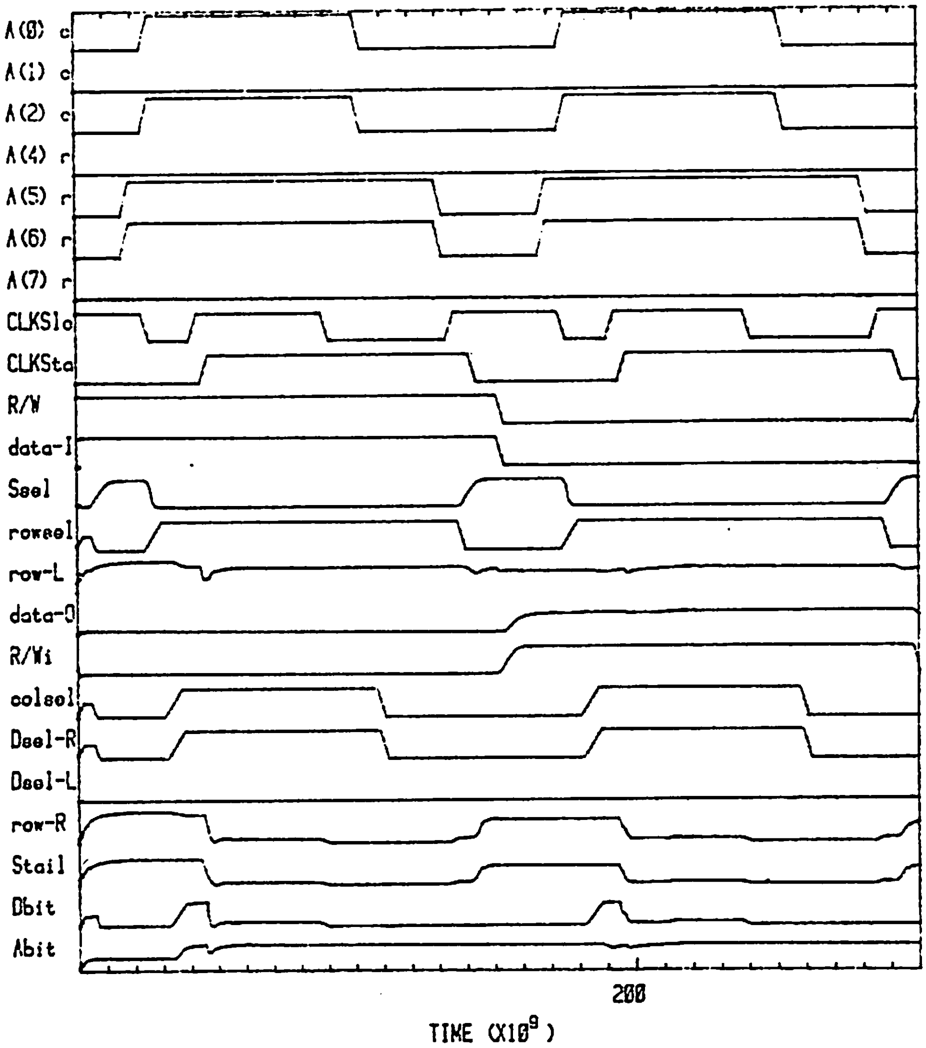


Fig. 6.10 Output Waveforms for a "Write 1 Read 1" Operation on the RAM circuit.

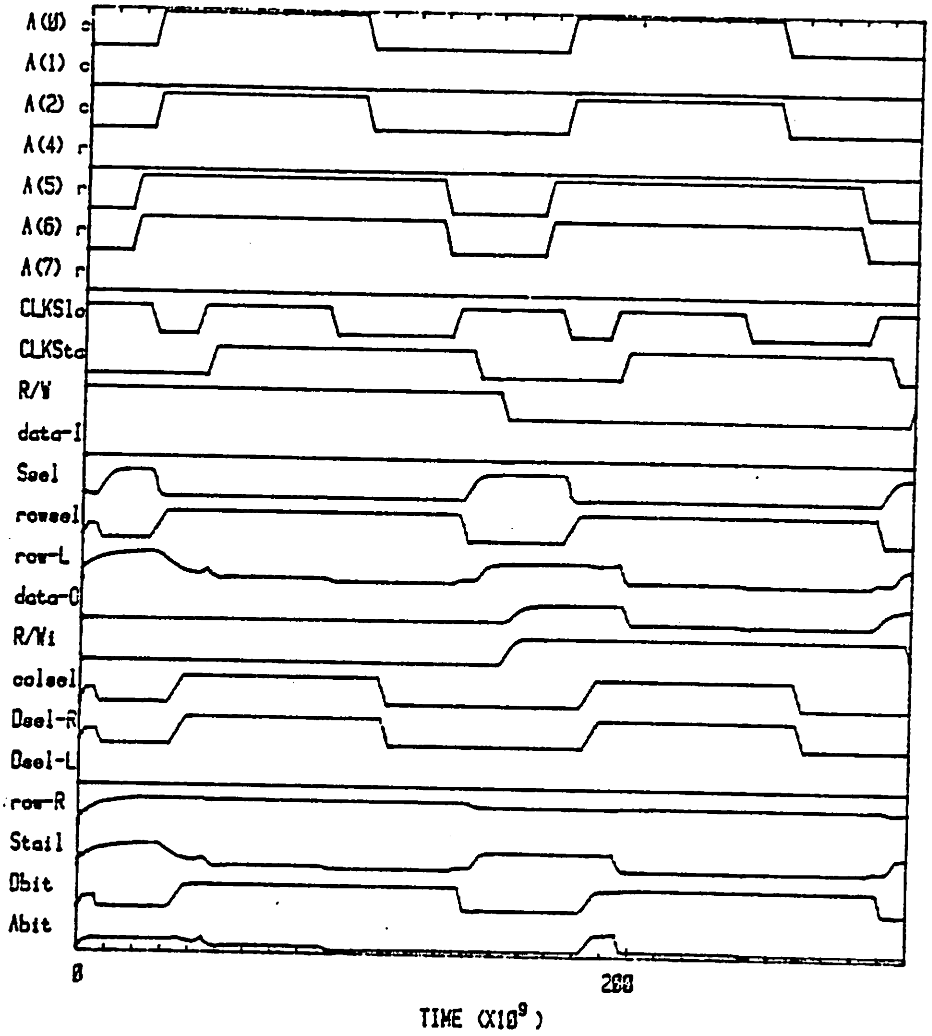


Fig. 6.11 Output waveforms for a "Write 0 Read 0" Operation on the RAM circuit.

were not available. The reason the analysis of this circuit does not show the same improvement over SPICE2 as the previous example is that the program must perform a circuit analysis for the sense amplifier which is relatively time-consuming. Another factor is the parallel nature of the circuit. For the short period when the column line is switching, the fanouts of all the storage transistors in the column must be processed. Since each sense amplifier is connected to one of these transistors via the row, for that period of time all sixteen sense amplifiers are analyzed by the program. The separate analysis of these circuits is more time-consuming than a single circuit analysis in this case.

6.4. The Digital Filter

Fig. 6.12 shows the block diagram of an integrated circuit which performs a digital filtering function. This simulation involved the analysis of 700 MOS transistors which realize the blocks shown as solid lines in Fig. 6.12 and was performed using a timing analysis. The integrated circuit layout for this example is shown in Fig. 6.13 and the input data used for the analysis is included in App. 7. The input data for SPLICE was derived directly from the integrated circuit layout file, which produced the plot of Fig. 6.13, using another computer program [36].

The waveforms produced by SPLICE are shown in Fig. 6.14(a). The circuit is a serial, pipelined filter and the first 10 clock cycles are used to clear the circuit by clocking zeros into all the shift registers, adders, and multiplexers. A reset pulse is then issued and data pulse is entered into the filter. Note that two power supplies (7 and 12 volts) are used in the circuit, hence the different levels seen at the output of the second adder (Sum.B2).

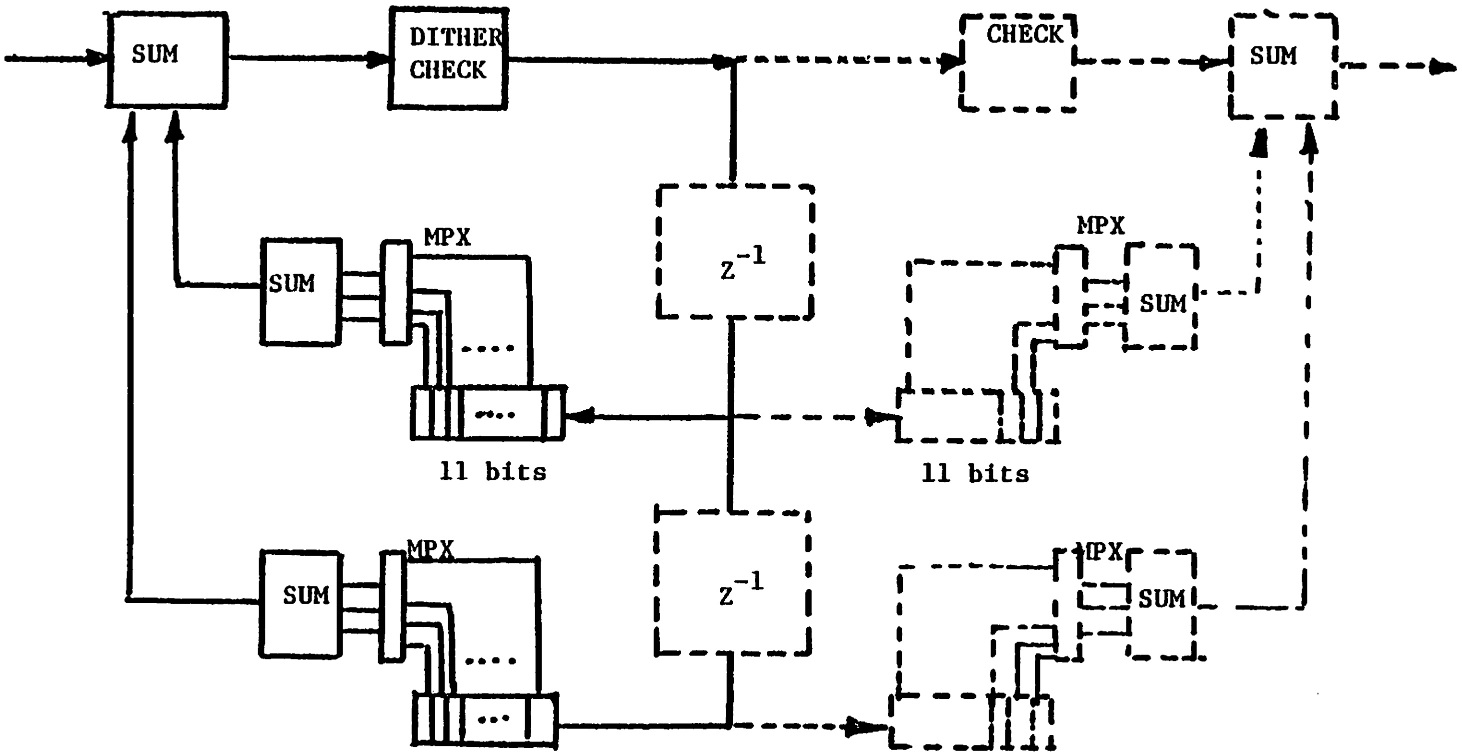
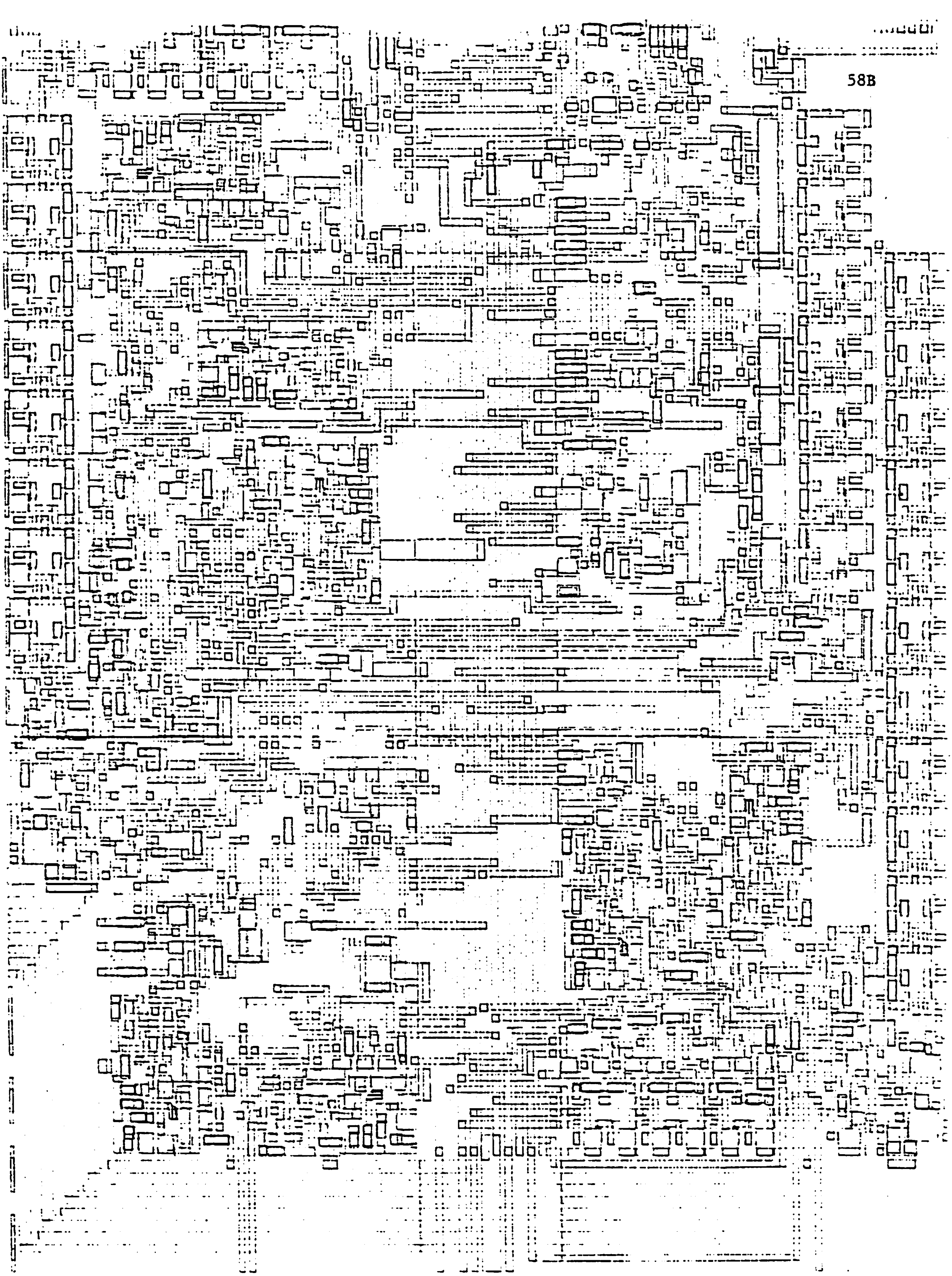


Fig. 6.12 Block Diagram of the Digital Filter Example Simulated on SPLICE.



58B

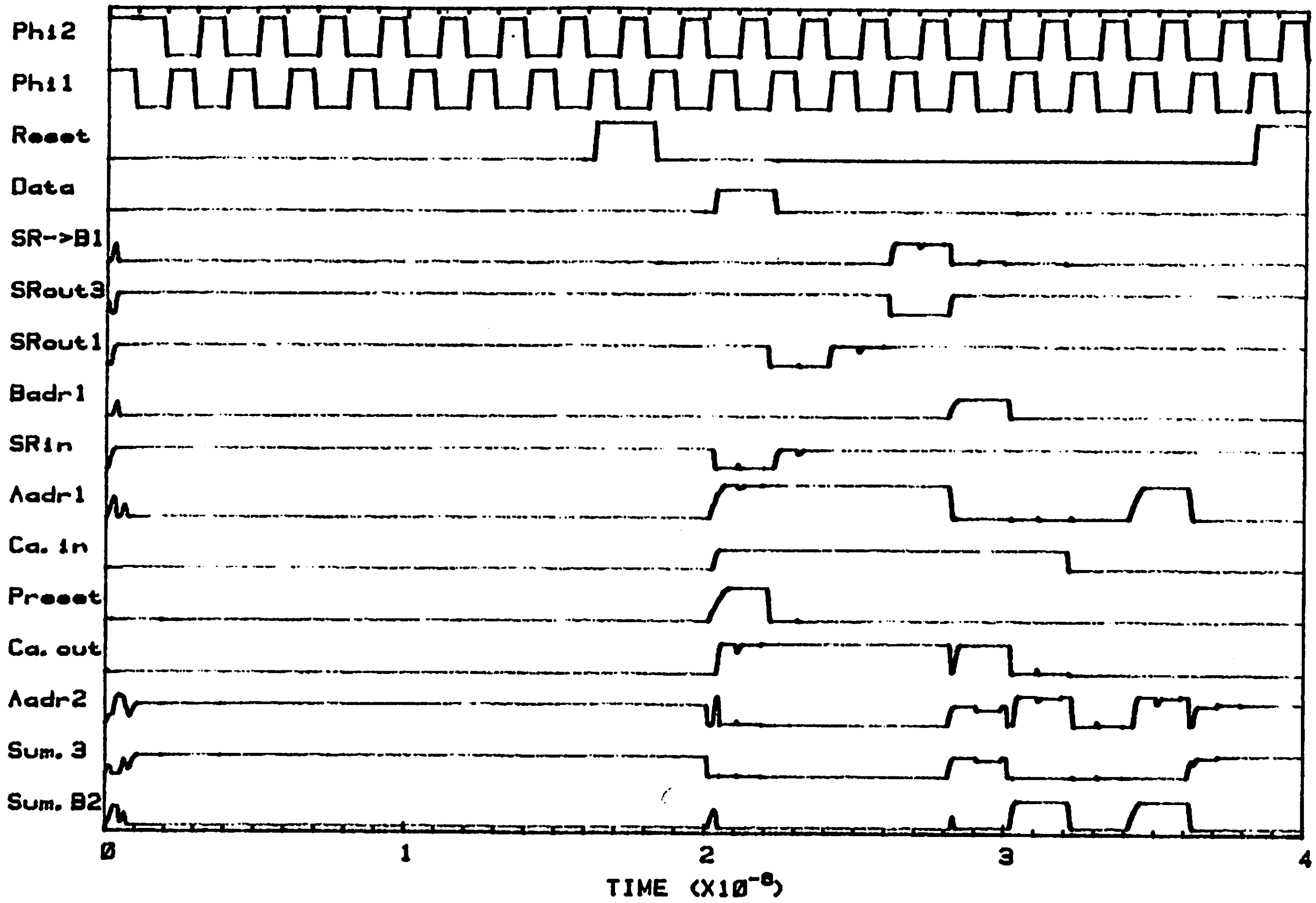


Fig.6.14(a) Output Waveforms for the Digital Filter Example from Program SPLICE.

Fig. 6.14(b) summarizes the statistics produced by SPLICE for this example. The simulation was performed for 4000 units of MRT (1ns) and the simulation time is estimated to be over 100 times less than would be required using SPICE2. The circuit was also simulated using a modified version of MOTIS-C [37] on an HP3000 computer and the output waveforms from this analysis are shown in Fig. 6.15. Note the numerical noise present in the MOTIS-C output, particularly evident at the output of the shift register (SR->B1 node). This noise is not present in the SPLICE output due to the filtering effect of the event scheduler (small, oscillatory changes in node voltages do not cause the fanouts to be scheduled) and the variable timestep algorithm used in SPLICE.

The savings obtained from the event scheduling scheme are illustrated in Fig. 6.16. This plot shows the average number of events per node per unit of MRT and corresponds to the number of nodes processed by the scheduler at each timepoint. The time average of these events indicates that the circuit is less than 20% active and only during clock transitions does the circuit become highly active. This result supports the claim made earlier regarding the relatively low activity of large digital circuits.

Fig. 6.17 shows the average number of timing analyses per node per unit of MRT. Since SPLICE uses an internal timestep for the timing analysis (and hence a number of internal steps may be used for each unit of MRT) this value can be greater than the number of events per node per unit of MRT of Fig. 6.16

```
SPLICE ANALYSIS STATISTICS:

393 nodes        :  393 timing
705 elements     :  705 timing

Element Storage : 2786 words
Node Storage    : 5710 words

Setup Time      :   14 seconds
Analysis Time   :  460 seconds

Events Scheduled:  248626
Timing Analyses :  455460
```

	SPLICE	SPICE2 (estimate)
Total Memory Required for Data (words$_{10}$)	10102	210000
Total Central Processor Time (seconds)	475	50000

Fig. 6.14(b) Summary of Statistics for the Analysis of the Digital Filter Circuit.

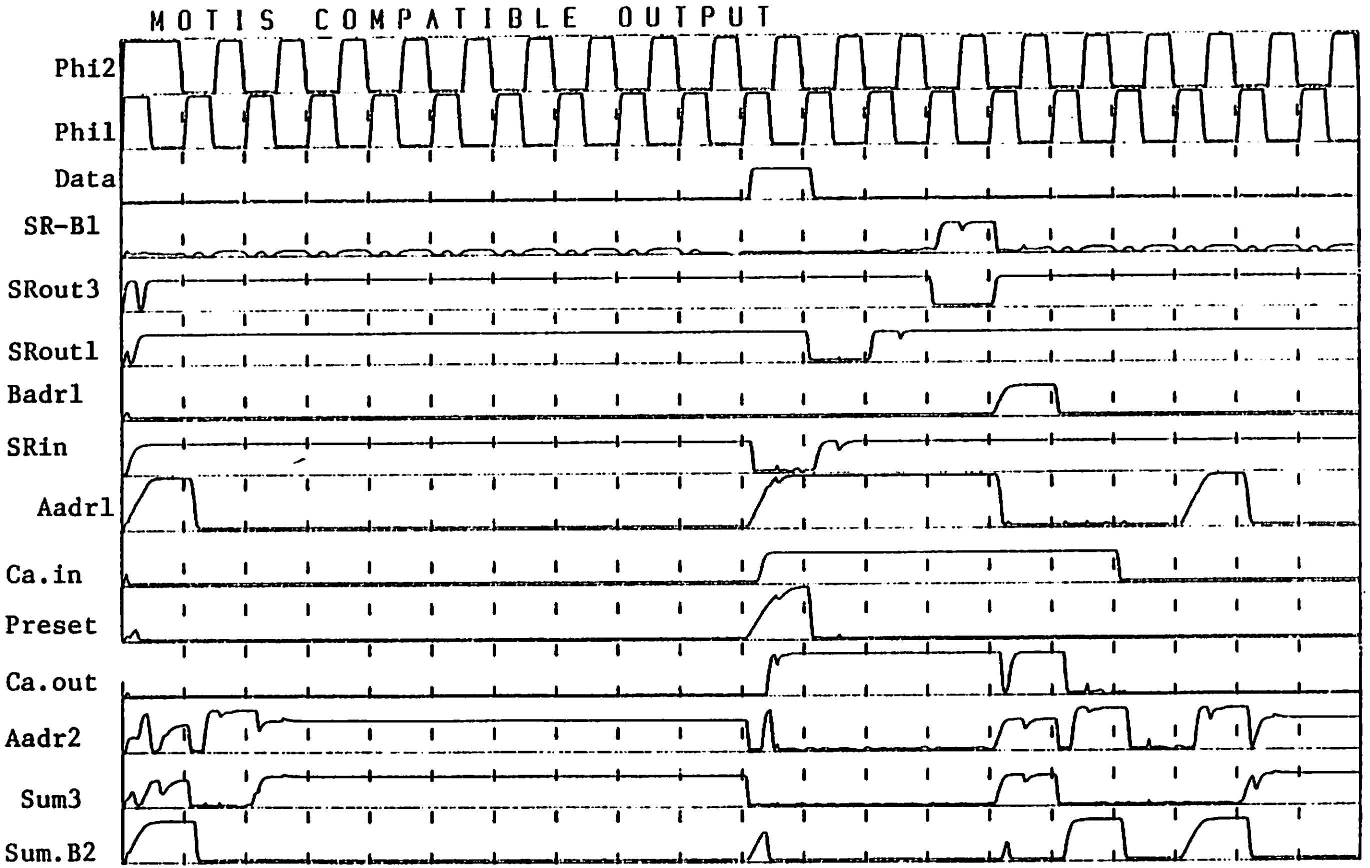


Fig.6.15 Output waveforms for the Digital Filter Example from Program MOTIS-C.

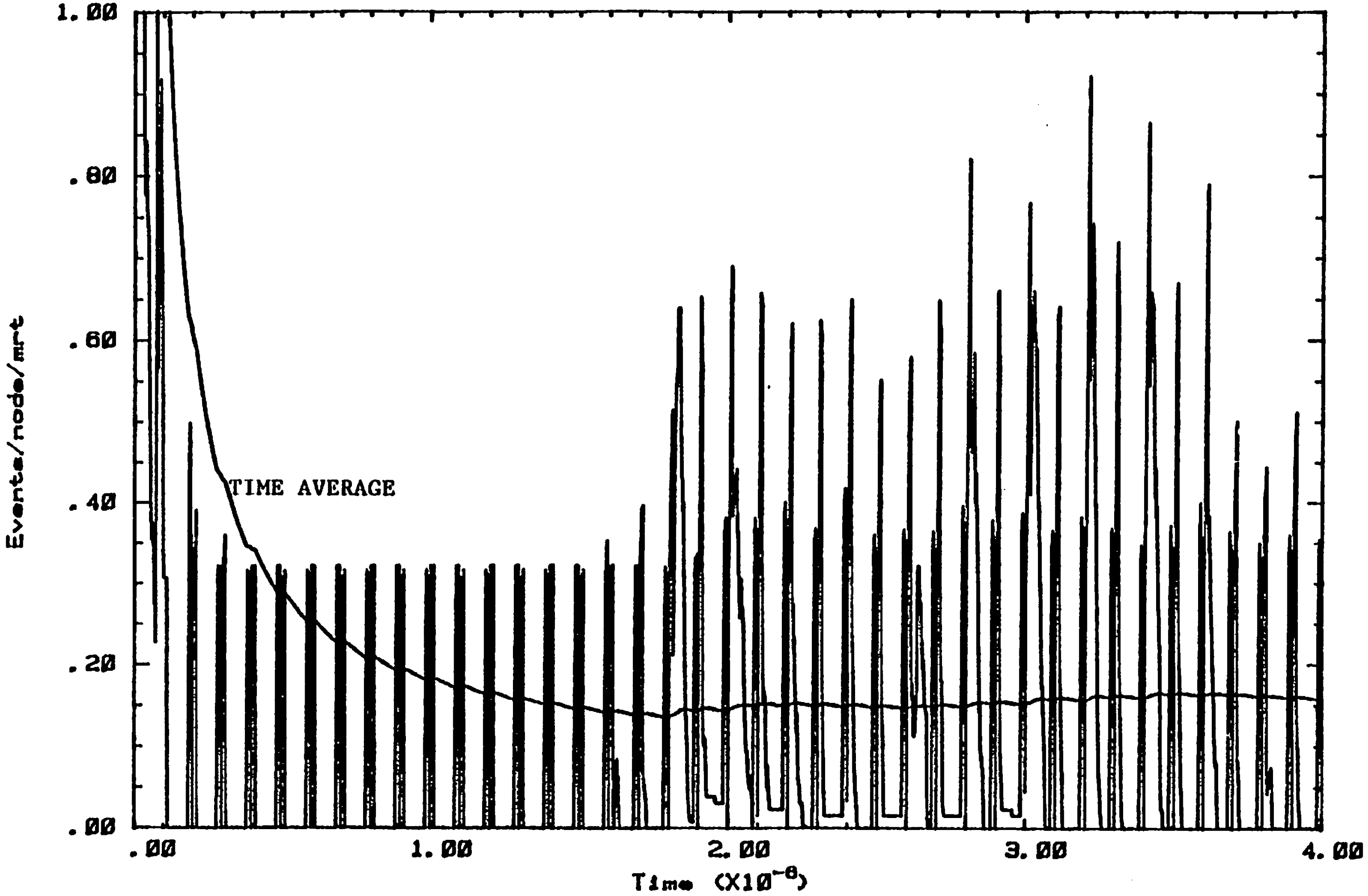


Fig. 6.16 SPLICE analysis events per node per unit of MRT for the Digital Filter Example.

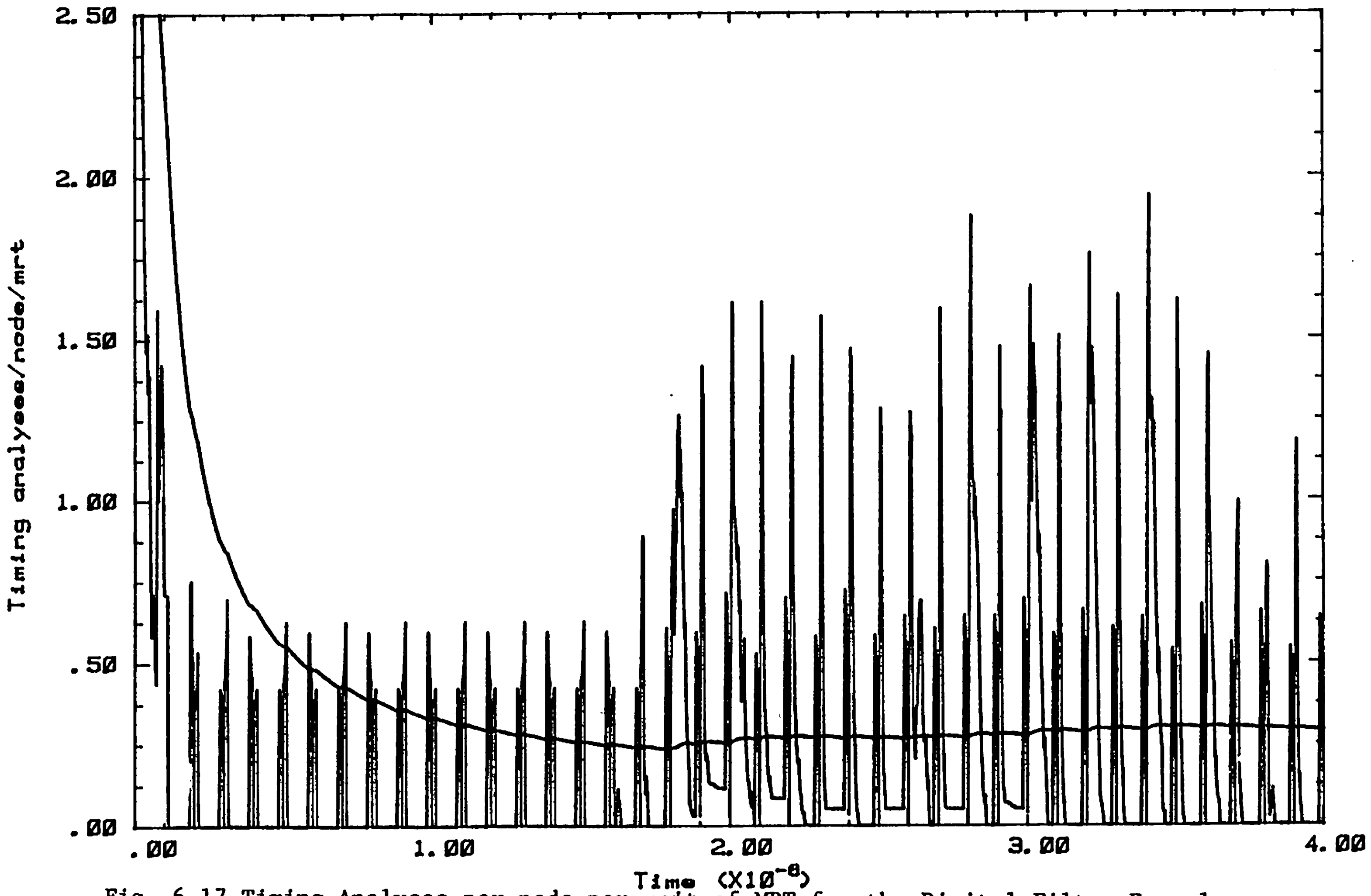


Fig. 6.17 Timing Analyses per node per unit of MRT for the Digital Filter Example.

CHAPTER 7

CONCLUSIONS

Circuit simulation programs which accurately predict the voltage and current waveforms of an electronic circuit are generally too expensive to use for the analysis of LSI circuits. Logic simulators can be used for first-order timing analysis of digital circuits but do not provide the detailed waveform information required in critical parts of the circuit or where tightly-coupled circuit blocks are present.

The hybrid analysis program SPLICE is a simulation program for large-scale integrated circuits which can perform circuit, timing and logic analyses in parallel. While SPLICE is written for use on a CDC 6400 computer, it was designed for use with a minicomputer or similar intelligent terminal as well.

The program uses event-scheduling algorithms, coupled with circuit analysis and timing simulation techniques, to take advantage of some of the properties of large integrated circuits. These techniques reduce the simulation time and memory requirements of the analysis compared with an equivalent circuit simulation. SPLICE is typically between one and three orders of magnitude faster than a circuit-level simulation program when the hybrid analysis techniques are used.

The algorithms developed to permit the parallel analysis of circuit, timing and logic blocks have been presented. Techniques developed for the partitioning of the analysis which exploit the low circuit activity of large integrated circuits have also been described and a number of example simulations for large integrated circuits are presented. These examples include a 256-by-1 bit dynamic RAM circuit, which was simulated using concurrent logic, timing and circuit analyses, and a 700 MOS transistor digital filter circuit which illustrates the time savings of the event-scheduling algorithm.

The use of an event scheduler for the control of all three forms of analysis has proven very efficient and the analysis instability conditions which result from the decoupling of the circuit elements can be resolved in most cases.

SPLICE was written for the analysis of MOS integrated circuits. The analysis techniques used in the program can be extended to other integrated circuit technologies, such as I^2L, and possibly to high-speed bipolar technologies like Emitter Function Logic (EFL). Before the program can be used by a circuit designer, a number of enhancements must be added to both the input and the output processors. These include the ability of the input processor to expand nested subcircuits, as in program SPICE2 [1], [11]. A users manual is also required.

Future work in the area of extending techniques of the type used in SPLICE to Register Transfer Level and Functional level simulation seems promising. Errors are often introduced in the process of converting a schematic circuit description to integrated circuit layout, and visa versa. Efficient algorithms to perform these tasks must also be developed to automate the entire design cycle and reduce the chance of human error. In the simplification process used to generate circuit macromodels, it is essential that the accuracy of the model be maintained. Techniques for aiding the designer in the determination of macromodel parameters (such as gate delays, node capacitances, etc.) are also required. This work may include the use of efficient optimization algorithms for the adjustment of macromodel parameters [15].

APPENDIX 1

A Table Model for an MOS Transistor

The drain current of an MOS transistor may be expressed in terms of the branch voltages

$$I_{DS} = F(V_{DS}, V_{GS}, V_{BS}) \tag{A1-1}$$

where I_{DS} is the drain-to-source current and V_{DS}, V_{GS}, and V_{BS} are the drain-to-source, gate-to-source and bulk-to-source branch voltages respectively, as shown in Fig. A1.1.

A table look-up model requires a table containing values of I_{DS} which is indexed by the independent variables V_{DS}, V_{GS}, and V_{BS}. If n discrete values were chosen for each independent variable the resulting table would require n^3 storage locations. In practice, n should be at least 100 for circuit or timing simulation and hence the resulting table would require 10^6 entries. This number can be reduced substantially via two simple transformations, as shown in this appendix. The transformations will be developed using a simple quadratic MOS model and then extended to a more accurate set of model equations later.

Consider the Shichman-Hodges model[38] as follows:

$$I_{DS}(V_{GS}, V_{DS}, V_{BS}) = K'(V_{GS} - V_{bi} - \gamma\sqrt{\phi_s - V_{BS}} - \frac{V_{DS}}{2})V_{DS}. \tag{A1-2}$$

where V_{bi}, γ, ϕ_s and K' are physical constants for a given device[18]. If the effective gate voltage V_{ge} is defined as

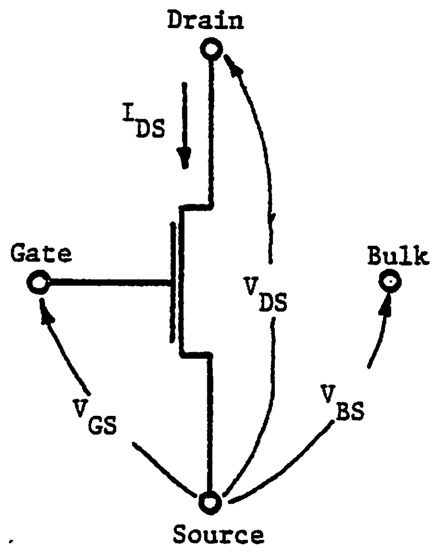


Fig. A1.1 An MOS Transistor and the Branch Conventions.

$$V_{ge} \equiv V_{GS} - V_{bi} - \gamma\sqrt{\phi_s - V_{BS}} \tag{A1-3}$$

$$= V_{GS} - V_t \tag{A1-4}$$

where V_t is the effective threshold voltage, then a linear table may be used to store the values of $V_t(V_{BS})$

$$V_t(V_{BS}) = T_B(V_{BS}). \tag{A1-5}$$

Hence Eqn.(A1-2) may be written in the form:

$$I_{DS}(V_{ge}, V_{DS}) = K'(V_{ge} - \frac{V_{DS}}{2})V_{DS}. \tag{A1-6}$$

If the maximum permitted gate voltage (corresponding to the largest value of the V_{GS} index) is V_{ge}^{max} and

$$\Delta V = V_{ge}^{max} - V_{ge} \tag{A1-7}$$

then Eqn.(A1-6) may be rewritten in the form:

$$I_{DS}(V_{ge}, V_{DS}) = K'(V_{ge}^{max} - \frac{(V_{DS} + \Delta V)}{2})(V_{DS} + \Delta V) - K'(V_{ge}^{max} - \frac{\Delta V}{2})\Delta V \tag{A1-8}$$

$$= I_{DS}(V_{ge}^{max}, V_{DS} + \Delta V) - I_{DS}(V_{ge}^{max}, \Delta V). \tag{A1-9}$$

Thus the drain current at any given V_{DS} and V_{ge} less than V_{ge}^{max} may now be obtained from the single drain characteristic evaluated at V_{ge}^{max} and the second linear table required is:

$$I_{DS}(V_{DS}) = T_D(V_{DS}, V_{ge}^{max}). \tag{A1-10}$$

This transformation is illustrated graphically in Fig. A1.2 and corresponds to moving the origin of the drain characteristic along the characteristic evaluated at V_{ge}^{max}.

A third linear table is required to store the output conductance in saturation for each value of effective gate voltage:

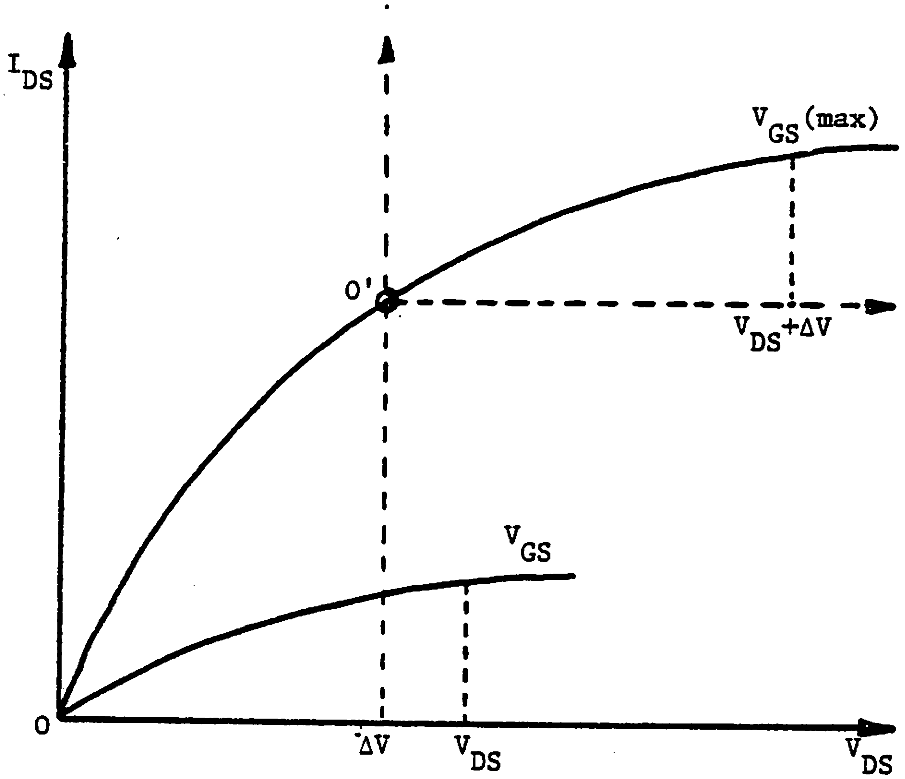


Fig. A1.2 The Effect of the Gate Transformation is to move the Origin along the V_{GS}(max) Characteristic.

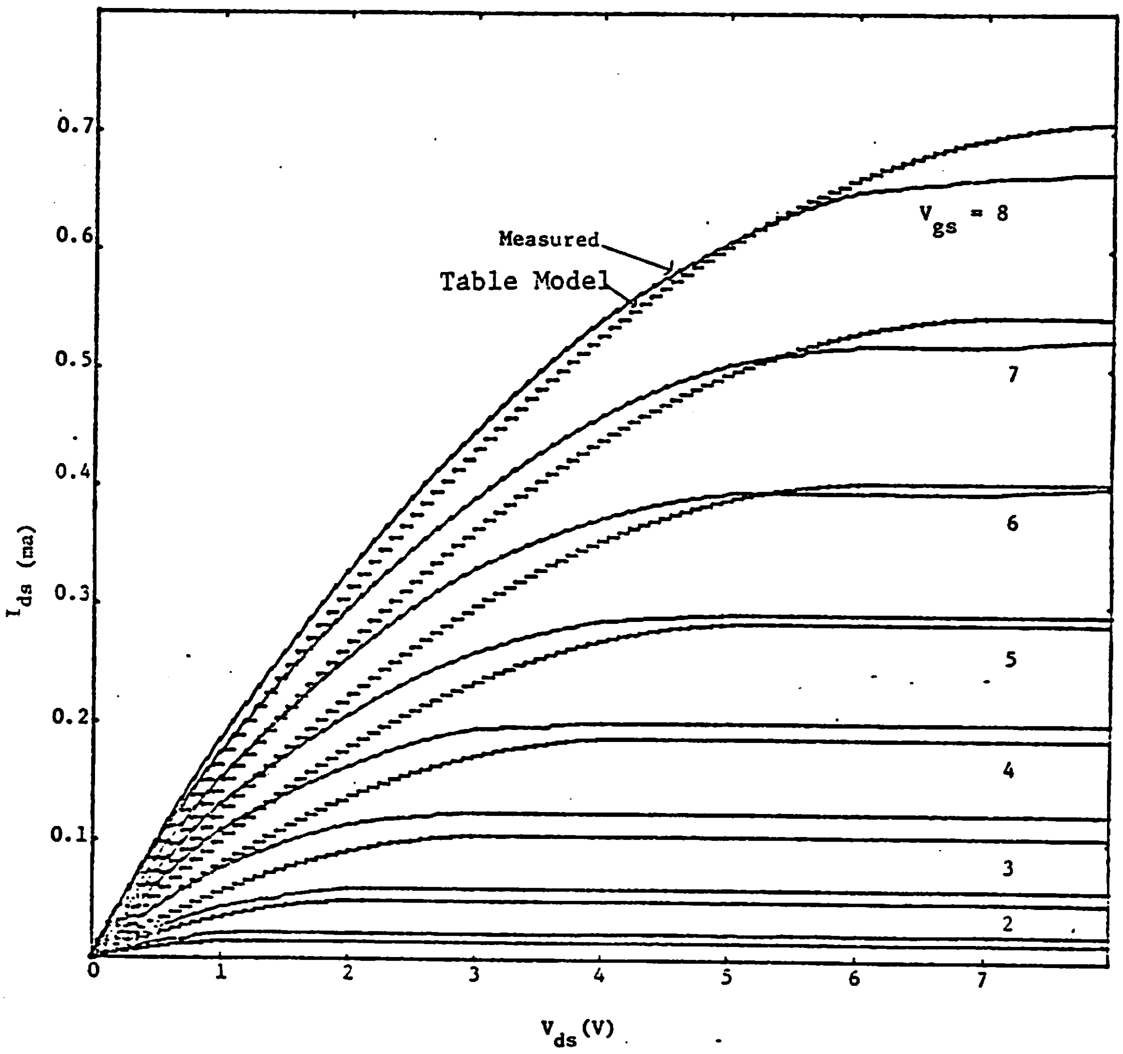


Fig. A1.3 Comparison of Table Look-Up Model with Real Device.
For the device, W= 11u L=22u.

$$G_{sat}(V_{ge}) = T_G(V_{ge}). \tag{A1-11}$$

Hence rather than a table of n^3 entries only 3n locations and two additional subtractions are required to obtain the drain current at any device operating point.

This approach relies on the quadratic form of the Shichman-Hodges model. The transformations may however be used with a more accurate model if some pre-scaling of the current is performed. Consider the following model where the first-order effects of variation of depletion charge stored under the channel with drain voltage are included [39],[18]:

$$\begin{aligned} I_{DS}(V_{GS}, V_{DS}, V_{BS}) = {} & K'(V_{GS} - V_{bi} - \frac{V_{DS}}{2})V_{DS} \\ & - \frac{2}{3}\gamma[(V_{DS} + \phi_s - V_{BS})^{1.5} - (\phi_s - V_{BS})^{1.5}] \end{aligned} \tag{A1-12}$$

where V_{bi}, γ, K' and ϕ_s are as before. It is necessary to maintain the threshold voltage V_t and the saturation current of the device I_{Dsat} under the transformation. The saturation voltage V_{Dsat} is defined as the value of V_{DS} for which $\frac{\partial I_{DS}}{\partial V_{DS}} = 0$ and hence from Eqn.(A1-12)

$$V_{Dsat}(V_{GS}, V_{BS}, V_{bi}) = V_{GS} - V_{bi} + \frac{\gamma^2}{2}[1 - \sqrt{1 + \frac{4}{\gamma^2}(V_{GS} - V_{bi} + \phi_s + V_{BS})}] \tag{A1-13}$$

For the previous model, from Eqn.(A1-2)

$$V_{Dsat}(V_{GS}, V_{BS}, V_{bi}) = V_{GS} - V_{bi} - \gamma\sqrt{\phi_s - V_{BS}}. \tag{A1-14}$$

Now substitute Eqn.(A1-14) into Eqn.(A1-12) and solve for an effective built-in voltage V'_{bi} to maintain V_t under the transformation:

$$V'_{bi} = V_t - \gamma\sqrt{V_{GS} + \phi_s - V_t} \tag{A1-15}$$

where

$$V_t = V_{bi} + \gamma\sqrt{\phi_s - V_{BS}}. \qquad (A1\text{-}16)$$

A scale factor may then be computed at V_{ge}^{max} and used to generate the linear table T_D

$$V_{Dsat} = V_{Dsat}(V_{ge}^{max}, 0, V_{bi}) \qquad (A1\text{-}17)$$

$$V'_{Dsat} = V_{Dsat}(V_{ge}^{max}, 0, V'_{bi}) \qquad (A1\text{-}18)$$

$$I_{Dsat} = I_{DS}(V_{ge}^{max}, V_{Dsat}, 0) \qquad (A1\text{-}19)$$

$$I'_{Dsat} = I_{DS}(V_{ge}^{max}, V'_{Dsat}, 0) \qquad (A1\text{-}20)$$

$$A = \frac{I'_{Dsat}}{I_{Dsat}} \qquad (A1\text{-}21)$$

The table is then generated by using Eqn.(A1-12) scaled by A

$$T_D(V_{GS}, V_{DS}, V_{BS}) = A\, I_{DS}(V_{GS}, V_{DS}, V_{BS}) \qquad (A1\text{-}22)$$

The back-gate table T_B and the output conductance table T_G are generated as described for the simple model.

This table has been used successfully in simulation programs MOTIS-C(App. 8), SPICE2(App. 9) and SPLICE(Ch. 5). An example of the table generated for an actual device is included as Fig. A1.3.

APPENDIX 2

Floating Capacitors in Timing Analysis

The decoupling process used in a timing simulator introduces a numerical *delay* between the circuit nodes and for tightly coupled nodes this delay can cause inaccuracy. This problem arrises with both the floating capacitor and floating MOS transistor, or transfer gate (Fig. A2.1), models which are included in SPLICE and MOTIS-C.

An investigation of the stability and accuracy of the decoupling procedure when used with these elements has shown that for floating capacitors the admittance matrix must be inverted at each timepoint. For transfer gates the decoupling scheme is satisfactory provided the change in drain to source voltage is kept small between timepoints.

Consider the floating capacitor shown in Fig. A2.2. It is connected between nodes 1 and 2 which have conductances G_1 and G_2 to ground and are driven by currents I_1 and I_2 respectively. If the Trapezoidal Rule is used to integrate the capacitor current, at time t_{n+1}:

$$\begin{aligned} I_c^{n+1} &= \frac{2C}{h}(V_c^{n+1} - V_c^n) - I_c^n \\ &= \frac{2C}{h}(\Delta V_2 - \Delta V_1) - I_c^n \end{aligned} \qquad \text{(A2-1)}$$

where $\Delta V_2 = V_2^{n+1} - V_2^n$, $\Delta V_1 = V_1^{n+1} - V_1^n$ and h is the integration timestep. Kirchhoff's current law is then applied to nodes 1 and 2 respectively:

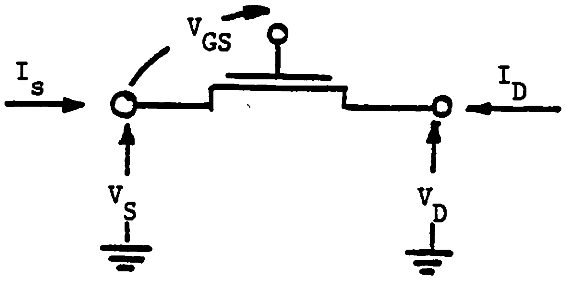


Fig. A2.1 An MOS Transfer Gate

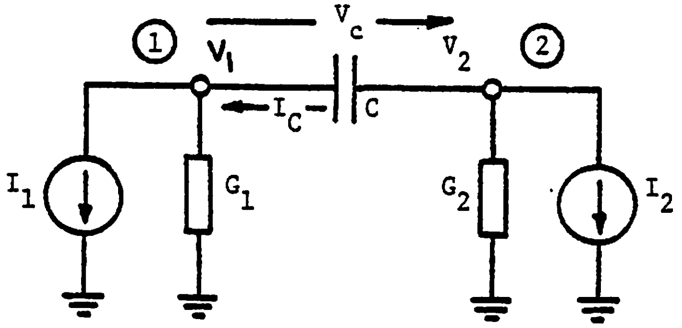


Fig. A2.2 Floating Capacitor and Terminations

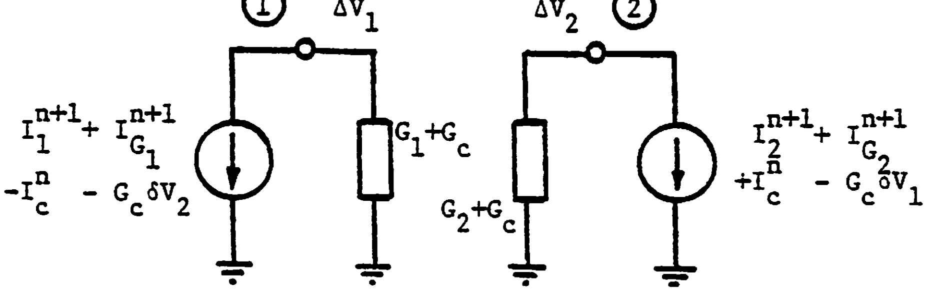

1
ΔV_1
ΔV_2
2
$I_1^{n+1} + I_{G_1}^{n+1}$
$-I_c^n - G_c \delta V_2$
G_1+G_c
G_2+G_c
$I_2^{n+1} + I_{G_2}^{n+1}$
$+I_c^n - G_c \delta V_1$

Fig. A2.3 Equivalent Circuit Model for Decoupled Equations.

$$(G_1 + G_c)\Delta V_1 - G_c\Delta V_2 = I_c^n - I_{G_1}^n - I_1^{n+1} \tag{A2-2}$$

$$-G_c\Delta V_1 + (G_2 + G_c)\Delta V_2 = -I_c^n - I_{G_2}^n - I_2^{n+1} \tag{A2-3}$$

where $G_c = \frac{2C}{h}$

Eqns. A2-2 and A2-3 are coupled and a matrix inversion is required for their solution. This inversion is relatively expensive in a timing simulation program. By replacing ΔV_2 in Eqn. 2-2 by δV_2 and ΔV_1 in Eqn. A2-3 by δV_1, where $\delta V = V^n - V^{n-1}$, equations A2-2 and A2-3 become:

$$(G_1 + G_c)\Delta V_1 = I_c^n - I_{G_1}^n - I_1^{n+1} + G_c\delta V_2 \tag{A2-4}$$

$$(G_2 + G_c)\Delta V_2 = -Ic^n - I_{G_2}^n - I_2^{n+1} + G_c\delta V_1 \tag{A2-5}$$

where all the quantities in the right-hand-side of Eqns. A2-4 and A2-5 are now known following the solution at t_n. This substitution has introduced a numerical delay into the equations relating the voltages at nodes 1 and 2 and will adversely effect the integration accuracy.

The equivalent circuit model for the solution of equations A2-4 and A2-5 is shown in Fig. A2-3. A stability analysis of the above model produces the characteristic polynomial:

$$\left[\frac{h^2G_1G_2}{4C^2} + \frac{hG_1}{2C} + \frac{hG_2}{2C} + 1\right]z^4 + \left[\frac{h^2G_1G_2}{2C^2} - 2\right]z^3 + \left[\frac{h^2G_1G_2}{4C^2} - \frac{hG_1}{2C} - \frac{hG_2}{2C}\right]z^2 + 2z - 1 = 0 \tag{A2-6}$$

The roots of this polynomial, for various values of stepsize h, are plotted in Fig. A2.4. Since the roots lie within the unit circle for all values of h the scheme is unconditionally stable. The presence of complex roots indicates the method

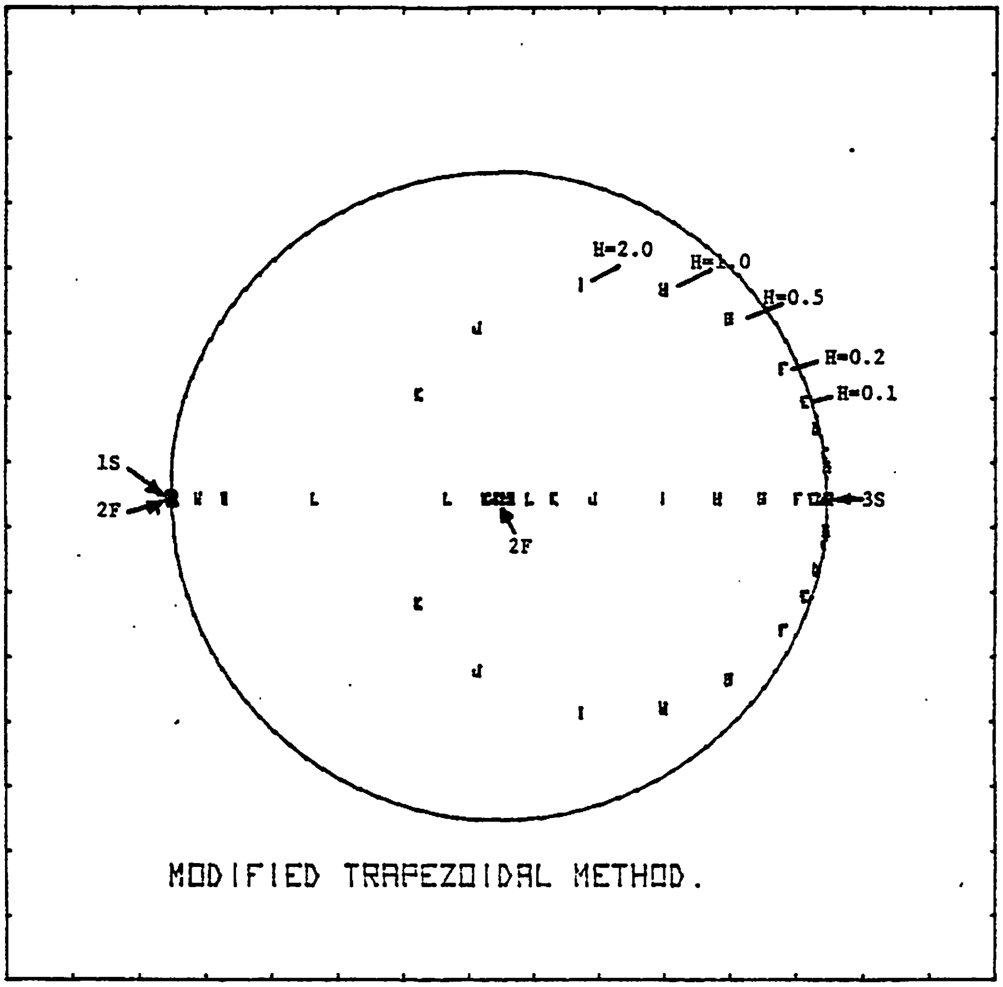


Fig. A2.4 Root Locus for Modified Trapezoidal Method.
"S" Indicates Position of Root $\lim_{h\to 0}$
"F" Indicates Position of Root $\lim_{h\to\infty}$

produces an oscillatory solution for large h.

The inclusion of linear and quadratic predictors to obtain a better estimate for the voltage at the next timepoint does not significantly improve the accuracy of this approximation. For this reason, rather than reduce the timestep to a very small value to avoid oscillation, the admittance matrix for the floating capacitor must be inverted.

The transfer gate is almost always operating in the linear region and hence is not completely bilateral. For this reason, somewhat larger timesteps may be used before the analysis begins to show oscillation and the decoupling scheme described above is used for transfer gates.

APPENDIX 3

SPLICE Input Elements

The following tables describe the input format and parameters for logic, timing and circuit elements used in SPLICE.

MODEL TYPE	DESCRIPTION	PARAMETERS	
BUF	Buffer	Tr Tf	
INV	Inverter	Tr Tf	
AND	AND gate	Tr Tf	
OR	OR gate	Tr Tf	
NAND	NAND gate	Tr Tf	
NOR	NOR gate	Tr Tf	
XOR	Exclusive OR gate	Tr Tf	*
XNOR	Exclusive NOR gate	Tr Tf	*
ROM	Read-Only Memory	Tr Tf No Ni (data)	*
RAM	Random Acess Memory	Tr Tf No Ni	*
NALA	NAND latch	Tr Tf	*
NOLA	NOR latch	Tr Tf	*
TWOP	Two Phase Flip-Flop	Tr Tf	*
DFF	D-type Flip Flop	Tr Tf	*
DFF2	Two Phase D-type Flip-Flop	Tr Tf	*
DSR	Shift Register Section	Tr Tf	*
SAH	Sample and Hold	Tr Tf	*
LSRC	Logic Source	LO L1 D P (breakpoints) -1	
THRESH	Thresholder	VO V1 V*	

Tr = Rise Time, Tf = Fall Time, No = Number of Output Ports, Ni = Number of Input Ports
LO = first logic level, L1 = Second Logic Level, D = Source Delay, P = Source Period,
VO = Voltage Level for Logic 0, V1 = Voltage Level for Logic 1, V* = level for Logic *.
* indicates the element has not been fully implemented or extensively tested.

Table 3.1 SPLICE Logic Elements and their Parameters

MODEL TYPE	DESCRIPTION	PARAMETERS	
GCAPR	Grounded Capacitor	C	
NDRIV	N-Channel Driver Transistor	W/L Vt Kp Gam Phi Lam	
PDRIV	P-Channel Driver (eqns)	W/L Vt Kp Gam Phi Lam	
NLOAD	N-Channel Load (eqns)	W/L Vt Kp Gam Phi Lam Vdd	
PLOAD	P-Channel Load (eqns)	W/L Vt Kp Gam Phi Lam Vdd	
CINVT	CMOS Inverter	W/L Vt Kp Gam Phi Lam Nst Vdd	*
FCAPR	Floating Capacitor	C	
NTXG	N-Channel Transfer Gate	W/L Vt Kp Gam Phi Lam	
PTXG	P-Channel Transfer Gate	W/L Vt Kp Gam Phi Lam	
NTTDV	N-Ch. Driver (Table)	W/L Vt Kp Gam Phi Lam Nst	
PTTDV	P-Ch. Driver (Table)	W/L Vt Kp Gam Phi Lam Nst	
NTTLO	N-Ch. Load (Table)	W/L Vt Kp Gam Phi Lam Nst Vdd	
PTTLO	P-Ch. Load (Table)	W/L Vt Kp Gam Phi Lam Nst Vdd	
NTTXG	N-Ch. Transfer (Table)	W/L Vt Kp Gam Phi Lam Nst	
PTTXG	P-Ch. Transfer (Table)	W/L Vt Kp Gam Phi Lam Nst	
TSRC	Timing Voltage Source	V0 V1 D P (breakpoints) -1	
LTV	Logic-to-Voltage Converter	Tr Tf V0 V1 V*	
LTI	Logic-to-Current Converter	Tr Tf I0 I1 I*	*

W/L = width to Length ratio, Vt=zero bias threshold voltage, Kp=Transconductance per unit gate voltage, Phi = surface potential for strong inversion, Lam = output conductance factor for saturation region, Nst=Number of table steps, Vdd=effective supply voltage for loads, V0 = voltage first, V1 + second voltage, D=delay, P=period, Tr=risetime, Tf=falltime, * indicates the element has not been fully implemented or extensively tested.

Table 3.2 SPLICE Timing Elements and their Parameters.

MODEL TYPE	DESCRIPTION	PARAMETERS	
NMOSE	N-Ch. MOS transistor (Eqns)	W/L Vt Kp Gam Phi Lam	
PMOSE	P-Ch. MOS tr. (Eqns)	W/L Vt Kp Gam Phi Lam	
NMOST	N-Ch. MOS tr. (1d table)	W/L Vt Kp Gam Phi Lam	
PMOST	P-Ch. MOS tr. (1d table)	W/L Vt Kp Gam Phi Lam	
NMOSC	N-Ch. MOS tr. (empirical)	(measured table values)	*
PMOSC	P-Ch. MOS tr. (empirical)	(measured table values)	*
R	Resistor	value	
C	Capacitor	value	

W/L = width-to-length ratio, Vt=zero bias threshold voltage, Kp=transconductance per unit gate voltage, Phi = surface potential for strong inversion, Lam = output conductance factor for saturation region, * indicates the element has not been fully implemented or extensively tested.

Table 3.3 SPLICE Circuit Elements and their parameters.

STATEMENT	DESCRIPTION	
PRINT	Print node voltages	
PLOT	Plot node voltages	
TIME	Set simulation time and MRT	
LOPTS	Set logic analysis options	
TOPTS	Set timting analysis options	
COPTS	Set circuit analysis options	
GO	Begin analysis	
END	End this simulation run	
SETDC	Set node voltages for a dc analysis	*
SETTR	Set node voltages for a transient analysis	
DCOP	Compute the dc operating point	*
SUST	Simulate until stable (scheduler queue is empty)	*
UNDEF	List all nodes whose signal levels are not 1 or 0	*

* indicates command not fully implemented or extensively tested.

Table 3.4 SPLICE Analysis Commands

APPENDIX 4

Input Processor Data Structure

Fig. A4.1 shows the structure of the file produced by the input processor. Device models are stored first followed by elements, control statements and the node map which translates the user-defined signal paths to SPLICE internal node numbers.

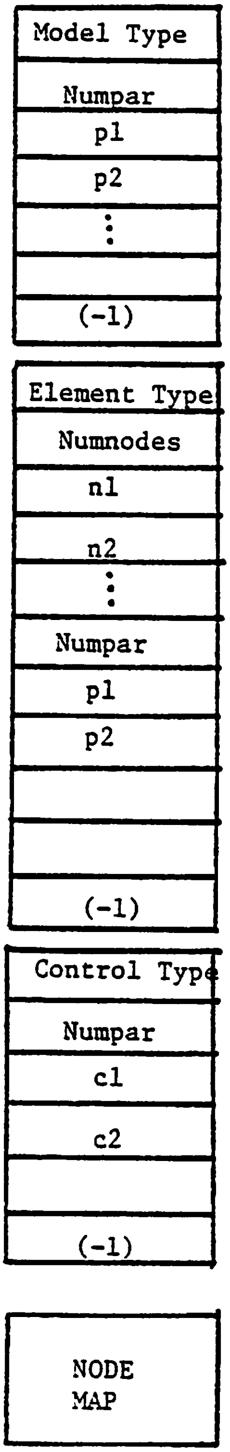
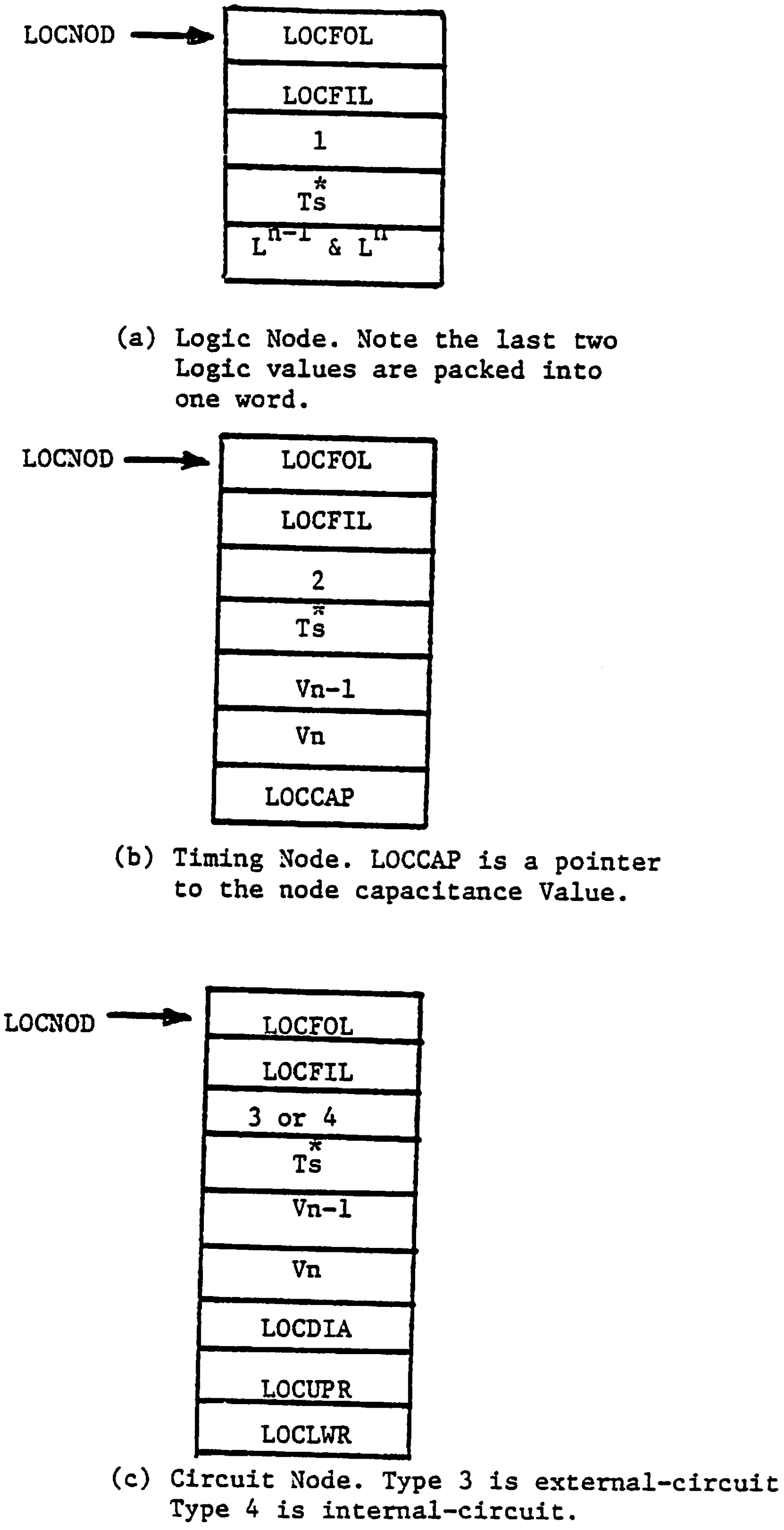


Fig. A4.1 Structure of the file produced by the Input Pre-processor. Numpar is the number of parameters and Numnod is the number of nodes.

APPENDIX 5

Setup and Analysis Data Structures

The following figures illustrate the setup and analysis data structures, with program variable names used wherever appropriate.



(a) Logic Node. Note the last two Logic values are packed into one word.

(b) Timing Node. LOCCAP is a pointer to the node capacitance Value.

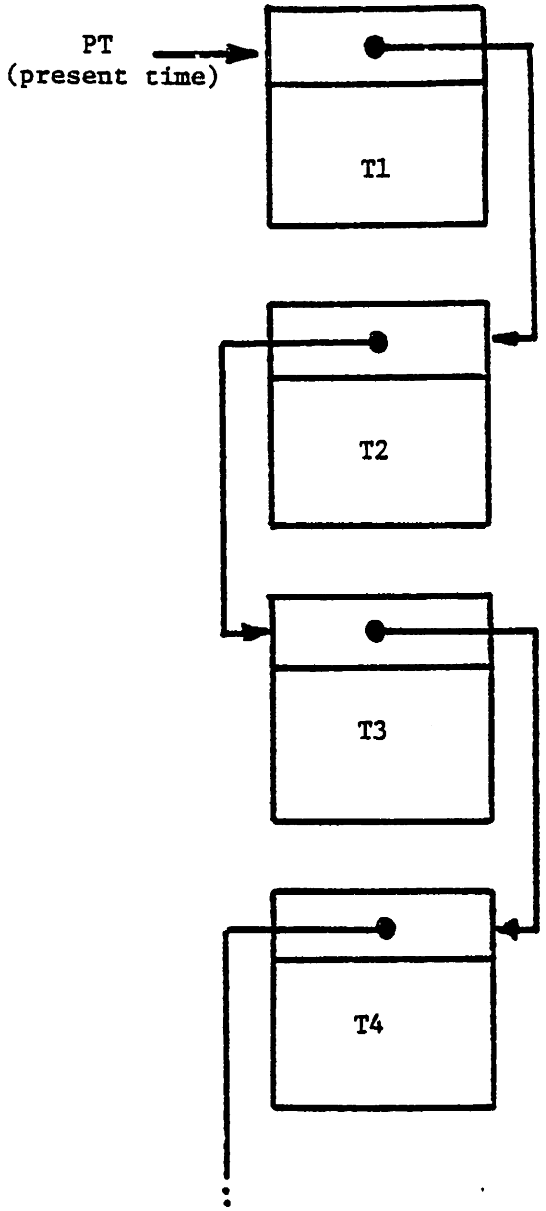
(c) Circuit Node. Type 3 is external-circuit Type 4 is internal-circuit.

Fig.A5.1 Data Structure for Logic, Timing and Circuit Nodes

The fanout list origin is then used by the scheduler to schedule the fanouts to be processed at the appropriate time in the future.

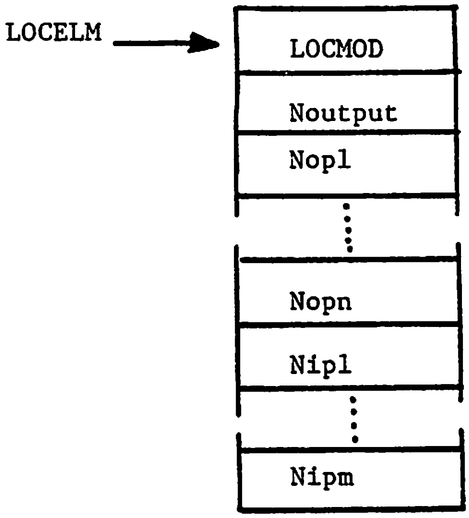
Thus the lists for an element contains the element connection information and a pointer to the model information. The node list contains information about the state of the signal level at the node, pointers to fanout and fanin lists, and some parameter information.

4.2.3. Data Structure of the Time Queue Efficient processing of the time queue is critical to the overall performance of the program. The time queue contains the fanout lists of all nodes scheduled to be processed and the time at which they are scheduled to be processed. A simple way of storing these entries is the linked list structure of Fig. 4.6. The scheduler moves along this time-ordered linked list where each entry in the list contains the time the fanouts of a node are to be processed and a pointer to the fanout list of the node. As each event is executed, the scheduler processes each element on the fanout list in an arbitrary order. Once the elements on th list have been processed, the simulator moves to the next entry in the time queue. It may contain a list to be processed at the same time as the last list or a list to be processed some time in the future. Any events generated as each list is processed are inserted in the time queue at the time they are due to occur. Unfortunately, if an event is scheduled to occur more than one unit of MRT into the future, the process of inserting it in the time queue may involve searching many entries already in the queue before its place can be determined and hence this scheme is relatively inefficient. By observing that most events occur within a few units of MRT from the present time (PT), a more efficient scheme may be used [5], [33].

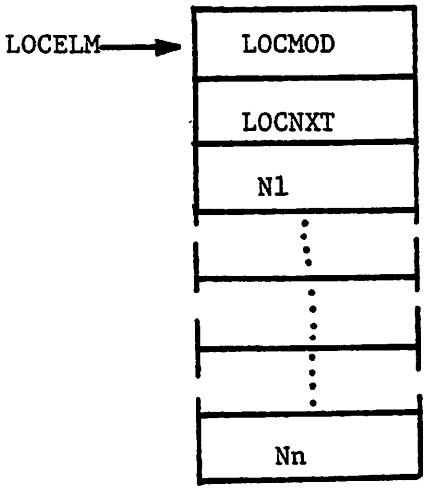


T4 > T3 > T2 > T1 = PT

Fig. 4.6 Simple Linked-List Time Queue



(a) Structure of Logic and Timing Elements
Noutput is the number of output nodes.
Nopl-Nopn are the output node pointers and
Nipl-Nipm are the input node pointers.



(b) Structure of Circuit Element. Same as above except the second word is used to link the elements associated with the same Model Control Block.

Fig. A5.2 Data Structure for Logic, Timing and Circuit Elements.

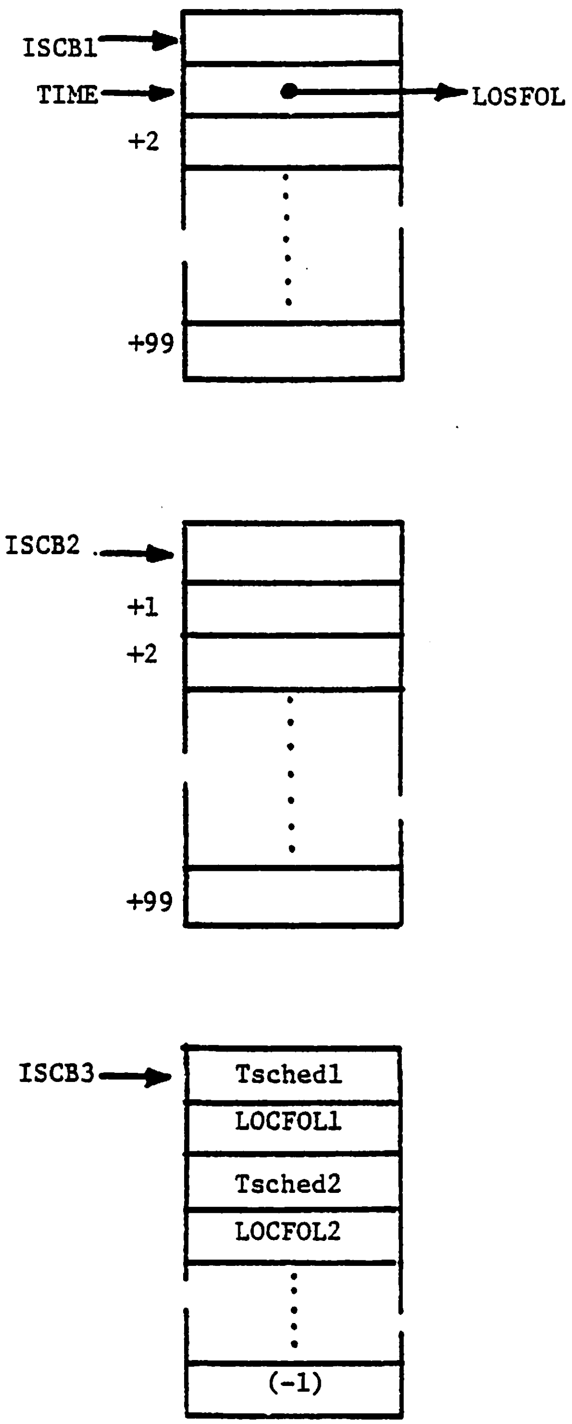


Fig. A5.3 Data Structure for the Scheduler.
TIME is the present analysis time
and LOSFOL is the scheduled fanout list.
See subroutine SWAP for more details

APPENDIX 6

Output Processor Data Structure

Fig. A6.1 shows the structure of the file produced by SPLICE for the output post-processor.

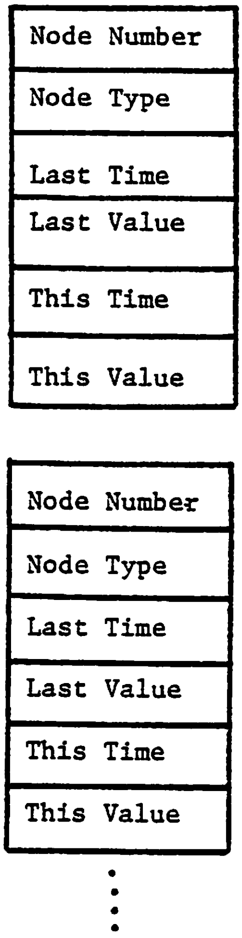


Fig. A6.1 Format of the file produced for the Output Post-Processor. Last Time and Last Value are the previously Scheduled Time and Value respectively.

APPENDIX 7

Input Data for Example Circuits

This appendix contains the input data used by SPLICE for the analysis of the Binary-to-Hexidecimal Decoder, 256-by-1 bit Dynamic RAM and Digital Filter examples of Chap. 6.

```
Y-AXIS ONE-OF-SIXTEEN DECODER: HYBRID ANALYSIS (LOGIC AND TIMING)
*
* MODELS
MODEL  S0 LSRC( 0 1 0NS 100NS 0NS  40NS  50NS  90NS 100NS -1 )
MODEL  S1 LSRC( 0 1 0NS 200NS 0NS  90NS 100NS 190NS 200NS -1 )
MODEL  S2 LSRC( 0 1 0NS 400NS 0NS 130NS 200NS 390NS 400NS -1 )
MODEL  S3 LSRC( 0 1 0NS 800NS 0NS 390NS 400NS 790NS 800NS -1 )
MODEL IDRI NDRIV( 2.00 0.5 20U 0.7 0.6 0.0 5  100 )
MODEL ILOD NLOAD( 1.00 -5  20U 0.7 0.6 0.0 5  100 )
MODEL INV  INV  ( 4NS 2NS )
MODEL NOR  NOR  ( 4NS 2NS )
MODEL LTV  LTV  ( 6NS 3NS 0 5 2.5 )
MODEL   C GCAPR
*
* INPUT SOURCES
S0 1 S0
S1 2 S1
S2 3 S2
S3 4 S3
*
* INPUT INVERTERS
I1  37  1 INV
I2  38  2 INV
I3  39  3 INV
I4  40  4 INV
*
* OUTPUT INVERTERS
I5   5 41 IDRI
I6   6 42 IDRI
I7   7 43 IDRI
I8   8 44 IDRI
I9   9 45 IDRI
I10 10 46 IDRI
I11 11 47 IDRI
I12 12 48 IDRI
I13 13 49 IDRI
I14 14 50 IDRI
I15 15 51 IDRI
I16 16 52 IDRI
I17 17 53 IDRI
I18 18 54 IDRI
I19 19 55 IDRI
I20 20 56 IDRI
L5   5     ILOD
L6   6     ILOD
L7   7     ILOD
L8   8     ILOD
L9   9     ILOD
L10  10    ILOD
L11  11    ILOD
L12  12    ILOD
L13  13    ILOD
L14  14    ILOD
L15  15    ILOD
L16  16    ILOD
L17  17    ILOD
L18  18    ILOD
L19  19    ILOD
L20  20    ILOD
*
* NOR GATES
N0  21  1  2  3  4 NOR
```

```
N1   22 37  2  3  4 NOR
N2   23  1 38  3  4 NOR
N3   24 37 38  3  4 NOR
N4   25  1  2 39  4 NOR
N5   26 37  2 39  4 NOR
N6   27  1 38 39  4 NOR
N7   28 37 38 39  4 NOR
N8   29  1  2  3 40 NOR
N9   30 37  2  3 40 NOR
N10  31  1 38  3 40 NOR
N11  32 37 38  3 40 NOR
N12  33  1  2 39 40 NOR
N13  34 37  2 39 40 NOR
N14  35  1 38 39 40 NOR
N15  36 37 38 39 40 NOR
*
* LOGIC-TO-VOLTAGE CONVERTERS
V1   41 21 LTV
V2   42 22 LTV
V3   43 23 LTV
V4   44 24 LTV
V5   45 25 LTV
V6   46 26 LTV
V7   47 27 LTV
V8   48 28 LTV
V9   49 29 LTV
V10  50 30 LTV
V11  51 31 LTV
V12  52 32 LTV
V13  53 33 LTV
V14  54 34 LTV
V15  55 35 LTV
V16  56 36 LTV
*
* NODE CAPACITANCES
C5    5 C 0.08P
C6    6 C 0.08P
C7    7 C 0.08P
C8    8 C 0.08P
C9    9 C 0.08P
C10  10 C 0.08P
C11  11 C 0.08P
C12  12 C 0.08P
C13  13 C 0.08P
C14  14 C 0.08P
C15  15 C 0.08P
C16  16 C 0.08P
C17  17 C 0.08P
C18  18 C 0.08P
C19  19 C 0.08P
C20  20 C 0.08P
*
* ANALYSIS REQUESTS
OPTS 1  7.0
TOPTS 0.1   0.05   1000 0.1
TIME 2.0NS 800NS
PLOT 1, 2, 3, 4
PLOT 5, 6, 7, 8, 9, 10, 11, 12
PLOT 13, 14, 15, 16, 17, 18, 19, 20
*
GO
*
```

```
Y-AXIS ONE-OF-SIXTEEN DECODER: TIMING ANALYSIS
*
* MODELS
MODEL   S0 TSRC( 0 5 0NS 100NS 0NS  40NS  50NS  90NS 100NS -1 )
MODEL   S1 TSRC( 0 5 0NS 200NS 0NS  90NS 100NS 190NS 200NS -1 )
MODEL   S2 TSRC( 0 5 0NS 400NS 0NS 190NS 200NS 390NS 400NS -1 )
MODEL   S3 TSRC( 0 5 0NS 800NS 0NS 390NS 400NS 790NS 800NS -1 )
MODEL IDRI NDRIV( 2.00 0.5 20U 0.7 0.6 0.0 5  100 )
MODEL ILOD NLOAD( 1.00 -5  20U 0.7 0.6 0.0 5  100 )
MODEL NDRI NDRIV( 1.50 0.5 20U 0.7 0.6 0.0 5  100 )
MODEL NLOD NLOAD( 0.75 -5  20U 0.7 0.6 0.0 5  100 )
MODEL    C GCAPR
*
* INPUT SOURCES
S0 1 S0
S1 2 S1
S2 3 S2
S3 4 S3
*
* INPUT INVERTERS
I1  37  1 IDRI
I2  38  2 IDRI
I3  39  3 IDRI
I4  40  4 IDRI
L1  37    ILOD
L2  38    ILOD
L3  39    ILOD
L4  40    ILOD
*
* OUTPUT INVERTERS
I5   5 21 IDRI
I6   6 22 IDRI
I7   7 23 IDRI
I8   8 24 IDRI
I9   9 25 IDRI
I10 10 26 IDRI
I11 11 27 IDRI
I12 12 28 IDRI
I13 13 29 IDRI
I14 14 30 IDRI
I15 15 31 IDRI
I16 16 32 IDRI
I17 17 33 IDRI
I18 18 34 IDRI
I19 19 35 IDRI
I20 20 36 IDRI
L5   5    ILOD
L6   6    ILOD
L7   7    ILOD
L8   8    ILOD
L9   9    ILOD
L10  10   ILOD
L11  11   ILOD
L12  12   ILOD
L13  13   ILOD
L14  14   ILOD
L15  15   ILOD
L16  16   ILOD
L17  17   ILOD
L18  18   ILOD
L19  19   ILOD
L20  20   ILOD
```

```
*
* NOR GATES
N0   21   1 NORI
N0   21   2 NORI
N0   21   3 NORI
N0   21   4 NORI
*
N1   22  37 NORI
N1   22   2 NORI
N1   22   3 NORI
N1   22   4 NORI
*
N2   23   1 NORI
N2   23  38 NORI
N2   23   3 NORI
N2   23   4 NORI
*
N3   24  37 NORI
N3   24  38 NORI
N3   24   3 NORI
N3   24   4 NORI
*
N4   25   1 NORI
N4   25   2 NORI
N4   25  39 NORI
N4   25   4 NORI
*
N5   26  37 NORI
N5   26   2 NORI
N5   26  39 NORI
N5   26   4 NORI
*
N6   27   1 NORI
N6   27  38 NORI
N6   27  39 NORI
N6   27   4 NORI
*
N7   28  37 NORI
N7   28  38 NORI
N7   28  39 NORI
N7   28   4 NORI
*
N8   29   1 NORI
N8   29   2 NORI
N8   29   3 NORI
N8   29  40 NORI
*
N9   30  37 NORI
N9   30   2 NORI
N9   30   3 NORI
N9   30  40 NORI
*
N10  31   1 NORI
N10  31  38 NORI
N10  31   3 NORI
N10  31  40 NORI
*
N11  32  37 NORI
N11  32  38 NORI
N11  32   3 NORI
N11  32  40 NORI
*
```

```
N12 33  1 NDRI
N12 33  2 NDRI
N12 33 39 NDRI
N12 33 40 NDRI
*
N13 34 37 NDRI
N13 34  2 NDRI
N13 34 39 NDRI
N13 34 40 NDRI
*
N14 35  1 NDRI
N14 35 38 NDRI
N14 35 39 NDRI
N14 35 40 NDRI
*
N15 36 37 NDRI
N15 36 38 NDRI
N15 36 39 NDRI
N15 36 40 NDRI
*
M0  21    NLOD
M1  22    NLOD
M2  23    NLOD
M3  24    NLOD
M4  25    NLOD
M5  26    NLOD
M6  27    NLOD
M7  28    NLOD
M8  29    NLOD
M9  30    NLOD
M10 31    NLOD
M11 32    NLOD
M12 33    NLOD
M13 34    NLOD
M14 35    NLOD
M15 36    NLOD
*
* NODE CAPACITANCES
C5   5 C 0.08P
C6   6 C 0.08P
C7   7 C 0.08P
C8   8 C 0.08P
C9   9 C 0.08P
C10 10 C 0.08P
C11 11 C 0.08P
C12 12 C 0.08P
C13 13 C 0.08P
C14 14 C 0.08P
C15 15 C 0.08P
C16 16 C 0.08P
C17 17 C 0.08P
C18 18 C 0.08P
C19 19 C 0.08P
C20 20 C 0.08P
C21 21 C 0.10P
C22 22 C 0.10P
C23 23 C 0.10P
C24 24 C 0.10P
C25 25 C 0.10P
C26 26 C 0.10P
C27 27 C 0.10P
C28 28 C 0.10P
```

```
C29  29  C  0.10P
C30  30  C  0.10P
C31  31  C  0.10P
C32  32  C  0.10P
C33  33  C  0.10P
C34  34  C  0.10P
C35  35  C  0.10P
C36  36  C  0.10P
C37  37  C  0.08P
C38  38  C  0.08P
C39  39  C  0.08P
C40  40  C  0.08P
*
* ANALYSIS REQUESTS
OPTS 1  7.0
TOPTS 0.1   0.05   1000 0.1
TIME 1.5NS 800NS
PLOT 1, 2, 3, 4
PLOT 5, 6, 7, 8, 9, 10, 11, 12
PLOT 13, 14, 15, 16, 17, 18, 19, 20
*
GO
*
END
```

```
256-BIT RAM: HYBRID ANALYSIS (CIRCUIT, TIMING & LOGIC)
*
* MODELS
* INPUT SOURCE MODELS FOR ADDRESS LINES AND CLOCKS
MODEL  A0 LSRC( 0 1 0NS 200NS 0NS  20NS  23NS 190NS 200NS -1 )
MODEL  A1 LSRC( 0 0 0NS 200NS 0NS  20NS  23NS 190NS 200NS -1 )
MODEL  A2 LSRC( 0 1 0NS 200NS 0NS  20NS  23NS 190NS 200NS -1 )
MODEL  A3 LSRC( 0 0 0NS 200NS 0NS  20NS  23NS 190NS 200NS -1 )
*
MODEL  A4 LSRC( 0 0 10NS 200NS 0NS  20NS  23NS 190NS 200NS -1 )
MODEL  A5 LSRC( 0 1 10NS 200NS 0NS  20NS  23NS 190NS 200NS -1 )
MODEL  A6 LSRC( 0 1 10NS 200NS 0NS  20NS  23NS 190NS 200NS -1 )
MODEL  A7 LSRC( 0 0 10NS 200NS 0NS  20NS  23NS 190NS 200NS -1 )
*
MODEL C11 TSRC(12 0 0N 200N 0N 25N 28N 40N 43N 86N 90N 124N 127N 200N -1)
MODEL C12 TSRC( 0 12 0NS 200NS 0NS 33NS 36NS 127NS 130NS 200NS -1 )
MODEL RW  TSRC( 0 12 0NS 400NS 0NS 190NS 200NS 390NS 400NS -1)
MODEL DI  TSRC( 0  0 0NS 400NS 0NS 190NS 200NS 390NS 400NS -1 )
*
* TIMING TRANSISTOR MODELS FOR DECODER AND I/O CIRCUITS
MODEL IDRI NDRIV( 2.00 0.5 20U 0.7 0.6 0.0 5  100 )
MODEL ILCD NLOAD( 1.00 -5  20U 0.7 0.6 0.0 5  100 )
MODEL TXSN NTXG ( 1.00 0.5 20U 0.7 0.6 0.0 5  100 )
MODEL NSTG NTXG ( 1.00 0.5 20U 0.7 0.6 0.0 5  100 )
MODEL DRSN NDRIV( 1.00 0.5 20U 0.7 0.6 0.0 5  100 )
*
* CIRCUIT TRANSISTOR MODELS FOR SENSE AMPLIFIERS
MODEL TXSL NMOSE( 3.00 0.5 20U 0.7 0.6 0.0 )
MODEL TXSF NMOSE( 30.0 0.5 20U 0.7 0.6 0.0 )
MODEL TXSS NMOSE( 1.00 0.5 20U 0.7 0.6 0.0 )
MODEL DSEN NMOSE( 20.0 0.5 20U 0.7 0.6 0.0 )
*
* MODELS FOR DUMMY-SELECT DECODERS
MODEL INV  INV  ( 4NS 2NS )
MODEL NOR  NOR  ( 4NS 2NS )
*
* LOGIC-TO-VOLTAGE CONVERTER AND CAPACITOR MODELS
MODEL LTV  LTV  ( 6NS 3NS 0 5 2.5 )
MODEL    C GCAPR
*
* COLUMN DECODER
*
* ADDRESS LINES: 1-A0, 2-A1, 3-A2, 4-A3
*
* INPUT INVERTERS
I1  37  1 INV
I2  38  2 INV
I3  39  3 INV
I4  40  4 INV
*
*
* NOR GATES
N0  21  1  2  3  4 NOR
N1  22 37  2  3  4 NOR
N2  23  1 38  3  4 NOR
N3  24 37 38  3  4 NOR
N4  25  1  2 39  4 NOR
N5  26 37  2 39  4 NOR
N6  27  1 38 39  4 NOR
N7  28 37 38 39  4 NOR
N8  29  1  2  3 40 NOR
N9  30 37  2  3 40 NOR
```

```
N10 31  1 38  3 40 NOR
N11 32 37 38  3 40 NOR
N12 33  1  2 39 40 NOR
N13 34 37  2 39 40 NOR
N14 35  1 38 39 40 NOR
N15 36 37 38 39 40 NOR
*
* LOGIC-TO-VOLTAGE CONVERTERS
V1  41 21 LTV
V2  42 22 LTV
V3  43 23 LTV
V4  44 24 LTV
V5  45 25 LTV
V6  46 26 LTV
V7  47 27 LTV
V8  48 28 LTV
V9  49 29 LTV
V10 50 30 LTV
V11 51 31 LTV
V12 52 32 LTV
V13 53 33 LTV
V14 54 34 LTV
V15 55 35 LTV
V16 56 36 LTV
*
* Y-AXIS ADDRESS DECODER AND I/O SELECT
*
* INPUT ADDRESS LINES: 101-A4, 102-A5, 103-A6, 104-A7
* DATA IN: 161, DATA OUT: 162, R/W SELECT: 163
*
*
* INPUT INVERTERS
I1  137 101 INV
I2  138 102 INV
I3  139 103 INV
I4  140 104 INV
*
* OUTPUT INVERTERS
I5  1000 141 IDRI
I6  1100 142 IDRI
I7  1200 143 IDRI
I8  1300 144 IDRI
I9  1400 145 IDRI
I10 1500 146 IDRI
I11 1600 147 IDRI
I12 1700 148 IDRI
I13 1800 149 IDRI
I14 1900 150 IDRI
I15 2000 151 IDRI
I16 2100 152 IDRI
I17 2200 153 IDRI
I18 2300 154 IDRI
I19 2400 155 IDRI
I20 2500 156 IDRI
L5  1000     ILOD
L6  1100     ILOD
L7  1200     ILOD
L8  1300     ILOD
L9  1400     ILOD
L10 1500    ILOD
L11 1600    ILOD
L12 1700    ILOD
```

```
L13 1800    ILOD
L14 1900    ILOD
L15 2000    ILOD
L16 2100    ILOD
L17 2200    ILOD
L18 2300    ILOD
L19 2400    ILOD
L20 2500    ILOD
*
* NOR GATES
H0   121 101 102 103 104 NOR
H1   122 137 102 103 104 NOR
H2   123 101 138 103 104 NOR
H3   124 137 138 103 104 NOR
H4   125 101 102 139 104 NOR
H5   126 137 102 139 104 NOR
H6   127 101 138 139 104 NOR
H7   128 137 138 139 104 NOR
H8   129 101 102 103 140 NOR
H9   130 137 102 103 140 NOR
H10  131 101 138 103 140 NOR
H11  132 137 138 103 140 NOR
H12  133 101 102 139 140 NOR
H13  134 137 102 139 140 NOR
H14  135 101 138 139 140 NOR
H15  136 137 138 139 140 NOR
*
* LOGIC-TO-VOLTAGE CONVERTERS
V1   141 121 LTV
V2   142 122 LTV
V3   143 123 LTV
V4   144 124 LTV
V5   145 125 LTV
V6   146 126 LTV
V7   147 127 LTV
V8   148 128 LTV
V9   149 129 LTV
V10  150 130 LTV
V11  151 131 LTV
V12  152 132 LTV
V13  153 133 LTV
V14  154 134 LTV
V15  155 135 LTV
V16  156 136 LTV
*
* I/O ROW-SELECT TRANSMISSION GATES
T1   1001 160 141 HSTG
T2   1101 160 142 HSTG
T3   1201 160 143 HSTG
T4   1301 160 144 HSTG
T5   1401 160 145 HSTG
T6   1501 160 146 HSTG
T7   1601 160 147 HSTG
T8   1701 160 148 HSTG
T9   1801 160 149 HSTG
T10  1901 160 150 HSTG
T11  2001 160 151 HSTG
T12  2101 160 152 HSTG
T13  2201 160 153 HSTG
T14  2301 160 154 HSTG
T15  2401 160 155 HSTG
T16  2501 160 156 HSTG.
```

```
*
* DATA READ/WRITE CIRCUITRY
TDI 161 160 163 NSTG
TDO 160 162 164 NSTG
IRW 164 163     IDRI
LRW 164         ILOD
C160 160 C 0.08P
C162 162 C 0.08P
C164 164 C 0.08P
*
* DATA  IN: NODE 161, TIMING SOURCE
* DATA OUT: NODE 162, VOLTAGE
* R/W SELT: NODE 163, TIMING SOURCE ( 12-WRITE, 0-READ )
*
*
* NODE CAPACITANCES
C5  1000 C 0.08P
C6  1100 C 0.08P
C7  1200 C 0.08P
C8  1300 C 0.08P
C9  1400 C 0.08P
C10 1500 C 0.08P
C11 1600 C 0.08P
C12 1700 C 0.08P
C13 1800 C 0.08P
C14 1900 C 0.08P
C15 2000 C 0.08P
C16 2100 C 0.08P
C17 2200 C 0.08P
C18 2300 C 0.08P
C19 2400 C 0.08P
C20 2500 C 0.08P
*
*
* SENSE AMP AND STORAGE DEVICES, ROW 1000
*
* LEFT  BIT LINE: 1001
* RIGHT BIT LINE: 1002
* STORAGE  NODES: 1041 - 1048 (LEFT), 1049 - 1056
* ROW SELECT    : 1000
* VDD           :   12
*
* SENSE AMP
TL1   10 1001   11 TXSL
TL2   10 1002   11 TXSL
TS0 1001 1002 1000 TXSS
TF1 1001 1003 1002 TXSF
TF2 1002 1003 1001 TXSF
TT0 1003   12      DSEN
C01 1001 C 0.8P
C02 1002 C 0.8P
C03 1003 C 0.01P
*
* DUMMY CELLS
C01 1001 1004    4 TXSN
C02 1002 1040   40 TXSN
CP1 1004 1000      DRSN
CP2 1040 1000      DRSN
CD1 1004 C 0.01P
CD2 1040 C 0.01P
*
* STORAGE CELLS
```

```
C41 1001 1041   41 TXSH
C42 1001 1042   42 TXSH
C43 1001 1043   43 TXSH
C44 1001 1044   44 TXSH
C45 1001 1045   45 TXSH
C46 1001 1046   46 TXSH
C47 1001 1047   47 TXSH
C48 1001 1048   48 TXSH
C49 1002 1049   49 TXSH
C50 1002 1050   50 TXSH
C51 1002 1051   51 TXSH
C52 1002 1052   52 TXSH
C53 1002 1053   53 TXSH
C54 1002 1054   54 TXSH
C55 1002 1055   55 TXSH
C56 1002 1056   56 TXSH
S41 1041 C 0.06P
S42 1042 C 0.06P
S43 1043 C 0.06P
S44 1044 C 0.06P
S45 1045 C 0.06P
S46 1046 C 0.06P
S47 1047 C 0.06P
S48 1048 C 0.06P
S49 1049 C 0.06P
S50 1050 C 0.06P
S51 1051 C 0.06P
S52 1052 C 0.06P
S53 1053 C 0.06P
S54 1054 C 0.06P
S55 1055 C 0.06P
S56 1056 C 0.06P
*
* SENSE AMP AND STORAGE DEVICES, ROW 1100
*
* LEFT  BIT LINE: 1101
* RIGHT BIT LINE: 1102
* STORAGE  NODES: 1141 - 1148 (LEFT), 1149 - 1156
* ROW SELECT     : 1100
* VDD            :   12
*
* SENSE AMP
TL1   10 1101   11 TXSL
TL2   10 1102   11 TXSL
TS0 1101 1102 1100 TXSS
TF1 1101 1103 1102 TXSF
TF2 1102 1103 1101 TXSF
TT0 1103   12      DSEN
C01 1101 C 0.8P
C02 1102 C 0.8P
C03 1103 C 0.01P
*
* DUMMY CELLS
C01 1101 1104    4 TXSH
C02 1102 1140   40 TXSH
CP1 1104 1100      DRSH
CP2 1140 1100      DRSH
CD1 1104 C 0.01P
CD2 1140 C 0.01P
*
* STORAGE CELLS
C41 1101 1141   41 TXSH
```

```
C42 1101 1142   42 TXSH
C43 1101 1143   43 TXSH
C44 1101 1144   44 TXSH
C45 1101 1145   45 TXSH
C46 1101 1146   46 TXSH
C47 1101 1147   47 TXSH
C48 1101 1148   48 TXSH
C49 1102 1149   49 TXSH
C50 1102 1150   50 TXSH
C51 1102 1151   51 TXSH
C52 1102 1152   52 TXSH
C53 1102 1153   53 TXSH
C54 1102 1154   54 TXSH
C55 1102 1155   55 TXSH
C56 1102 1156   56 TXSH
S41 1141 C 0.06P
S42 1142 C 0.06P
S43 1143 C 0.06P
S44 1144 C 0.06P
S45 1145 C 0.06P
S46 1146 C 0.06P
S47 1147 C 0.06P
S48 1148 C 0.06P
S49 1149 C 0.06P
S50 1150 C 0.06P
S51 1151 C 0.06P
S52 1152 C 0.06P
S53 1153 C 0.06P
S54 1154 C 0.06P
S55 1155 C 0.06P
S56 1156 C 0.06P
*
* SENSE AMP AND STORAGE DEVICES, ROW 1200
*
* LEFT  BIT LINE: 1201
* RIGHT BIT LINE: 1202
* STORAGE  NODES: 1241 - 1248 (LEFT), 1249 - 1256
* ROW SELECT    : 1200
* VDD           :   12
*
* SENSE AMP
TL1    10 1201   11 TXSL
TL2    10 1202   11 TXSL
TS0 1201 1202 1200 TXSS
TF1 1201 1203 1202 TXSF
TF2 1202 1203 1201 TXSF
TT0 1203   12      DSEN
C01 1201 C 0.8P
C02 1202 C 0.8P
C03 1203 C 0.01P
*
* DUMMY CELLS
C01 1201 1204    4 TXSH
C02 1202 1240   40 TXSH
CP1 1204 1200      DRSH
CP2 1240 1200      DRSH
CD1 1204 C 0.01P
CD2 1240 C 0.01P
*
* STORAGE CELLS
C41 1201 1241   41 TXSH
C42 1201 1242   42 TXSH
```

```
C43 1201 1243   43 TXSH
C44 1201 1244   44 TXSH
C45 1201 1245   45 TXSH
C46 1201 1246   46 TXSH
C47 1201 1247   47 TXSH
C48 1201 1248   48 TXSH
C49 1202 1249   49 TXSH
C50 1202 1250   50 TXSH
C51 1202 1251   51 TXSH
C52 1202 1252   52 TXSH
C53 1202 1253   53 TXSH
C54 1202 1254   54 TXSH
C55 1202 1255   55 TXSH
C56 1202 1256   56 TXSH
S41 1241 C 0.06P
S42 1242 C 0.06P
S43 1243 C 0.06P
S44 1244 C 0.06P
S45 1245 C 0.06P
S46 1246 C 0.06P
S47 1247 C 0.06P
S48 1248 C 0.06P
S49 1249 C 0.06P
S50 1250 C 0.06P
S51 1251 C 0.06P
S52 1252 C 0.06P
S53 1253 C 0.06P
S54 1254 C 0.06P
S55 1255 C 0.06P
S56 1256 C 0.06P
*
* SENSE AMP AND STORAGE DEVICES, ROW 1300
*
* LEFT  BIT LINE: 1301
* RIGHT BIT LINE: 1302
* STORAGE  NODES: 1341 - 1348 (LEFT), 1349 - 1356
* ROW SELECT    : 1300
* VDD           :   12
*
* SENSE AMP
TL1   10 1301   11 TXSL
TL2   10 1302   11 TXSL
TS0 1301 1302 1300 TXSS
TF1 1301 1303 1302 TXSF
TF2 1302 1303 1301 TXSF
TT0 1303   12      DSEH
C01 1301 C 0.8P
C02 1302 C 0.8P
C03 1303 C 0.01P
*
* DUMMY CELLS
C01 1301 1304    4 TXSH
C02 1302 1340   40 TXSH
CP1 1304 1300      DRSH
CP2 1340 1300      DRSH
CD1 1304 C 0.01P
CD2 1340 C 0.01P
*
* STORAGE CELLS
C41 1301 1341   41 TXSH
C42 1301 1342   42 TXSH
C43 1301 1343   43 TXSH
```

```
C44 1301 1344    44 TXSN
C45 1301 1345    45 TXSN
C46 1301 1346    46 TXSN
C47 1301 1347    47 TXSN
C48 1301 1348    48 TXSN
C49 1302 1349    49 TXSN
C50 1302 1350    50 TXSN
C51 1302 1351    51 TXSN
C52 1302 1352    52 TXSN
C53 1302 1353    53 TXSN
C54 1302 1354    54 TXSN
C55 1302 1355    55 TXSN
C56 1302 1356    56 TXSN
S41 1341 C 0.06P
S42 1342 C 0.06P
S43 1343 C 0.06P
S44 1344 C 0.06P
S45 1345 C 0.06P
S46 1346 C 0.06P
S47 1347 C 0.06P
S48 1348 C 0.06P
S49 1349 C 0.06P
S50 1350 C 0.06P
S51 1351 C 0.06P
S52 1352 C 0.06P
S53 1353 C 0.06P
S54 1354 C 0.06P
S55 1355 C 0.06P
S56 1356 C 0.06P
*
* SENSE AMP AND STORAGE DEVICES, ROW 1400
*
* LEFT  BIT LINE: 1401
* RIGHT BIT LINE: 1402
* STORAGE  NODES: 1441 - 1448 (LEFT), 1449 - 1456
* ROW SELECT    : 1400
* VDD           :   12
*
* SENSE AMP
TL1   10 1401   11 TXSL
TL2   10 1402   11 TXSL
TS0 1401 1402 1400 TXSS
TF1 1401 1403 1402 TXSF
TF2 1402 1403 1401 TXSF
TT0 1403   12      DSEN
C01 1401 C 0.8P
C02 1402 C 0.8P
C03 1403 C 0.01P
*
* DUMMY CELLS
C01 1401 1404    4 TXSN
C02 1402 1440   40 TXSN
CP1 1404 1400      DRSN
CP2 1440 1400      DRSN
CD1 1404 C 0.01P
CD2 1440 C 0.01P
*
* STORAGE CELLS
C41 1401 1441   41 TXSN
C42 1401 1442   42 TXSN
C43 1401 1443   43 TXSN
C44 1401 1444   44 TXSN
```

```
C45 1401 1445   45 TXSH
C46 1401 1446   46 TXSH
C47 1401 1447   47 TXSH
C48 1401 1448   48 TXSH
C49 1402 1449   49 TXSH
C50 1402 1450   50 TXSH
C51 1402 1451   51 TXSH
C52 1402 1452   52 TXSH
C53 1402 1453   53 TXSH
C54 1402 1454   54 TXSH
C55 1402 1455   55 TXSH
C56 1402 1456   56 TXSH
S41 1441 C 0.06P
S42 1442 C 0.06P
S43 1443 C 0.06P
S44 1444 C 0.06P
S45 1445 C 0.06P
S46 1446 C 0.06P
S47 1447 C 0.06P
S48 1448 C 0.06P
S49 1449 C 0.06P
S50 1450 C 0.06P
S51 1451 C 0.06P
S52 1452 C 0.06P
S53 1453 C 0.06P
S54 1454 C 0.06P
S55 1455 C 0.06P
S56 1456 C 0.06P
*
* SENSE AMP AND STORAGE DEVICES, ROW 1500
*
* LEFT  BIT LINE: 1501
* RIGHT BIT LINE: 1502
* STORAGE  NODES: 1541 - 1548 (LEFT), 1549 - 1556
* ROW SELECT    : 1500
* VDD           :   12
*
* SENSE AMP
TL1   10 1501   11 TXSL
TL2   10 1502   11 TXSL
TS0 1501 1502 1500 TXSS
TF1 1501 1503 1502 TXSF
TF2 1502 1503 1501 TXSF
TT0 1503   12      DSEN
C01 1501 C 0.9P
C02 1502 C 0.8P
C03 1503 C 0.01P
*
* DUMMY CELLS
C01 1501 1504    4 TXSH
C02 1502 1540   40 TXSH
CP1 1504 1500      DRSH
CP2 1540 1500      DRSH
CD1 1504 C 0.01P
CD2 1540 C 0.01P
*
* STORAGE CELLS
C41 1501 1541   41 TXSH
C42 1501 1542   42 TXSH
C43 1501 1543   43 TXSH
C44 1501 1544   44 TXSH
C45 1501 1545   45 TXSH
```

```
C46 1501 1546   46 TXSH
C47 1501 1547   47 TXSH
C48 1501 1548   48 TXSH
C49 1502 1549   49 TXSH
C50 1502 1550   50 TXSH
C51 1502 1551   51 TXSH
C52 1502 1552   52 TXSH
C53 1502 1553   53 TXSH
C54 1502 1554   54 TXSH
C55 1502 1555   55 TXSH
C56 1502 1556   56 TXSH
S41 1541 C 0.06P
S42 1542 C 0.06P
S43 1543 C 0.06P
S44 1544 C 0.06P
S45 1545 C 0.06P
S46 1546 C 0.06P
S47 1547 C 0.06P
S48 1548 C 0.06P
S49 1549 C 0.06P
S50 1550 C 0.06P
S51 1551 C 0.06P
S52 1552 C 0.06P
S53 1553 C 0.06P
S54 1554 C 0.06P
S55 1555 C 0.06P
S56 1556 C 0.06P
*
* SENSE AMP AND STORAGE DEVICES, ROW 1600
*
* LEFT  BIT LINE: 1601
* RIGHT BIT LINE: 1602
* STORAGE  NODES: 1641 - 1648 (LEFT), 1649 - 1656
* ROW SELECT     : 1600
* VDD            :   12
*
* SENSE AMP
TL1   10 1601   11 TXSL
TL2   10 1602   11 TXSL
TS0 1601 1602 1600 TXSS
TF1 1601 1603 1602 TXSF
TF2 1602 1603 1601 TXSF
TT0 1603   12      DSEH
C01 1601 C 0.8P
C02 1602 C 0.8P
C03 1603 C 0.01P
*
* DUMMY CELLS
C01 1601 1604    4 TXSH
C02 1602 1640   40 TXSH
CP1 1604 1600      DRSH
CP2 1640 1600      DRSH
CD1 1604 C 0.01P
CD2 1640 C 0.01P
*
* STORAGE CELLS
C41 1601 1641   41 TXSH
C42 1601 1642   42 TXSH
C43 1601 1643   43 TXSH
C44 1601 1644   44 TXSH
C45 1601 1645   45 TXSH
C46 1601 1646   46 TXSH
```

```
C47 1601 1647    47 TXSH
C48 1601 1648    48 TXSH
C49 1602 1649    49 TXSH
C50 1602 1650    50 TXSH
C51 1602 1651    51 TXSH
C52 1602 1652    52 TXSH
C53 1602 1653    53 TXSH
C54 1602 1654    54 TXSH
C55 1602 1655    55 TXSH
C56 1602 1656    56 TXSH
S41 1641 C 0.06P
S42 1642 C 0.06P
S43 1643 C 0.06P
S44 1644 C 0.06P
S45 1645 C 0.06P
S46 1646 C 0.06P
S47 1647 C 0.06P
S48 1648 C 0.06P
S49 1649 C 0.06P
S50 1650 C 0.06P
S51 1651 C 0.06P
S52 1652 C 0.06P
S53 1653 C 0.06P
S54 1654 C 0.06P
S55 1655 C 0.06P
S56 1656 C 0.06P
*
* SENSE AMP AND STORAGE DEVICES, ROW 1700
*
* LEFT  BIT LINE: 1701
* RIGHT BIT LINE: 1702
* STORAGE  NODES: 1741 - 1748 (LEFT), 1749 - 1756
* ROW SELECT    : 1700
* VDD           :   12
*
* SENSE AMP
TL1    10 1701    11 TXSL
TL2    10 1702    11 TXSL
TS0 1701 1702 1700 TXSS
TF1 1701 1703 1702 TXSF
TF2 1702 1703 1701 TXSF
TT0 1703   12      DSEN
C01 1701 C 0.8P
C02 1702 C 0.8P
C03 1703 C 0.01P
*
* DUMMY CELLS
C01 1701 1704     4 TXSH
C02 1702 1740    40 TXSH
CP1 1704 1700       DRSH
CP2 1740 1700       DRSH
CD1 1704 C 0.01P
CD2 1740 C 0.01P
*
* STORAGE CELLS
C41 1701 1741    41 TXSH
C42 1701 1742    42 TXSH
C43 1701 1743    43 TXSH
C44 1701 1744    44 TXSH
C45 1701 1745    45 TXSH
C46 1701 1746    46 TXSH
C47 1701 1747    47 TXSH
```

```
C48 1701 1748   48 TXSH
C49 1702 1749   49 TXSH
C50 1702 1750   50 TXSH
C51 1702 1751   51 TXSH
C52 1702 1752   52 TXSH
C53 1702 1753   53 TXSH
C54 1702 1754   54 TXSH
C55 1702 1755   55 TXSH
C56 1702 1756   56 TXSH
S41 1741 C 0.06P
S42 1742 C 0.06P
S43 1743 C 0.06P
S44 1744 C 0.06P
S45 1745 C 0.06P
S46 1746 C 0.06P
S47 1747 C 0.06P
S48 1748 C 0.06P
S49 1749 C 0.06P
S50 1750 C 0.06P
S51 1751 C 0.06P
S52 1752 C 0.06P
S53 1753 C 0.06P
S54 1754 C 0.06P
S55 1755 C 0.06P
S56 1756 C 0.06P
*
* SENSE AMP AND STORAGE DEVICES, ROW 1800
*
* LEFT  BIT LINE: 1801
* RIGHT BIT LINE: 1802
* STORAGE  NODES: 1841 - 1848 (LEFT), 1849 - 1856
* ROW SELECT    : 1800
* VDD           :   12
*
* SENSE AMP
TL1   10 1801   11 TXSL
TL2   10 1802   11 TXSL
TS0 1801 1902 1800 TXSS
TF1 1801 1803 1802 TXSF
TF2 1802 1903 1801 TXSF
TT0 1803   12      DSEN
C01 1801 C 0.8P
C02 1802 C 0.8P
C03 1903 C 0.01P
*
* DUMMY CELLS
C01 1801 1804    4 TXSH
C02 1802 1840   40 TXSH
CP1 1804 1800      DRSH
CP2 1840 1900      DRSH
CD1 1804 C 0.01P
CD2 1840 C 0.01P
*
* STORAGE CELLS
C41 1801 1841   41 TXSH
C42 1801 1842   42 TXSH
C43 1901 1943   43 TXSH
C44 1801 1844   44 TXSH
C45 1801 1845   45 TXSH
C46 1801 1846   46 TXSH
C47 1801 1847   47 TXSH
C48 1801 1848   48 TXSH
```

```
C49 1802 1849    49 TXSH
C50 1802 1850    50 TXSH
C51 1802 1851    51 TXSH
C52 1802 1852    52 TXSH
C53 1902 1953    53 TXSH
C54 1802 1854    54 TXSH
C55 1802 1855    55 TXSH
C56 1902 1856    56 TXSH
S41 1941 C 0.06P
S42 1942 C 0.06P
S43 1843 C 0.06P
S44 1844 C 0.06P
S45 1845 C 0.06P
S46 1846 C 0.06P
S47 1847 C 0.06P
S48 1848 C 0.06P
S49 1849 C 0.06P
S50 1850 C 0.06P
S51 1851 C 0.06P
S52 1852 C 0.06P
S53 1853 C 0.06P
S54 1854 C 0.06P
S55 1855 C 0.06P
S56 1856 C 0.06P
*
* SENSE AMP AND STORAGE DEVICES, ROW 1900
*
* LEFT  BIT LINE: 1901
* RIGHT BIT LINE: 1902
* STORAGE  NODES: 1941 - 1948 (LEFT), 1949 - 1956
* ROW SELECT    : 1900
* VDD           :   12
*
* SENSE AMP
TL1    10 1901    11 TXSL
TL2    10 1902    11 TXSL
TS0 1901 1902 1900 TXSS
TF1 1901 1903 1902 TXSF
TF2 1902 1903 1901 TXSF
TT0 1903   12      DSEH
C01 1901 C 0.8P
C02 1902 C 0.8P
C03 1903 C 0.01P
*
* DUMMY CELLS
C01. 1901 1904     4 TXSH
C02 1902 1940    40 TXSH
CP1 1904 1900       DRSH
CP2 1940 1900       DRSH
CD1 1904 C 0.01P
CD2 1940 C 0.01P
*
* STORAGE CELLS
C41 1901 1941    41 TXSH
C42 1901 1942    42 TXSH
C43 1901 1943    43 TXSH
C44 1901 1944    44 TXSH
C45 1901 1945    45 TXSH
C46 1901 1946    46 TXSH
C47 1901 1947    47 TXSH
C48 1901 1948    48 TXSH
C49 1902 1949    49 TXSH
```

```
C50 1902 1950   50 TXSN
C51 1902 1951   51 TXSN
C52 1902 1952   52 TXSN
C53 1902 1953   53 TXSN
C54 1902 1954   54 TXSN
C55 1902 1955   55 TXSN
C56 1902 1956   56 TXSN
S41 1941 C 0.06P
S42 1942 C 0.06P
S43 1943 C 0.06P
S44 1944 C 0.06P
S45 1945 C 0.06P
S46 1946 C 0.06P
S47 1947 C 0.06P
S48 1948 C 0.06P
S49 1949 C 0.06P
S50 1950 C 0.06P
S51 1951 C 0.06P
S52 1952 C 0.06P
S53 1953 C 0.06P
S54 1954 C 0.06P
S55 1955 C 0.06P
S56 1956 C 0.06P
*
* SENSE AMP AND STORAGE DEVICES, ROW 2000
*
* LEFT  BIT LINE: 2001
* RIGHT BIT LINE: 2002
* STORAGE  NODES: 2041 - 2048 (LEFT), 2049 - 2056
* ROW SELECT    : 2000
* VDD           :   12
*
* SENSE AMP
TL1   10 2001   11 TXSL
TL2   10 2002   11 TXSL
TS0 2001 2002 2000 TXSS
TF1 2001 2003 2002 TXSF
TF2 2002 2003 2001 TXSF
TT0 2003   12      DSEN
C01 2001 C 0.8P
C02 2002 C 0.8P
C03 2003 C 0.01P
*
* DUMMY CELLS
C01 2001 2004    4 TXSN
C02 2002 2040   40 TXSN
CP1 2004 2000      DRSN
CP2 2040 2000      DRSN
CD1 2004 C 0.01P
CD2 2040 C 0.01P
*
* STORAGE CELLS
C41 2001 2041   41 TXSN
C42 2001 2042   42 TXSN
C43 2001 2043   43 TXSN
C44 2001 2044   44 TXSN
C45 2001 2045   45 TXSN
C46 2001 2046   46 TXSN
C47 2001 2047   47 TXSN
C48 2001 2048   48 TXSN
C49 2002 2049   49 TXSN
C50 2002 2050   50 TXSN
```

```
C51 2002 2051   51 TXSN
C52 2002 2052   52 TXSN
C53 2002 2053   53 TXSN
C54 2002 2054   54 TXSN
C55 2002 2055   55 TXSN
C56 2002 2056   56 TXSN
S41 2041 C 0.06P
S42 2042 C 0.06P
S43 2043 C 0.06P
S44 2044 C 0.06P
S45 2045 C 0.06P
S46 2046 C 0.06P
S47 2047 C 0.06P
S48 2048 C 0.06P
S49 2049 C 0.06P
S50 2050 C 0.06P
S51 2051 C 0.06P
S52 2052 C 0.06P
S53 2053 C 0.06P
S54 2054 C 0.06P
S55 2055 C 0.06P
S56 2056 C 0.06P
*
* SENSE AMP AND STORAGE DEVICES, ROW 2100
*
* LEFT  BIT LINE: 2101
* RIGHT BIT LINE: 2102
* STORAGE  NODES: 2141 - 2148 (LEFT), 2149 - 2156
* ROW SELECT    : 2100
* VDD           :   12
*
* SENSE AMP
TL1   10 2101   11 TXSL
TL2   10 2102   11 TXSL
TS0 2101 2102 2100 TXSS
TF1 2101 2103 2102 TXSF
TF2 2102 2103 2101 TXSF
TT0 2103   12      DSEN
C01 2101 C 0.8P
C02 2102 C 0.8P
C03 2103 C 0.01P
*
* DUMMY CELLS
C01 2101 2104    4 TXSN
C02 2102 2140   40 TXSN
CP1 2104 2100      DRSN
CP2 2140 2100      DRSN
CD1 2104 C 0.01P
CD2 2140 C 0.01P
*
* STORAGE CELLS
C41 2101 2141   41 TXSN
C42 2101 2142   42 TXSN
C43 2101 2143   43 TXSN
C44 2101 2144   44 TXSN
C45 2101 2145   45 TXSN
C46 2101 2146   46 TXSN
C47 2101 2147   47 TXSN
C48 2101 2148   48 TXSN
C49 2102 2149   49 TXSN
C50 2102 2150   50 TXSN
C51 2102 2151   51 TXSN
```

```
C52 2102 2152   52 TXSN
C53 2102 2153   53 TXSN
C54 2102 2154   54 TXSN
C55 2102 2155   55 TXSN
C56 2102 2156   56 TXSN
S41 2141 C 0.06P
S42 2142 C 0.06P
S43 2143 C 0.06P
S44 2144 C 0.06P
S45 2145 C 0.06P
S46 2146 C 0.06P
S47 2147 C 0.06P
S48 2148 C 0.06P
S49 2149 C 0.06P
S50 2150 C 0.06P
S51 2151 C 0.06P
S52 2152 C 0.06P
S53 2153 C 0.06P
S54 2154 C 0.06P
S55 2155 C 0.06P
S56 2156 C 0.06P
*
* SENSE AMP AND STORAGE DEVICES, ROW 2200
*
* LEFT  BIT LINE: 2201
* RIGHT BIT LINE: 2202
* STORAGE  NODES: 2241 - 2248 (LEFT), 2249 - 2256
* ROW SELECT    : 2200
* VDD           :   12
*
* SENSE AMP
TL1   10 2201   11 TXSL
TL2   10 2202   11 TXSL
TS0 2201 2202 2200 TXSS
TF1 2201 2203 2202 TXSF
TF2 2202 2203 2201 TXSF
TT0 2203   12      DSEN
C01 2201 C 0.8P
C02 2202 C 0.8P
C03 2203 C 0.01P
*
* DUMMY CELLS
C01 2201 2204    4 TXSN
C02 2202 2240   40 TXSN
CP1 2204 2200      DRSN
CP2 2240 2200      DRSN
CD1 2204 C 0.01P
CD2 2240 C 0.01P
*
* STORAGE CELLS
C41 2201 2241   41 TXSN
C42 2201 2242   42 TXSN
C43 2201 2243   43 TXSN
C44 2201 2244   44 TXSN
C45 2201 2245   45 TXSN
C46 2201 2246   46 TXSN
C47 2201 2247   47 TXSN
C48 2201 2248   48 TXSN
C49 2202 2249   49 TXSN
C50 2202 2250   50 TXSN
C51 2202 2251   51 TXSN
C52 2202 2252   52 TXSN
```

```
C53 2202 2253   53 TXSN
C54 2202 2254   54 TXSN
C55 2202 2255   55 TXSN
C56 2202 2256   56 TXSN
S41 2241 C 0.06P
S42 2242 C 0.06P
S43 2243 C 0.06P
S44 2244 C 0.06P
S45 2245 C 0.06P
S46 2246 C 0.06P
S47 2247 C 0.06P
S48 2248 C 0.06P
S49 2249 C 0.06P
S50 2250 C 0.06P
S51 2251 C 0.06P
S52 2252 C 0.06P
S53 2253 C 0.06P
S54 2254 C 0.06P
S55 2255 C 0.06P
S56 2256 C 0.06P
*
* SENSE AMP AND STORAGE DEVICES, ROW 2300
*
* LEFT  BIT LINE: 2301
* RIGHT BIT LINE: 2302
* STORAGE  NODES: 2341 - 2348 (LEFT), 2349 - 2356
* ROW SELECT    : 2300
* VDD           :   12
*
* SENSE AMP
TL1   10 2301   11 TXSL
TL2   10 2302   11 TXSL
TS0 2301 2302 2300 TXSS
TF1 2301 2303 2302 TXSF
TF2 2302 2303 2301 TXSF
TT0 2303   12      DSEN
C01 2301 C 0.8P
C02 2302 C 0.8P
C03 2303 C 0.01P
*
* DUMMY CELLS
C01 2301 2304    4 TXSN
C02 2302 2340   40 TXSN
CP1 2304 2300      DRSN
CP2 2340 2300      DRSN
CD1 2304 C 0.01P
CD2 2340 C 0.01P
*
* STORAGE CELLS
C41 2301 2341   41 TXSN
C42 2301 2342   42 TXSN
C43 2301 2343   43 TXSN
C44 2301 2344   44 TXSN
C45 2301 2345   45 TXSN
C46 2301 2346   46 TXSN
C47 2301 2347   47 TXSN
C48 2301 2348   48 TXSN
C49 2302 2349   49 TXSN
C50 2302 2350   50 TXSN
C51 2302 2351   51 TXSN
C52 2302 2352   52 TXSN
C53 2302 2353   53 TXSN
```

```
C54 2302 2354   54 TXSH
C55 2302 2355   55 TXSH
C56 2302 2356   56 TXSH
S41 2341 C 0.06P
S42 2342 C 0.06P
S43 2343 C 0.06P
S44 2344 C 0.06P
S45 2345 C 0.06P
S46 2346 C 0.06P
S47 2347 C 0.06P
S48 2348 C 0.06P
S49 2349 C 0.06P
S50 2350 C 0.06P
S51 2351 C 0.06P
S52 2352 C 0.06P
S53 2353 C 0.06P
S54 2354 C 0.06P
S55 2355 C 0.06P
S56 2356 C 0.06P
*
* SENSE AMP AND STORAGE DEVICES, ROW 2400
*
* LEFT  BIT LINE: 2401
* RIGHT BIT LINE: 2402
* STORAGE  NODES: 2441 - 2448 (LEFT), 2449 - 2456
* ROW SELECT    : 2400
* VDD           :   12
*
* SENSE AMP
TL1   10 2401   11 TXSL
TL2   10 2402   11 TXSL
TS0 2401 2402 2400 TXSS
TF1 2401 2403 2402 TXSF
TF2 2402 2403 2401 TXSF
TT0 2403   12      DSEN
C01 2401 C 0.8P
C02 2402 C 0.8P
C03 2403 C 0.01P
*
* DUMMY CELLS
C01 2401 2404    4 TXSH
C02 2402 2440   40 TXSH
CP1 2404 2400      DRSH
CP2 2440 2400      DRSH
CD1 2404 C 0.01P
CD2 2440 C 0.01P
*
* STORAGE CELLS
C41 2401 2441   41 TXSH
C42 2401 2442   42 TXSH
C43 2401 2443   43 TXSH
C44 2401 2444   44 TXSH
C45 2401 2445   45 TXSH
C46 2401 2446   46 TXSH
C47 2401 2447   47 TXSH
C48 2401 2448   48 TXSH
C49 2402 2449   49 TXSH
C50 2402 2450   50 TXSH
C51 2402 2451   51 TXSH
C52 2402 2452   52 TXSH
C53 2402 2453   53 TXSH
C54 2402 2454   54 TXSH
```

```
C55 2402 2455   55 TXSH
C56 2402 2456   56 TXSH
S41 2441 C 0.06P
S42 2442 C 0.06P
S43 2443 C 0.06P
S44 2444 C 0.06P
S45 2445 C 0.06P
S46 2446 C 0.06P
S47 2447 C 0.06P
S48 2448 C 0.06P
S49 2449 C 0.06P
S50 2450 C 0.06P
S51 2451 C 0.06P
S52 2452 C 0.06P
S53 2453 C 0.06P
S54 2454 C 0.06P
S55 2455 C 0.06P
S56 2456 C 0.06P
*
* SENSE AMP AND STORAGE DEVICES, ROW 2500
*
* LEFT  BIT LINE: 2501
* RIGHT BIT LINE: 2502
* STORAGE  NODES: 2541 - 2548 (LEFT), 2549 - 2556
* ROW SELECT    : 2500
* VDD           :   12
*  .
* SENSE AMP
TL1   10 2501   11 TXSL
TL2   10 2502   11 TXSL
TS0 2501 2502 2500 TXSS
TF1 2501 2503 2502 TXSF
TF2 2502 2503 2501 TXSF
TT0 2503   12      DSEN
C01 2501 C 0.8P
C02 2502 C 0.8P
C03 2503 C 0.01P
*
* DUMMY CELLS
C01 2501 2504    4 TXSH
C02 2502 2540   40 TXSH
CP1 2504 2500      DRSH
CP2 2540 2500      DRSH
CD1 2504 C 0.01P
CD2 2540 C 0.01P
*
* STORAGE CELLS
C41 2501 2541   41 TXSH
C42 2501 2542   42 TXSH
C43 2501 2543   43 TXSH
C44 2501 2544   44 TXSH
C45 2501 2545   45 TXSH
C46 2501 2546   46 TXSH
C47 2501 2547   47 TXSH
C48 2501 2548   48 TXSH
C49 2502 2549   49 TXSH
C50 2502 2550   50 TXSH
C51 2502 2551   51 TXSH
C52 2502 2552   52 TXSH
C53 2502 2553   53 TXSH
C54 2502 2554   54 TXSH
C55 2502 2555   55 TXSH
```

```
C56 2502 2556   56 TXSN
S41 2541 C 0.06P
S42 2542 C 0.06P
S43 2543 C 0.06P
S44 2544 C 0.06P
S45 2545 C 0.06P
S46 2546 C 0.06P
S47 2547 C 0.06P
S48 2548 C 0.06P
S49 2549 C 0.06P
S50 2550 C 0.06P
S51 2551 C 0.06P
S52 2552 C 0.06P
S53 2553 C 0.06P
S54 2554 C 0.06P
S55 2555 C 0.06P
S56 2556 C 0.06P
* ANALYSIS REQUESTS
OPTS 1  15.0
TOPTS 0.1   0.05   1000 0.1
TIME   1NS 300NS
PLOT 1 2 3 14 101 102 103 104
PLOT 11 12 161 162 163 164
PLOT 1600 1601 1602 1603 1646 1640 147 46 13
*
GO
*
END
```

```
SERIAL ARITHMETIC UNIT FOR DIGITAL FILTER
*
* SOURCE MODELS
MODEL VD1 TSRC(  7  7 ON 1 ON 1 -1 )
MODEL VD2 TSRC( 12 12 ON 1 ON 1 -1 )
MODEL CK1 TSRC( 12  0 110N 200N ON 105N 115N 190N 200N -1 )
MODEL CK2 TSRC( 12  0  10N 200N ON 105N 115N 190N 200N -1 )
MODEL SI1 TSRC(  0 12  30N 2.2U 0U 1.59U 1.6U 1.79U 1.8U 2.2U -1 )
MODEL SI2 TSRC(  0  7  30N 3.0U 0U 1.99U 2.0U 2.19U 2.2U 3.0U -1 )
MODEL VGN TSRC(  0  0  ON 1 ON 1 -1 )
*
*MOSFET MODELS ( TIMING )
MODEL N001 NTXG ( 1.0000  0.5 10U 0.7 0.6 0.0 12 100 )
MODEL N002 NDRIV( 4.5000  0.5 10U 0.7 0.6 0.0 12 100 )
MODEL I003 NLOAD( 0.8333 -5.3 10U 0.7 0.6 0.0 12 100 )
MODEL N004 NTXG ( 2.2500  0.5 10U 0.7 0.6 0.0 12 100 )
MODEL N005 NDRIV( 2.2500  0.5 10U 0.7 0.6 0.0 12 100 )
MODEL I006 NLOAD( 1.5000 -5.3 10U 0.7 0.6 0.0 12 100 )
MODEL I007 NTXG ( 1.5000 -5.3 10U 0.7 0.6 0.0 12 100 )
MODEL N010 NDRIV( 7.2500  0.5 10U 0.7 0.6 0.0 12 100 )
MODEL N011 NDRIV( 6.5000  0.5 10U 0.7 0.6 0.0 12 100 )
MODEL I012 NLOAD( 1.6667 -5.3 10U 0.7 0.6 0.0 12 100 )
MODEL I013 NLOAD( 0.8333 -5.3 10U 0.7 0.6 0.0  7 100 )
MODEL N014 NDRIV( 3.7500  0.5 10U 0.7 0.6 0.0 12 100 )
MODEL N015 NDRIV( 4.0000  0.5 10U 0.7 0.6 0.0 12 100 )
MODEL I016 NLOAD( 0.6250 -5.3 10U 0.7 0.6 0.0 12 100 )
MODEL N017 NDRIV( 3.0000  0.5 10U 0.7 0.6 0.0 12 100 )
MODEL I020 NLOAD( 0.4545 -5.3 10U 0.7 0.6 0.0 12 100 )
MODEL N021 NDRIV( 2.7500  0.5 10U 0.7 0.6 0.0 12 100 )
MODEL N022 NDRIV( 3.5000  0.5 10U 0.7 0.6 0.0 12 100 )
MODEL I023 NLOAD( 0.3846 -5.3 10U 0.7 0.6 0.0 12 100 )
MODEL N024 NDRIV( 1.0000  0.5 10U 0.7 0.6 0.0 12 100 )
MODEL N025 NDRIV( 2.5000  0.5 10U 0.7 0.6 0.0 12 100 )
MODEL I026 NLOAD( 0.4545 -5.3 10U 0.7 0.6 0.0  7 100 )
MODEL N027 NTXG ( 1.2500  0.5 10U 0.7 0.6 0.0 12 100 )
MODEL I030 NLOAD( 0.4167 -5.3 10U 0.7 0.6 0.0  7 100 )
MODEL N031 NDRIV( 1.2500  0.5 10U 0.7 0.6 0.0 12 100 )
MODEL I032 NLOAD( 0.6250 -5.3 10U 0.7 0.6 0.0  7 100 )
MODEL I033 NLOAD( 0.5000 -5.3 10U 0.7 0.6 0.0  7 100 )
MODEL I034 NLOAD( 0.5000 -5.3 10U 0.7 0.6 0.0 12 100 )
MODEL N035 NDRIV( 5.0000  0.5 10U 0.7 0.6 0.0 12 100 )
MODEL I036 NLOAD( 0.7143 -5.3 10U 0.7 0.6 0.0 12 100 )
MODEL I037 NTXG ( 0.8333 -5.3 10U 0.7 0.6 0.0 12 100 )
MODEL N040 NDRIV( 3.2500  0.5 10U 0.7 0.6 0.0 12 100 )
MODEL I041 NLOAD( 0.5556 -5.3 10U 0.7 0.6 0.0  7 100 )
MODEL I042 NLOAD( 0.4167 -5.3 10U 0.7 0.6 0.0 12 100 )
MODEL N043 NDRIV( 6.0000  0.5 10U 0.7 0.6 0.0 12 100 )
MODEL N044 NDRIV( 1.7500  0.5 10U 0.7 0.6 0.0 12 100 )
MODEL N045 NDRIV( 4.2500  0.5 10U 0.7 0.6 0.0 12 100 )
MODEL N046 NDRIV( 7.0000  0.5 10U 0.7 0.6 0.0 12 100 )
MODEL I047 NLOAD( 0.5556 -5.3 10U 0.7 0.6 0.0 12 100 )
MODEL I050 NLOAD( 0.2632 -5.3 10U 0.7 0.6 0.0 12 100 )
MODEL I051 NLOAD( 1.0000 -5.3 10U 0.7 0.6 0.0 12 100 )
MODEL I052 NLOAD( 0.3846 -5.3 10U 0.7 0.6 0.0  7 100 )
MODEL N053 NTXG ( 4.7500  0.5 10U 0.7 0.6 0.0 12 100 )
MODEL N054 NDRIV( 5.7500  0.5 10U 0.7 0.6 0.0 12 100 )
MODEL N055 NTXG ( 0.8750  0.5 10U 0.7 0.6 0.0 12 100 )
MODEL I056 NLOAD( 0.7143 -5.3 10U 0.7 0.6 0.0  7 100 )
MODEL N057 NDRIV(10.0000  0.5 10U 0.7 0.6 0.0 12 100 )
MODEL I060 NLOAD( 1.1667 -5.3 10U 0.7 0.6 0.0  7 100 )
MODEL I061 NLOAD( 1.6000 -5.3 10U 0.7 0.6 0.0  7 100 )
MODEL I062 NTXG ( 0.5000 -5.3 10U 0.7 0.6 0.0 12 100 )
MODEL N063 NDRIV( 3.3333  0.5 10U 0.7 0.6 0.0 12 100 )
MODEL N064 NDRIV( 2.0000  0.5 10U 0.7 0.6 0.0 12 100 )
MODEL I065 NLOAD( 0.3571 -5.3 10U 0.7 0.6 0.0  7 100 )
MODEL I066 NLOAD( 0.8125 -5.3 10U 0.7 0.6 0.0 12 100 )
*
* CAPACITOR MODEL
MODEL C GCAPR
*
* INPUT SOURCES
S1 15 VD1
S2  6 VD2
S3 83 CK1
S4 82 CK2
S5  1 SI1
S6 10 SI2
S7 135 VGN
*
*CIRCUIT GENERATED FROM ARTWORK FILE
M0001      2      1     83  N001
M0002      3      2         N002
M0003      3                I003
M0004      5      3     82  N004
M0005      7      5         N005
M0006      7                I006
M0007      8      6      7  I007
M0010      8-     5         N010
M0011      9      5         N011
M0012      9                I012
M0013     12     15      9  N001
M0014     14     12         N011
M0015     14                I013
M0016     16     45         N014
M0017     17     28         N015
M0020     17                I003
M0021     18                I016
M0022     18     40         N017
M0023     19                I020
M0024     19     21         N017
M0025     19     87         N021
M0026     20                I020
M0027     21     20     82  N001
M0030     20     22         N022
M0031     22     23     83  N001
M0032     16     25     26  N001
M0033     16                I023
M0034     25     17         N024
M0035     16     37         N017
M0036     26                I003
M0037     26     30         N015
M0040     27     29         N025
M0041     27                I026
M0042     28     27     82  N027
M0043     30     29     82  N027
M0044     32     83         N024
M0045     32     31      1  N001
M0046     29     31         N021
M0047     29                I026
M0050     34                I026
M0051     35     34     17  N001
M0052     34     36         N017
M0053     34     67         N017
M0054     35     37         N017
```

```
M0055     38     80          H017
M0056     38                 I026
M0057     39     41          H017
M0060     39     87          H017
M0061     40     39     82   H001
M0062     42     10     83   H001
M0063     43     42          H022
M0064     44     43     82   H001
M0065     43                 I030
M0066     45                 I030
M0067     45     44          H022
M0070     46     45     83   H001
M0071     47     46          H022
M0072     48     47     82   H001
M0073     47                 I030
M0074     49                 I030
M0075     49     48          H022
M0076     50     49     83   H001
M0077     51     50          H022
M0100     51                 I030
M0101     52                 I030
M0102     53     18          H017
M0103     53                 I003
M0104     54      6     56   H027
M0105     19     54     55   H027
M0106     24     18     95   H027
M0107     24     53     55   H027
M0110     24     18     56   H027
M0111     55                 I003
M0112     54     95          H031
M0113     55     56          H005
M0114     55     95          H005
M0115     57     58          H017
M0116     58     19          H017
M0117     59     19          H017
M0120     80     60     83   H001
M0121     62     61     82   H001
M0122     62     60          H022
M0123     62                 I026
M0124     63                 I032
M0125     63     61          H022
M0126     64     63     83   H001
M0127     66     65     82   H001
M0130     66     64          H022
M0131     66                 I026
M0132     67     65          H022
M0133     67                 I032
M0134     38     35     26   H001
M0135     69     35     83   H001
M0136     70     72          H017
M0137     70                 I033
M0140     71                 I026
M0141     71     69          H017
M0142     72     71     82   H001
M0143     73                 I003
M0144     23      6     57   H027
M0145     24     23     75   H027
M0146     74     24     59   H027
M0147     74     24     57   H027
M0150     75                 I003
M0151     23     59          H031
M0152     75     57          H005
```

```
M0153     75     59            H005
M0154     57     76            H021
M0155     57                   I003
M0156     58                   I003
M0157     76                   I003
M0160     76     25            H017
M0161     59     25            H017
M0162     59                   I003
M0163     77     68     83     H001
M0164     79     78     82     H001
M0165     79     77            H022
M0166     79                   I026
M0167     90                   I032
M0170     80     78            H022
M0171     84                   I020
M0172     85     84            H021
M0173     85                   I020
M0174     96                   I034
M0175     84     92            H017
M0176     84     97            H017
M0177     39                   I026
M0200     41     54     83     H001
M0201     88     51     82     H001
M0202     52     88            H022
M0203     89     52     83     H001
M0204     90     99            H022
M0205     91     90     82     H001
M0206     90                   I030
M0207     92                   I030
M0210     92     91            H022
M0211     56     94            H021
M0212     56                   I003
M0213     56     93            H017
M0214     93     86            H017
M0215     93                   I003
M0216     94                   I003
M0217     94     85            H017
M0220     95     85            H017
M0221     95                   I003
M0222     95     86            H017
M0223     73     24            H017
M0224     86     96            H017
M0225     86    120            H021
M0226     97    100            H035
M0227     98    100            H014
M0230     99    101            H022
M0231    100     99     82     H004
M0232     99                   I016
M0233     98                   I036
M0234     87      6     98     I037
M0235    194     87     83     H001
M0236      1    101     83     H001
M0237    198    194            H015
M0240    102     92     83     H001
M0241    103    102            H022
M0242    104    103     82     H001
M0243    103                   I030
M0244    105                   I030
M0245    105    104            H022
M0246    106    105     83     H001
M0247    107    106            H022
M0250    207    107     82     H001
```

```
M0251   107                I030
M0252   120                I030
M0253   120   207          H022
M0254   208   120    83    H001
M0255   108   110          H022
M0256   109                I032
M0257   109   111          H021
M0260   110   109    82    H027
M0261   109                I026
M0262   111    74    83    H001
M0263   112    74          H017
M0264   112                I032
M0265    74    73    75    H027
M0266   122    70          H040
M0267   122                I034
M0270   123                I016
M0271   123   108          H040
M0272   124                I026
M0273   124   125          H017
M0274   125   129          H017
M0275   126                I032
M0276   126   127          H022
M0277   137   127    83    H001
M0300   129                I003
M0301   129   130          H005
M0302   129   133          H005
M0303   130   132          H021
M0304   130                I003
M0305   130   131          H017
M0306   131   124          H017
M0307   131                I003
M0310   132                I003
M0311   132    70          H017
M0312   133    70          H017
M0313   133                I003
M0314   133   124          H017
M0315   134                I016
M0316   136                I032
M0317   125   126    82    H027
M0320   138                I041
M0321   139   138    82    H001
M0322   138   140          H017
M0323   141   140    93    H001
M0324   142   128          H040
M0325   142   139          H022
M0326   142                I016
M0327   143   147          H040
M0330   143                I033
M0331   145   144    83    H001
M0332   147   146    82    H027
M0333   148                I003
M0334   148   150          H022
M0335   149   147          H017
M0336   149                I033
M0337   151   150    92    H001
M0340   151                I042
M0341   151   145          H005
M0342   152                I034
M0343   152   145          H040
M0344    13                I013
M0345   153                I013
M0346   154   153    82    H004
```

```
M0347   153   178         H014
M0350   155   154         H025
M0351     6   156   155   I007
M0352   157   179         H010
M0353   156   154         H010
M0354   155               I006
M0355   159   123         H017
M0356   159               I003
M0357   137     6   130   H027
M0360   123   137   129   H027
M0361   160   123   133   H027
M0362   160   159   129   H027
M0363   160   123   130   H027
M0364   137   133         H031
M0365   161   149         H017
M0366   161               I003
M0367   141     6   163   H027
M0370   149   141   162   H027
M0371   144   149   166   H027
M0372   144   161   162   H027
M0373   144   149   163   H027
M0374   162               I003
M0375   141   166         H031
M0376   162   163         H005
M0377   162   166         H005
M0400   163   165         H021
M0401   163               I003
M0402   163   164         H017
M0403   164   142         H017
M0404   164               I003
M0405   165               I003
M0406   165   160         H017
M0407   166   160         H017
M0410   166               I003
M0411   166   142         H017
M0412   134   135         H014
M0413   167               I032
M0414   136   170         H040
M0415    68   169   134   H027
M0416    68   170    83   H027
M0417    68   167   135   H027
M0420   169    10         H043
M0421   169               I013
M0422   171   193         H015
M0423   171               I003
M0424   290               I003
M0425   193   290   292   H027
M0426   172   188         H015
M0427   174   186         H044
M0430   174               I023
M0431    15   176   174   H001
M0432   168   173         H045
M0433    15   168   177   I037
M0434   177               I013
M0435   177   173         H040
M0436     1   178    33   H001
M0437    13    10         H002
M0440   179   174         H017
M0441   180   176         H046
M0442   180               I003
M0443   179               I003
M0444   157               I006
```

```
M0445    167    181           N040
M0446    181    136     82    N027
M0447    297    128     83    N027
M0450    298    297           N014
M0451    298                  I047
M0452    174    184           N005
M0453    183                  I042
M0454    184    183     82    N027
M0455    183    185           N021
M0456    186    185     83    N001
M0457    172                  I003
M0460    187                  I003
M0461    172    173    186    N027
M0462    187    189           N045
M0463    188    187     82    N027
M0464    189    173     83    N027
M0465    171    173    190    N027
M0466    190                  I034
M0467    290    148           N017
M0470    315    316           N022
M0471    191                  I026
M0472    315                  I030
M0473    191    192           N022
M0474    316    191     82    N001
M0475    192     10     83    N001
M0476    317    315     83    N001
M0477    190    186           N021
M0500    186    182           N021
M0501    136                  I050
M0502    329     83           N031
M0503    329    330      1    N027
M0504    309    330           N022
M0505     37                  I013
M0506     36                  I013
M0507    196     36           N015
M0510     36    196           N015
M0511     36    225           N015
M0512     97    238           N015
M0513     97    195           N015
M0514    195     97           N015
M0515    195     87           N015
M0516    196     87           N014
M0517    197     87           N014
M0520     37    128           N015
M0521     37    197           N015
M0522    197     37           N015
M0523    197                  I013
M0524    196                  I013
M0525    198                  I016
M0526    128                  I051
M0527    128    199           N043
M0530    199    198     82    N001
M0531    201                  I052
M0532    201    202           N046
M0533    203    202     82    N001
M0534    203    206           N021
M0535    205    204    201    N053
M0536    205    206     83    N001
M0537    205    193           N054
M0540    205                  I033
M0541    209    208           N022
M0542    210    209     82    N001
```

```
M0543   209               I030
M0544   211               I030
M0545   211   210         H022
M0546   212   211    83   H001
M0547   213   212         H022
M0550   214   213    92   H001
M0551   213               I030
M0552   200               I030
M0553   200   214         H022
M0554   215   200    83   H001
M0555   216   215         H022
M0556   217   216    82   N001
M0557   216               I030
M0560   218               I030
M0561   219   128    83   H055
M0562   221   220    82   H001
M0563   221   219         H022
M0564   221               I026
M0565   222               I032
M0566   222   220         H022
M0567   223   222    83   H001
M0570   224   223         H022
M0571   224               I026
M0572   225               I032
M0573   226   291         H025
M0574   227               I032
M0575   228               I056
M0576   227   229         H015
M0577    81   231         H057
M0600   228   230         H021
M0601   229   223    82   H001
M0602   230   226    83   H001
M0603   231   232    82   H001
M0604   232               I052
M0605   232   233         H014
M0606   234   204    93   H001
M0607   234   233   128   H001
M0610   201   204   205   H053
M0611   203               I052
M0612   204               I033
M0613    96   264         H015
M0614    97               I060
M0615   195               I013
M0616   236   235         H005
M0617   239   238         H005
M0620   240   239         H025
M0621   240   274         H005
M0622   241   242         H005
M0623   242   225         H005
M0624   242               I052
M0625   226               I026
M0626   218   217         H022
M0627   243   218    93   H001
M0630   244   243         H022
M0631   245   244    92   H001
M0632   244               I030
M0633   246               I030
M0634   246   245         H022
M0635   247   246    83   H001
M0636   248   247         H022
M0637   249   248    82   H001
M0640   248               I030
```

```
M0641    250                  I030
M0642    250     249          H022
M0643    224     251      82  H001
M0644    225     251          H022
M0645    252     225      83  H001
M0646    254     253      92  H001
M0647    254     252          H022
M0650    254                  I026
M0651    238                  I032
M0652    238     253          H022
M0653    255     238      83  H001
M0654    257     256      82  H001
M0655    257     255          H022
M0656    257                  I026
M0657    235                  I032
M0660    235     256          H022
M0661     96                  I061
M0662    258     260          H021
M0663    259     260          H021
M0664    259                  I034
M0665    258       6     259  I062
M0666     96     263          H015
M0667    263      96          H015
M0670    263      87          H015
M0671    262      87          H015
M0672    262     261          H015
M0673    261     262          H015
M0674    261     273          H015
M0675    260     273          H015
M0676    260                  I056
M0677    263                  I013
M0700    261                  I013
M0701    262                  I013
M0702    237     236          H005
M0703    237     279          H005
M0704    265     264          H005
M0705    266     265          H063
M0706    267     235      83  H001
M0707    269     268      82  H001
M0710    269     267          H022
M0711    269                  I026
M0712    264                  I032
M0713    264     268          H022
M0714    270     264      83  H001
M0715    272     271      82  H001
M0716    272     270          H022
M0717    272                  I026
M0720    273                  I032
M0721    273     271          H022
M0722    274     276          H064
M0723    274                  I065
M0724    275                  I065
M0725    275     276          H005
M0726    276                  I065
M0727    276     278          H064
M0730    266     276          H005
M0731    277                  I065
M0732    277     278      82  H001
M0733    237                  I065
M0734    281     237          H005
M0735    231     240          H005
M0736    239                  I065
```

M0737	240			I065
M0740	241			I065
M0741	241	280		H005
M0742	281	241		H064
M0743	281			I065
M0744	265			I065
M0745	236			I065
M0746	281	266		H005
M0747	280			I065
M0750	266			I065
M0751	282	283		H005
M0752	277	285		H064
M0753	280	276		H005
M0754	280	286		H005
M0755	285	287	1	H001
M0756	287	286	83	H001
M0757	279	286		H064
M0760	279			I052
M0761	282	299	82	H001
M0762	282			I030
M0763	275	288		H064
M0764	291	6	293	H027
M0765	148	291	292	H027
M0766	193	148	296	H027
M0767	193	148	293	H027
M0770	292			I003
M0771	291	296		H031
M0772	292	293		H005
M0773	292	296		H005
M0774	293	295		H021
M0775	293			I003
M0776	294			I003
M0777	295			I003
M1000	295	227		H017
M1001	296	227		H017
M1002	296			I003
M1003	299			I034
M1004	146			I020
M1005	146	306		H040
M1006	300	298	82	H027
M1007	299	300		H005
M1010	301	291	83	H027
M1011	302	301		H017
M1012	302	303	82	H027
M1013	302			I034
M1014	304			I047
M1015	304	303		H017
M1016	304	299		H017
M1017	306	305	83	H027
M1020	307			I003
M1021	307	308		H015
M1022	309	308	82	H027
M1023	311	310	82	H027
M1024	311			I016
M1025	312	310		H015
M1026	312			I003
M1027	313	97		H040
M1030	10	314	307	H027
M1031	293	294		H017
M1032	294	304		H017
M1033	296	304		H017
M1034	318	317		H022

```
M1035   319   318    82   H001
M1036   313               I030
M1037   320               I030
M1040   320   319         H022
M1041   321   320    33   H001
M1042   322   321         H022
M1043   323   322    82   H001
M1044   322               I030
M1045   324               I030
M1046   324   323         H022
M1047   325   324    83   H001
M1050   326   325         H022
M1051   327   326    82   H001
M1052   326               I030
M1053   328               I030
M1054   309               I066
M1055   311   309         H040
M1056   284   283     1   H001
M1057    15   284    83   H027
M1060   331               I036
M1061   331   182    82   H027
M1062   334   333.        H017
M1063   334               I003
M1064   333   332   332   H027
M1065   305   333   336   H027
M1066   305   334   352   H027
M1067   305   333   353   H027
M1070   331   335         H014
M1071   261   335    83   H027
M1072   332   336    83   H027
M1073   286   337         H040
M1074   286               I052
M1075   275   286         H005
M1076   338   337    82   H001
M1077   338   339         H005
M1100   338               I052
M1101   340   339     1   H001
M1102   340   275    93   H001
M1103   341   336         H040
M1104   341    97         H017
M1105   342               I047
M1106   343               I026
M1107   344               I032
M1110   344   374         H022
M1111   345   307         H031
M1112    87   345   312   H027
M1113   346    87   307   H027
M1114   346   312         H031
M1115   347               I042
M1116   341               I042
M1117   349   348   312   H027
M1120   349    96         H040
M1121   349   364         H040
M1122   313   328         H040
M1123   313               I016
M1124   314     6   312   H027
M1125   350   313   312   H027
M1126   351   313         H015
M1127   351   350   307   H027
M1130   351               I003
M1131   349               I016
M1132     6   332   353   H027
```

```
M1133   352                 I003
M1134   332    356          H031
M1135   352    353          H005
M1136   352    356          H005
M1137   353    355          H021
M1140   353                 I003
M1141   353    354          H017
M1142   354    347          H017
M1143   354                 I003
M1144   355                 I003
M1145   355    314          H017
M1146   356    314          H017
M1147   356                 I003
M1150   356    347          H017
M1151   328    327          H022
M1152   357    328    83    H001
M1153   358    357          H022
M1154   359    358    82    H001
M1155   358                 I030
M1156   360                 I030
M1157   360    359          H022
M1160   361    360    83    H001
M1161   362    361          H022
M1162   363    362    82    H001
M1163   362                 I030
M1164   364                 I030
M1165   364    363          H022
M1166   341    365    82    H027
M1167   347    365          H021
M1170   342    333          H005
M1171   366    368          H002
M1172   366    345          H021
M1173   366                 I003
M1174   368    367    82    H027
M1175   367                 I047
M1176   367    346          H021
M1177   367    371          H021
M1200   369                 I016
M1201   369    348   307    H027
M1202   369    261          H040
M1203   369    390          H040
M1204   372    371    83    H027
M1205   379    373    83    H001
M1206   343    374    82    H001
M1207   343    373          H022
M1210   382    375    83    H001
M1211   377    376    82    H001
M1212   377    375          H022
M1213   377                 I026
M1214   378                 I032
M1215   379    376          H022
M1216   386    379    83    H001
M1217   381    380    82    H001
M1220   381    379          H022
M1221   381                 I026
M1222   382                 I032
M1223   382    380          H022
M1224   383    370    83    H001
M1225   395    384    82    H001
M1226   385    383          H022
M1227   385                 I026
M1230   386                 I032
```

```
M1231    386   384         N022
M1232    387   364    83   N001
M1233    388   387         N022
M1234    389   388    82   N001
M1235    388               I030
M1236    390               I030
M1237    390   389         N022
M1240    391   390    83   N001
M1241    392   391         N022
M1242    393   392    82   N001
M1243    392               I030
M1244    370               I030
M1245    370   393         N022
M1246    394   366         N017
M1247    394               I003
M1250    372     6   396   N027
M1251    366   372   395   N027
M1252    333   366   399   N027
M1253    333   394   395   N027
M1254    333   366   396   N027
M1255    395               I003
M1256    372   399         N031
M1257    395   396         N005
M1260    395   399         N005
M1261    396   398         N021
M1262    396               I003
M1263    396   397         N017
M1264    397   350         N017
M1265    397               I003
M1266    398               I003
M1267    398   348         N017
M1270    399   348         N017
M1271    399               I003
M1272    399   350         N017
*
* NODE CAPACITANCES ( COMPUTED FROM ARTWORK FILE )
C0001      1 C       .9617P
C0002      2 C       .0856P
C0003      3 C       .0631P
C0004      5 C       .2714P
C0005      6 C      3.6071P
C0006      7 C       .0744P
C0007      8 C       .0688P
C0010      9 C       .0790P
C0011     10 C       .0668P
C0013     12 C       .1172P
C0014     13 C       .4862P
C0015     14 C       .0654P
C0016     15 C      3.4672P
C0017     16 C       .0696P
C0020     17 C       .1368P
C0021     18 C       .1214P
C0022     19 C       .1534P
C0023     20 C       .0644P
C0024     21 C       .0634P
C0025     22 C       .0689P
C0026     23 C       .0606P
C0027     24 C       .1842P
C0030     25 C       .1775P
C0031     26 C       .1544P
C0032     27 C       .0674P
C0033     28 C       .0783P
```

```
C0034     29  C        .0939P
C0035     30  C        .0738P
C0036     31  C        .1096P
C0037     32  C        .0673P
C0040     34  C        .0737P
C0041     35  C        .0601P
C0042     36  C        .4301P
C0043     37  C        .4709P
C0044     38  C        .0672P
C0045     39  C        .0772P
C0046     40  C        .0691P
C0047     41  C        .0737P
C0050     42  C        .0742P
C0051     43  C        .0602P
C0052     44  C        .0701P
C0053     45  C        .1161P
C0054     46  C        .0742P
C0055     47  C        .0602P
C0056     48  C        .0701P
C0057     49  C        .0625P
C0060     50  C        .0742P
C0061     51  C        .0602P
C0062     52  C        .0625P
C0063     53  C        .0627P
C0064     54  C        .0785P
C0065     55  C        .0920P
C0066     56  C        .1625P
C0067     57  C        .1610P
C0070     58  C        .1010P
C0071     59  C        .1563P
C0072     60  C        .0746P
C0073     61  C        .0704P
C0074     62  C        .0686P
C0075     63  C        .0607P
C0076     64  C        .0746P
C0077     65  C        .0704P
C0100     66  C        .0686P
C0101     67  C        .0989P
C0102     68  C        .1670P
C0103     69  C        .0683P
C0104     70  C        .2767P
C0105     71  C        .0625P
C0106     72  C        .0694P
C0107     73  C        .0627P
C0110     74  C        .0908P
C0111     75  C        .0920P
C0112     76  C        .0969P
C0113     77  C        .0746P
C0114     78  C        .0704P
C0115     79  C        .0696P
C0116     80  C        .1019P
C0117     81  C        .4022P
C0120     82  C       3.1465P
C0121     83  C       3.0046P
C0122     84  C        .1019P
C0123     85  C        .1703P
C0124     86  C        .1966P
C0125     87  C       1.1171P
C0126     88  C        .0701P
C0127     89  C        .0724P
C0130     90  C        .0602P
C0131     91  C        .0701P
```

```
C0132     92 C        .1034P
C0133     93 C        .1010P
C0134     94 C        .0969P
C0135     95 C        .1563P
C0136     96 C        .6314P
C0137     97 C        .6641P
C0140     98 C        .0728P
C0141     99 C        .0671P
C0142    100 C        .1556P
C0143    101 C        .0691P
C0144    102 C        .0742P
C0145    103 C        .0602P
C0146    104 C        .0701P
C0147    105 C        .0625P
C0150    106 C        .0742P
C0151    107 C        .0602P
C0152    108 C        .2290P
C0153    109 C        .0659P
C0154    110 C        .0677P
C0155    111 C        .0658P
C0156    112 C        .0671P
C0166    120 C        .1137P
C0170    122 C        .0769P
C0171    123 C        .2139P
C0172    124 C        .1361P
C0173    125 C        .0771P
C0174    126 C        .0689P
C0175    127 C        .0668P
C0176    128 C        .7281P
C0177    129 C        .0920P
C0200    130 C        .1625P
C0201    131 C        .1010P
C0202    132 C        .0969P
C0203    133 C        .1563P
C0204    134 C        .1057P
C0205    136 C        .0625P
C0206    137 C        .0610P
C0207    138 C        .0686P
C0210    139 C        .0668P
C0211    140 C        .0692P
C0212    141 C        .0624P
C0213    142 C        .1493P
C0214    143 C        .0728P
C0215    144 C        .0601P
C0216    145 C        .1013P
C0217    146 C        .1916P
C0220    147 C        .1319P
C0221    148 C        .2672P
C0222    149 C        .1445P
C0223    150 C        .0710P
C0224    151 C        .0630P
C0225    152 C        .0775P
C0226    153 C        .0681P
C0227    154 C        .1839P
C0230    155 C        .0750P
C0231    156 C        .5385P
C0232    157 C        .5546P
C0234    159 C        .0627P
C0235    160 C        .1609P
C0236    161 C        .0627P
C0237    162 C        .0920P
C0240    163 C        .1625P
```

```
C0241   164 C       .1010P
C0242   165 C       .0969P
C0243   166 C       .1563P
C0244   167 C       .0673P
C0245   168 C       .4528P
C0246   169 C       .0809P
C0247   170 C       .0632P
C0250   171 C       .1121P
C0251   172 C       .0808P
C0252   173 C       .1817P
C0253   174 C       .1176P
C0255   176 C       .1191P
C0256   177 C       .0621P
C0257   178 C       .0801P
C0260   179 C       .1517P
C0261   180 C       .0698P
C0262   181 C       .0636P
C0263   182 C       .2795P
C0264   183 C       .0609P
C0265   184 C       .0674P
C0266   185 C       .0659P
C0267   186 C       .1758P
C0270   187 C       .0628P
C0271   188 C       .0809P
C0272   189 C       .0790P
C0273   190 C       .0686P
C0274   191 C       .0682P
C0275   192 C       .0749P
C0276   193 C       .3353P
C0277   194 C       .0794P
C0300   195 C       .1365P
C0301   196 C       .1242P
C0302   197 C       .1275P
C0303   198 C       .0605P
C0304   199 C       .1037P
C0305   200 C       .1921P
C0306   201 C       .1352P
C0307   202 C       .1221P
C0310   203 C       .0639P
C0311   204 C       .0903P
C0312   205 C       .1371P
C0313   206 C       .0662P
C0314   207 C       .0701P
C0315   208 C       .0742P
C0316   209 C       .0602P
C0317   210 C       .0701P
C0320   211 C       .0625P
C0321   212 C       .0742P
C0322   213 C       .0602P
C0323   214 C       .0701P
C0324   215 C       .0742P
C0325   216 C       .0602P
C0326   217 C       .0701P
C0327   218 C       .0625P
C0330   219 C       .0744P
C0331   220 C       .0704P
C0332   221 C       .0686P
C0333   222 C       .0607P
C0334   223 C       .0711P
C0335   224 C       .0686P
C0336   225 C       .1924P
C0337   226 C       .1113P
```

```
C0340   227 C       .2255P
C0341   228 C       .0618P
C0342   229 C       .0745P
C0343   230 C       .0601P
C0344   231 C       .1629P
C0345   232 C       .0677P
C0346   233 C       .1216P
C0347   234 C       .0673P
C0350   235 C       .1358P
C0351   236 C       .1056P
C0352   237 C       .1153P
C0353   238 C       .2232P
C0354   239 C       .1072P
C0355   240 C       .1019P
C0356   241 C       .0920P
C0357   242 C       .0786P
C0360   243 C       .0742P
C0361   244 C       .0602P
C0362   245 C       .0701P
C0363   246 C       .0625P
C0364   247 C       .0742P
C0365   248 C       .0602P
C0366   249 C       .0701P
C0367   250 C       .1142P
C0370   251 C       .0704P
C0371   252 C       .0616P
C0372   253 C       .0704P
C0373   254 C       .0616P
C0374   255 C       .0746P
C0375   256 C       .0704P
C0376   257 C       .0686P
C0377   258 C       .1490P
C0400   259 C       .0748P
C0401   260 C       .1319P
C0402   261 C       .5328P
C0403   262 C       .1161P
C0404   263 C       .1356P
C0405   264 C       .2319P
C0406   265 C       .0974P
C0407   266 C       .1211P
C0410   267 C       .0746P
C0411   268 C       .0704P
C0412   269 C       .0686P
C0413   270 C       .0746P
C0414   271 C       .0704P
C0415   272 C       .0686P
C0416   273 C       .1802P
C0417   274 C       .1997P
C0420   275 C       .1838P
C0421   276 C       .2410P
C0422   277 C       .0602P
C0423   278 C       .0630P
C0424   279 C       .1885P
C0425   280 C       .1657P
C0426   281 C       .1886P
C0427   282 C       .0692P
C0430   293 C       .0788P
C0431   284 C       .0618P
C0432   285 C       .1531P
C0433   286 C       .2312P
C0434   297 C       .0673P
C0435   238 C       .0699P
```

```
C0437   290 C      .0627P
C0440   291 C      .0639P
C0441   292 C      .0920P
C0442   293 C      .1625P
C0443   294 C      .1010P
C0444   295 C      .0969P
C0445   296 C      .1563P
C0446   297 C      .0709P
C0447   298 C      .0614P
C0450   299 C      .1293P
C0451   300 C      .0624P
C0452   301 C      .0628P
C0453   302 C      .0684P
C0454   303 C      .0604P
C0455   304 C      .1527P
C0456   305 C      .1081P
C0457   306 C      .0686P
C0460   307 C      .2345P
C0461   308 C      .0744P
C0462   309 C      .1071P
C0463   310 C      .0842P
C0464   311 C      .0698P
C0465   312 C      .2220P
C0466   313 C      .1401P
C0467   314 C      .1711P
C0470   315 C      .0625P
C0471   316 C      .0703P
C0472   317 C      .0742P
C0473   318 C      .0602P
C0474   319 C      .0701P
C0475   320 C      .0625P
C0476   321 C      .0742P
C0477   322 C      .0698P
C0500   323 C      .0701P
C0501   324 C      .0625P
C0502   325 C      .0742P
C0503   326 C      .0602P
C0504   327 C      .0701P
C0505   328 C      .1088P
C0506   329 C      .0697P
C0507   330 C      .1004P
C0510   331 C      .0637P
C0511   332 C      .0665P
C0512   333 C      .2217P
C0513   334 C      .0627P
C0514   335 C      .0720P
C0515   336 C      .0638P
C0516   337 C      .0643P
C0517   338 C      .0676P
C0520   339 C      .1673P
C0521   340 C      .0673P
C0522   341 C      .0682P
C0523   342 C      .0814P
C0524   343 C      .0686P
C0525   344 C      .0617P
C0526   345 C      .0813P
C0527   346 C      .0922P
C0530   347 C      .1796P
C0531   348 C      .1487P
C0532   349 C      .0643P
C0533   350 C      .1724P
C0534   351 C      .0637P
```

```
C0535   352 C      .0920P
C0536   353 C      .1625P
C0537   354 C      .1010P
C0540   355 C      .0969P
C0541   356 C      .1563P
C0542   357 C      .0742P
C0543   358 C      .0602P
C0544   359 C      .0701P
C0545   360 C      .0625P
C0546   361 C      .0742P
C0547   362 C      .0602P
C0550   363 C      .0701P
C0551   364 C      .1100P
C0552   365 C      .0611P
C0553   366 C      .1397P
C0554   367 C      .0630P
C0555   368 C      .0823P
C0556   369 C      .0672P
C0557   370 C      .0795P
C0560   371 C      .0602P
C0561   372 C      .0634P
C0562   373 C      .0746P
C0563   374 C      .0704P
C0564   375 C      .0746P
C0565   376 C      .0704P
C0566   377 C      .0686P
C0567   378 C      .0607P
C0570   379 C      .0746P
C0571   380 C      .0704P
C0572   381 C      .0686P
C0573   382 C      .0607P
C0574   383 C      .0746P
C0575   394 C      .0704P
C0576   365 C      .0686P
C0577   396 C      .0607P
C0600   387 C      .0742P
C0601   388 C      .0602P
C0602   389 C      .0701P
C0603   390 C      .1163P
C0604   391 C      .0742P
C0605   392 C      .0602P
C0606   393 C      .0701P
C0607   394 C      .0627P
C0610   395 C      .0920P
C0611   396 C      .1532P
C0612   337 C      .1010P
C0613   398 C      .0969P
C0614   399 C      .1563P
*
* CONTROL STATEMENTS
OPTS 1 14.0 0
TOPTS 0.1 0.05  8 0.1
*TIME 5N 4000N
TIME  1N 420N
PLOT 70, 123, 124, 160, 137, 149, 144
PLOT 92,  83,   1,  10
*
GO
*
END
$E
```

APPENDIX 8

MOTIS-C: A New Circuit Simulator for MOS LSI Circuits

Presented at the 1977 IEEE International Symposium on Circuits and Systems, Phoenix, Arizona, April 1977.

MOTIS-C: A NEW CIRCUIT SIMULATOR FOR MOS LSI CIRCUITS

S. P. Fan, M. Y. Hsueh, A. R. Newton and D. O. Pederson

Department of Electrical Engineering and Computer Sciences
University of California, Berkeley, California 94720

Abstract

Recently a new type of circuit simulator, MOTIS, was reported for the efficient simulation of large-scale, integrated, MOS logic circuits. A new version of this type of program, MOTIS-C has been developed employing several new analysis techniques evolved from those used in MOTIS. The program takes data input at the logic gate level and plots time-domain waveforms at selected output nodes. The effecitveness of simulating large logic circuits relies on direct solution methods at each timepoint and a table look-up scheme for the evaluation of non-linear device currents.

The program is written in FORTRAN, has a SPICE-like input language for user convenience and the details of gates are macromodeled to minimize computation.

Preliminary results indicate the speed and circuit size capability of MOTIS-C are over two orders of magnitude greater than those of conventional time-domain circuit simulators such as SPICE2.

1. Introduction

The recently announced program MOTIS [1] is a new type of circuit analysis program designed explicitly for the simulation of large-scale MOS integrated circuits. MOTIS was written in assembly language on an HP2100 minicomputer. Many of the analysis techniques used in MOTIS have been adapted for use in a new version of this type of program, MOTIS-C (California). MOTIS-C simulates circuits at the device level but the method used for propagating voltage signals from node to node is similar to that used in logic simulators. The result is time-domain waveforms produced at much higher speeds than conventional circuit simulators such as SPICE2 [2,3]. MOTIS-C is written completely in FORTRAN and at present is being run on the CDC6400 computer at the University of California, Berkeley. MOTIS-C does not rely on the 60-bit word available on CDC machines, and could be run on a minicomputer after minor program modifications.†

The program stores all circuit node voltages on a disc file which is later interrogated by the output module of MOTIS-C. At present, line-printer plots of voltage waveforms are available and MOTIS-C can send data to an HP9830 programmable calculator for off-line plotting of an X-Y plotter or a Tektronix graphics display.

Performance of the prototype program indicates up to 2,500 gates can be analyzed in 140K octal memory on the CDC6400 computer at Berkeley, with an average central processor time of under 0.5ms per gate for each timepoint. Improvements of up to one order of magnitude are expected as the program is further refined. It should also be noted that in contrast to conventional circuit simulators the analysis time of MOTIS-C increases almost linearly with circuit size.

```
BINARY-TO-OCTAL DECODER
* SUBCIRCUIT MODEL CARDS
MODEL INVTR MNOS( 1 1 0.2 10F 10F 100F 0 1)
MODEL NOR MNOS( 1 1 0.2 10F 10F 100F 0 2)
MODEL NAND MNOS( 1 1 0.2 10F 10F 100F 2 0)
MODEL CLK SOURCE( 5 0 40NS 10NS 40NS 10NS)
*
* NOR GATES
X1 5 6 9 NOR
X2 2 5 10 NOR
X3 1 6 11 NOR
X4 1 2 12 NOR
X5 3 4 13 NOR
X6 4 7 14 NOR
*
* NAND GATES
X7 12 13 17 NAND
X8 10 13 18 NAND
X9 11 13 19 NAND
X10 9 13 20 NAND
X11 12 14 21 NAND
X12 10 14 22 NAND
X13 11 14 23 NAND
X14 9 14 24 NAND
*
* INPUT SIGNAL INVERTERS
X15 1 5 INVTR
X16 2 6 INVTR
X17 3 7 INVTR
*
* INPUT SOURCES
VA 1 0 CLK 0 1
VB 2 0 CLK 0 0 1 1
VC 3 0 CLK 0 0 0 0 1 1 1 1
VD 4 0 CLK 0
*
* ANALYSIS REQUESTS
TIME 300NS 2NS
PLOT 17 18 19 20 21 22 23 24
PLOT 1 2 3 4
*
* GLOBAL PARAMETERS
VD 5
END
```

Fig. 1 Typical MOTIS-C Input Data.

2. Circuit Description

An integrated circuit is described to the program by a set of data records very similar to those used in Program SPICE2. (An example of an input file to MOTIS-C is included as Fig. 1.) Each circuit element is a logic gate consisting of a collection of MOS devices, hence the term

† MOTIS is a proprietary program within Bell Laboratories, Murray Hill, N.J. However, MOTIS-C is available in the public domain. Interested parties should contact Professor D. O. Pederson at the above address.

macromodel, the parameters of which are described using a MODEL statement. Floating capacitors, as well as grounded capacitors, are included so that bootstrapping and dynamic logic families can be analyzed.

The non-linear device tables are generated by MOTIS-C using a modified form of the Frohmann-Grove MOSFET equations [4]. These tables are written on a file which can be saved on disc. They are automatically attached and may be used by the program during later analyses. The input and output capacitances for the gate macromodels are obtained by comparing switching simulations on SPICE2 and MOTIS-C and then choosing the appropriate values to match the results.

3. Program Structure

The program is written in a modular style with a main program calling three sub-programs in sequence. All model and device data is stored in labeled common blocks with a format decided during the read-in phase. To improve program speed during analysis, the program bypasses any macromodel whose driving voltages have not changed significantly over the past two timepoints. It also writes only node voltage changes to the output file for later interrogation by the post-preccessor.

4. Table Functions for Nonlinear Elements

For each nonlinear device type, MOTIS-C stores a table of output currents which is indexed directly by the appropriate combination of controlling voltages. Circuit voltages are scaled to permit direct indexing from rounded node voltages; for MOS devices and positive supply voltage is scaled to an internal voltage of 50 volts. In order to allow a limited range of capacitive bootstrapping, the device tables are stored for controlling-voltage values to twice the positive supply voltage.

Load devices are indexed directly by their independent node voltage as in MOTIS. Driver and transfer gate currents are more difficult to calculate. Fig. 2(a) shows a typical device with its three controlling voltages. MOTIS uses a two-dimensional array, indexed by both V_{gs} and V_{ds}, and a one-dimensional back-gate bias table to modify the final current in accordance with back-gate threshold shift. In MOTIS-C, however, 100-entry one-dimensional tables are used for all device characteristics. A vector of Id's, indexed by V_{ds} at the highest expected gate voltage, $V_{gs}(max)$, a vector of output conductances for 100 different gate voltages up to $V_{gs}(max)$ and a back-gate bias vector are stored.

Drain currents at any gate voltage below $V_{gs}(max)$ are obtained by an origin shifting operation on the single stored characteristic:

$$I_d(V_{gs},V_{ds}) = I_d(V_{gs}(max),\ V_{ds} + \Delta V) - I_d(V_{gs}(max),\ \Delta V) \quad (1)$$

where

$$\Delta V = V_{gs}(max) - V_{gs} \quad (2)$$

The drain characteristics of a real device and those generated by the program using the above transformation are compared in Fig. 2(b).

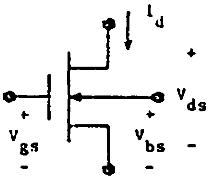


Fig. 2a MOSFET Transfer Device

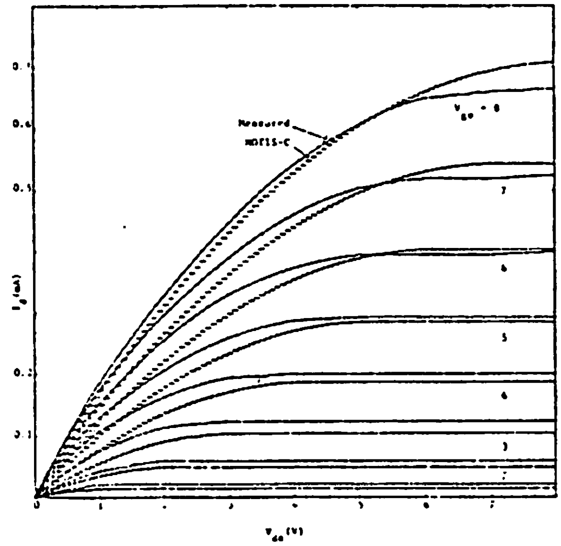


Fig. 2b Comparison of MOTIS-C MOSFET model and a typical device

5. Macromodeling of Logic Gates

At present, MOTIS-C has three logic gate macromodels, the AND-OR-INVERT, OR-AND-INVERT, and the transfer gate. (The AND-OR-INVERT and OR-AND-INVERT structures are shown in Fig. 3.) While these gates are analyzed using techniques very similar to those used in MOTIS [1], the transfer gate macromodel presents a more difficult problem. Since the device current is a function of two independent quantities, a set of two simultaneous equations must be solved at each timepoint, which in turn would substantially increase solution time.

The transfer gate may be represented as a nonlinear, voltage controlled current source as shown in Fig. 4(a), with nodes (1) and (2) as the controlling nodes:

$$I_T = f(V_1 - V_2) = f(x) \quad (3)$$

A first-order Taylor series expansion yields:

$$I_{t_{n+1}} = I_{T_n} + G_T\Delta x = I_{T_n} + G_T(\Delta V_1 - \Delta V_2) \quad (4)$$

where

$$G_T = \frac{\partial f}{\partial x} \text{ and } \Delta x = \Delta V_1 - \Delta V_2 \qquad (5)$$

At node (1), Equation (4) may be rewritten as:

$$I_{T_{n+1}} = G_T \Delta V_1 + (I_{T_n} - G_T \Delta V_2) \qquad (6)$$

The equivalent circuit of Fig. 4(b) may be used to represent this equation. Since ΔV_2 is not known at this stage, rather than use Eqn. (6), an explicit substitution is made for ΔV_2:

$$I_{T_{n+1}} = G_T \Delta V_1 + (I_{T_n} - G_T \delta V_2) \qquad (7)$$

where

$$\delta V_2 = V_{2_n} - V_{2_{n-1}} \qquad (8)$$

A similar process is carried out at node (2) and the resulting equivalent circuit for the ΔV's is shown in Fig. 4(c). The equations are now decoupled and may be solved separately.

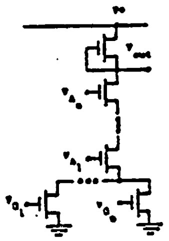


Fig. 3a OR-AND-INVERT Gate Structure

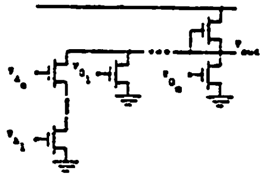


Fig. 3b AND-OR-INVERT Gate Structure

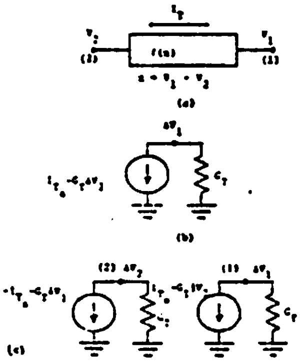


Fig. 4 Transfer Gate Models

6. Analysis Algorithms

MOTIS uses the backward Euler method, a first-order method, for integrating the capacitor equations. Previous programs of this type have used the inherently unstable forward Euler method. MOTIS-C uses Trapezoidal integration, a second-order method, which requires the saving of capacitor currents and only one more subtraction at each iteration. MOTIS-C solves for incremental node voltages at each timepoint using nodal analysis. Internal circuit voltages, capacitors and device look-up tables are scaled to permit the tables to be indexed directly by rounded node voltages.

While MOTIS-C is similar to MOTIS in that it uses only one iteration per timepoint, it maintains accuracy by automatically selecting a timestep to limit the maximum ΔV at each node. Fortunately, as the gate output reacnes either the positive or negative supply voltage, accumulated voltage errors are lost and the accuracy of the simulation is enhanced.

Floating capacitors cannot be handled using the decoupling process described for the transfer gate. The decoupling scheme results in a stable but inaccurate integration method. Instead, the two simultaneous equations describing the capacitor node voltages are solved directly. By insuring that the integration step size is less than the zero-valued time constant at each capacitor node, the number of long operations for an accurate solution at each timepoint may be reduced to 6 per floating capacitor, significantly reducing the overhead.

7. Timestep Selection

At present MOTIS-C does not use automatic timestep control durinc analysis, because most conventional timestep control scnemes are too expensive. However the program does select an internal timestep during the equation steup phase prior to analysis. As the program accumulates the node capacitances, it saves the value of the smallest node capacitance it finds, C_{min}, at any node other than a source node. The maximum node voltage swing, V_{max}, is estimated from the positive and negative supply voltages and the maximum gate pulldown current, I_{max}, is obtained from the driver table. The program then computes an internal timestep, h, using:

$$h = \frac{C_{min}\ \Delta V_{max}}{I_{max}} \qquad (9)$$

where

$$\Delta V_{max} = F.V_{max} \qquad (10)$$

F is an empirical constant chosen to maintain the difference between SPICE2 and MOTIS-C output waveforms at less than 10% for most circuits, while maximizing the speed of MOTIS-C.

8. Simulation Results

Table (1) shews a comparison between SPICE2 and MOTIS-C for two typical logic circuits. In both cases, SPICE2 was using assembly language routines on the CDC6400 computer to solve the set of linear equations at each Newton-Raphson iteration. It should be noted that in the 4-bit adder example, the SPICE2 simulation was performed on a Cyber 73 and not on the Berkeley CDC6400. MOTIS-C uses fixed length arrays for data storage. In both these examples, a substantial amount of storage remained for more elements and MOTIS-C was over two orders of magnitude faster than SPICE2 for 10% accuracy in the node voltage waveforms.

CIRCUIT	SPICE 2			MOTIS-C		
	Number of MOSFETS	Iterations	CPU Time(s)	Number of Gates	Analysis Points	CPU Time (s)
		Timepoints	Memory (K_8)			Memory (K_8)
Binary-to-Octal Decoder	48	1883	359	17	200	1.95
		422	120			27
4-bit NMOS Binary Adder	108	14865	6435	36	2200	36.4
		3305	150			27

Table 1: SPICE 2 and MOTIS-C Comparison for two typical circuits.

9. Conclusions

MOTIS-C has displayed a definite speed and size advantage over conventional circuit simulation programs for a special class of integrated circuits, namely MOS digital LSI circuits. The analysis techniques rely on saturating devices, repetitive structures such as gates and a relatively low circuit connectivity.

For handling circuits of this size, both convenient data input schemes and output data processing techniques must be employed. For this reason, MOTIS-C allows repetitive post-processing of data within the large computer or on an off-line, intelligent terminal. Ease of program modification is also most important and the modular structure and FORTRAN language of MOTIS-C permit the addition of new macromodels without detailed knowledge of the entire program operation.

10. Acknowledgements

The authors wish to thank G. R. Boyle and R. Smith for helpful discussions. This project was supported by Corporate Engineering at the Hewlett Packard Company, Palo Alto, California.

11. References

[1] R. R. Chawla, H. K. Gummel and P. Kozak, "MOTIS-An MOS Timing Simulator," Trans. IEEE, Vol. CAS-22, No. 12, pp. 901-910, Dec. 1975.

[2] L. W. Nagel, "SPICE2: A Computer Program to Simulate Semiconductor Circuits," ERL Memo No. ERL-M520, Electronics Research Laboratory, University of California, Berkeley, May, 1975.

[3] E. Cohen, "Program Reference for SPICE2," ERL Memo No. ERL-M592, Electronics Research Laboratory, University of California, Berkeley, June 1976.

[4] A. S. Grove, Physics and Technology of Semiconductor Devices, New York: Wiley, 1967.

APPENDIX 9

Analysis Time, Accuracy and Memory Requirement Tradeoffs in SPICE2

Presented at the Eleventh Annual Asilomar Conference on Circuits, Systems and Computers, Asilomar, California, November 1977.

ANALYSIS TIME, ACCURACY AND MEMORY REQUIREMENT TRADEOFFS IN SPICE2

A. R. Newton and D. O. Pederson
Department of Electrical Engineering and Computer Sciences
and the Electronics Research Laboratory
University of California, Berkeley, California 94720

Abstract

SPICE2 has proven to be an effective electronic circuit simulation program. Nonetheless, needs exist for faster computational performance and the ability to simulate economically larger circuits, especially large MOS circuits. Tradeoffs are necessary between accuracy and memory with circuit size and between accuracy and computational speed. For a benchmark example of a binary-to-octal decoder, the above aspects are illustrated. In addition, computational time that is required for different levels of device models is illustrated together with the savings achieved using a device bypass scheme and table lookup models.

1. INTRODUCTION

Many decisions must be made in the design of an electronic circuit simulation program which are based on the types of circuits the program is intended to simulate and the computer on which the program is to be executed. As circuit and device types and computers change, it is necessary to review these decisions and make adjustments and improvements where possible.

SPICE2 [1], [2] is an integrated-circuit simulation program which performs dc operating point, transfer curve and sensitivity analyses, small-signal ac, noise and low-level distortion analyses, and large-signal nonlinear time domain transient analysis. SPICE2 was written for a CDC6400 computer with 40000 words of available memory. Increasingly, a large percentage of both the instructional and research use of the program is for the time-domain analysis of large, MOS transistor circuits. These analyses prove to be relatively expensive. Therefore it is worthwhile to identify the areas where most of the computational effort is being expended in the program, reducing this cost where possible. For large circuits, central memory size is also limited. Most computer systems have either a limit on maximum memory partition size or apply a surcharge to the analysis cost based on the amount of memory used.

SPICE2, since its release in 1975, has proven to be an effective integrated-circuit simulator. As expected, modifications and additions have been made since its release to improve its effectiveness, repair and correct inevitable bugs, etc. Further, we are faced with a continuing need for faster performance and the ability to handle larger circuits.

For economic reasons, there have to be tradeoffs between accuracy and circuit size, and between accuracy and computational speed. In addition, tradeoffs are necessary between the memory requirements necessary for new features and the ability to handle large circuits for a given computer capability.

In this paper a detailed breakdown of these factors is presented for SPICE2. Computational time that is required for different levels of nonlinear electronic device modelling is developed for both MOS and bipolar devices as well as models including table look-up schemes. The effects and savings of device-level bypass schemes are evaluated. Finally, the cost of extensive user oriented features now available in SPICE2 is described in terms of required memory and speed.

2. DATA STORAGE

SPICE2 provides built-in models for circuit elements which can be included. The parameters for all active device models may be modified during the input phase but in general the same model is used for many devices. For this reason the device storage requirements dominate the total input data storage. Table 1 contains a summary of the CDC6400 60-bit words required for typical devices. Each device requires a number of words to store a model pointer, device-dependent parameters, such as resistor value or a MOSFET channel length, and pointers to the locations in the Modified Nodal Analysis (MNA) matrix in which the device circuit-model elements are included. These pointers are used during the analysis to save the search time otherwise required to find the corresponding diagonal and off-diagonal entries in the matrix. The number of words used by each device for these functions is shown in the second column of Table 1.

element	device storage	four state vectors	total words
resistor	14	0	14
capacitor	12	8	20
inductor	14	8	22
voltage source	16	0	16
current source	11	0	11
BJT	35	56	91
MOSFET	45	88	133
JFET	30	52	82
DIODE	20	20	40

Outputs: words = (noutp+1)*numttp

Codgen : words = 2.5 *iops

Table 1 Storage Requirements in SPICE2D.8.

For the trapezoidal integration scheme used in SPICE2, four past iteration values of capacitor charges and inductor fluxes are used to estimate the local truncation error and compute the next timestep value. This information, together with past copies of device branch voltages, partial derivatives and other information is stored in the "state vector". The storage required for the four past state vectors used by each device is shown in the third column of Table 1 and the total storage requirements for each device is included in the last column.

The derivatives and other information mentioned above are used to reduce model evaluation time by allowing a device to be "bypassed" if its branch voltages have not changed significantly since the last Newton-Raphson iteration. In bypass, the partial derivatives and currents evaluated at the last iteration are simply reloaded into the MNA matrix and right-hand side vector. For digital circuits, this technique proves very valuable as is shown later. For MOSFETS the bypass scheme uses 32 words/device in the present implementation. This storage could be reduced to 8 words/device without altering program performance.

Experiments with SPICE2 have shown that truncation error estimation for timestep control is of limited value in digital circuits and that the alternative method of iteration count is often more effective [1]. With iteration count timestep control only three state vectors are required by the program in its present form and another 22 words/device are saved.

SPICE2 stores its outputs in central memory. The memory used for this storage is prohibitive for large circuits, e.g., a circuit with greater than hundreds of devices. The total number of storage words required is equal to the number of user-requested output variables (noutp) plus one, times the total number of converged time-points (numttp) as shown in Table 1.

The program also has the facility to generate loopless machine code for the factorization and solution of the MNA equations. As indicated in Table 1, the memory used for this code is directly proportional to the number of operations performed during the solution process.

3. EXAMPLE: THE BINARY-TO-OCTAL DECODER

Speed and memory tradeoffs that can be achieved in SPICE2 are illustrated based on the transient analysis of the MOS circuit shown in Fig. 1. This circuit can be implemented with 48 MOS devices and provides 35 equations in SPICE2. The transient waveform outputs for the benchmark run are shown in Fig. 2. The input excitations are chosen to keep a portion of the circuit changing at all times.

The central memory required by the program for this circuit is summarized in Table 2. The element and output storage requirements dominate the total data storage. The generated machine code does not produce a significant overhead in memory usage.

4. MODEL COMPLEXITY VERSUS SPEED

There are three MOS models available in SPICE2D.8. The most complex of the three (MOS3) is based on the formulation of ElMansy and Boothroyd [3]. The MOS2 model is faster to evaluate. It is based on the work of a number of authors and is described in detail in [4]. The simplest model, MOS1, is similar to the Shichman and Hodges model as implemented in SPICE1 [1]. Both MOS2 and MOS3 contain voltage-dependent capacitors to model thin-oxide charge storage while MOS1 contains only constant capacitors.

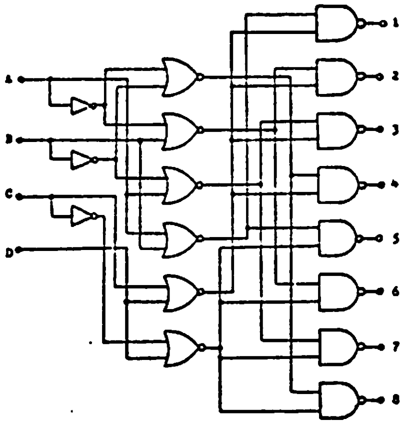


Fig. 1 Binary-to-Octal Decoder.

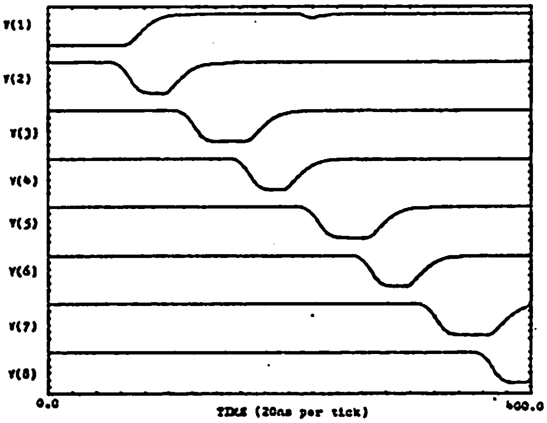


Fig. 2 Outputs of Binary-to-Octal Decoder for benchmark run.

		total words	percent
Elements	:	7500	58
Outputs	:	3200	25
Codgen	:	1100	8
Others	:	1200	9
Total	:	13000	100

Table 2 Storage requirements for Binary-to-Octal Decoder.

Table 3 shows the model evaluation time per device per iteration (cp/d/it) and the percentage of total time

model	cp/d/it (ms)	%cp load	it/tp averg	cp/pp ~~(ms)~~	cp/pp norm
GP	2.9	81	4.5	2.6*	12*
EM1	2.3	77	4.7	2.3*	10*
MOS3	9.8	88	·3.7	2.2	10
MOS2	4.3	79	3.1	1.3	6
MOS1	3.7	78	4.1	0.9	4
MOS0	0.24	21	4.6	0.22	1

*normalized by device count (48/88)
cp : central processor time for analysis
d : active device
it : Newton-Raphson iteration
tp : time point
pp : user requested print point

Table 3 Relative model performance for Binary-to-Octal Decoder.

per Newton-Raphson iteration spent evaluating the model and loading the MNA matrix entries. These models are "smart" in all of the sense that the bypass scheme is included consistently in the models.

The benchmark circuit has also been implemented using bipolar devices with both the Gummel-Poon (GP) and level-one Ebers-Moll (EM1) models available in SPICE2 [5] for comparison. It is evident that model evaluation and matrix loading dominates the total solution time, typically 80% of the total.

The results above prompt the inclusion of a table look-up model in the program similar to that used in MOTIS [6] and MOTIS-C [7]. Results using this model as implemented in MOTIS-C are included in Table 3 as MOS0. The results are for almost identical output with respect to the other models for this circuit and yet model evaluation time was less than 15 times that using MOS1. The speed increase is not entirely reflected in the run time however. The simpler models tend to have more abrupt switching points and transitions which increase the average number of iterations required per timepoint. This, in turn, increases the number of rejected timepoints. Both effects slow down the analysis slightly.

For MOS0, the model evaluation time is only 21% of the total analysis time. The other 79% is spend solving the matrix equations and computing the next timepoint. Since all of these analyses are made using the program-generated machine code, the matrix solution time is a very small percentage of the total time (less than 3 percent of the total time for MOS1). The remainder of the time is spent in the estimation of local truncation error and in overhead associated with the analysis.

5. ACCURACY VERSUS SPEED

SPICE2 uses both the relative change in active-device branch currents and node voltages, as well as their absolute change, to test for convergence of the Newton-Raphson iterations at a timepoint. In most cases it is the maximum permitted relative change, defined by the parameter RELTOL, which determines the point of convergence. If the number of iterations required to converge exceeds 10, the timepoint is unconditionally rejected and the timestep is cut. Otherwise an estimate of the local truncation error (LTE) incurred for capacitor currents and inductor voltages is computed and compared to the maximum permitted value. If this test fails the timestep is also cut. For stiff linear or "weakly nonlinear" circuits, the LTE stepsize control is essential to maintain reasonable accuracy. For highly nonlinear circuits, however, the iteration count stepsize control can provide comparable accuracy in voltage waveforms at much less cost.

A comparison of waveform accuracy and central processor time using both iteration count alone and iteration count with LTE, plotted against RELTOL, is shown in Fig. 3. Run time is normalized to the program default condition of LTE stepsize control with RELTOL = 0.1%. RELTOL is varied from its default value to 100% at which point the analysis is meaningless.

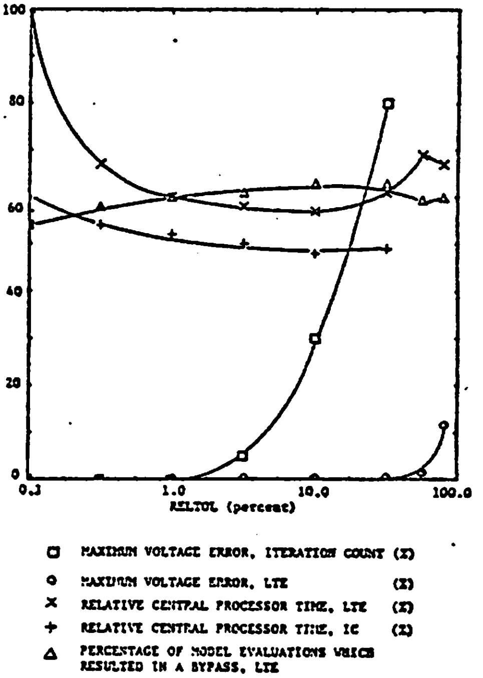


Fig. 3 Program performance for different convergence criteria.

These plots show that LTE is extremely conservative and can cost almost twice as much as iteration count on a large digital circuit. LTE maintains voltage waveform accuracy over a much wider range of values of RELTOL but with reasonable values of RELTOL the results are identical for the benchmark circuit of Fig. 1.

6. THE MODEL EVALUATION BYPASS SCHEME

As mentioned earlier, the nonlinear device models may not be evaluated at every iteration. If the change in device branch voltages is sufficiently small, the derivatives and currents evaluated at the previous timepoint will be loaded instead. Earlier experiments indicated that this scheme provided an average of only 4% saving in run time for a variety of circuits [1]. Our present studies indicate that for large digital circuits savings of over 40% are possible. Figure 3 includes a plot of the percentage of total model evaluations which resulted in a bypass, using LTE and the MOS2 model. These savings justify the extra memory required to implement this scheme.

If a much faster model evaluation scheme is used, such as table look-up, the advantage of bypass is significantly reduced. In fact, the time required to test for bypass may be comparable to the total model evaluation time.

7. SUMMARY

There are a number of areas where SPICE2 may be improved for the analysis of large, digital MOS circuits. The reduction of storage in the state vectors and the use of iteration-count timestep control reduce substantially the memory required for elements. The storage of outputs on disk, rather than in central memory, is also desirable for large analyses.

The machine code generated by the program does not require a substantial amount of memory and significantly improves run time if the device models can be evaluated cheaply.

In SPICE2D.8, model evaluation time still dominates total run time. Techniques such as table look-up need to be investigated further to provide a more uniform spread of computational effort while not degrading the solution accuracy. Finally, the bypass scheme used in SPICE2 is worthwhile for digital circuits. It provides substantial time savings for a relatively small penalty in memory requirements for cases where a model evaluation time is significant.

8. ACKNOWLEDGEMENTS

The authors wish to thank E. Cohen for helpful discussions and assistance with some of the programming details. L. Jensen and J. Crawford helped with many of the SPICE2 runs required to gather the statistics presented in this paper.

Research sponsored by the U.S. Army Research Office Grant DAAg29-76-G-0161 and the Hewlett Packard Company.

9. REFERENCES

[1] L. W. Nagel, "SPICE2: A Computer Program to Simulate Semiconductor Circuits," ERL Memo No. ERL-M520, Electronics Research Laboratory, University of California, Berkeley, May 1975.

[2] E. Cohen, "Program Reference for SPICE2," ERL Memo No. ERL-M592, Electronics Research Laboratory, University of California, Berkeley, June, 1976.

[3] A. R. Boothroyd and Y. A. El-Mansy, "A New Accurate Model of the IGFET for CAD Applications," International Electron Devices Meeting Technical Digest, pp. 31-34, Washington, D.C., 1974.

[4] D. R. Alexander, et al., "SPICE2 MOS Modeling Handbook," Report BDM/A-77-071-TR, BDM Corporation. Albuquerque, New Mexico, 1977.

[5] Ian Getreu, "Modeling the Bipolar Transistor," Tektronix, Inc., Beaverton, Oregon, March, 1976.

[6] B. R. Chawla, H. K. Gummel, P. Kozak, "MOTIS - An MOS Timing Simulator," Trans. IEEE, Vol. CAS-22, No. 12, pp. 901-910, Dec. 1975.

[7] S. P. Fan, M. Y. Hsueh, A. R. Newton and D. O. Pederson, "MOTIS-C: A New Circuit Simulator for MOS LSI Circuits," Proceedings, 1977 IEEE International Symposium on Circuits and Systems, Phoenix, Arizona, April 25-27, 1977.

APPENDIX 10

A Simulation Program with Large-Scale Integrated Circuit Emphasis

Presented at the 1978 IEEE International Symposium on Circuits and Systems, New York, New York, May 1978.

A SIMULATION PROGRAM WITH LARGE-SCALE INTEGRATED CIRCUIT EMPHASIS

A.R.Newton and D.O.Pederson

Department of Electrical Engineering and Computer Sciences,
University of California, Berkeley, Ca., 94720

Abstract: **SPLICE is a computer program for the simulation of large digital electronic circuits which combines circuit, timing and logic analyses into a single package. It provides detailed analog circuit simulation for critical parts of the network while signals between the circuit blocks may be processed using timing or logic analyses. All three types of analysis are performed simultaneously, while event control is used to enhance execution speed.**

1 INTRODUCTION

A number of simulation techniques are available for the analysis of electronic circuits. For small circuits where analog voltage levels are critical to circuit performance, or where tightly coupled feedback loops exist, a circuit simulator such as SPICE2 [1] can accurately predict circuit performance. As the size of the circuit increases, the cost and memory requirements of such an analysis become prohibitive. Fortunately, a large fraction of a typical LSI system is digital in nature. For this reason, certain simplifications may be made during the analysis which greatly increase execution speed and yet provide adequate information about circuit performance.

For circuits where a verification of the logical operation of the circuit and only first-order timing information is sufficient, a logic simulator may be used [2]-[4]. If dynamic charge-storage effects or bilateral circuit elements are important, or if a waveform analysis is required and the accuracy and expense of circuit simulation is not justified, a timing simulator can be used [5],[6].

A comparison of circuit, timing and logic analysis programs for the analysis of the same problem on the same computer [7] has shown MOTIS-C [6] to be typically two orders of magnitude faster than SPICE2, and SALOGS-3 [2] to be three orders of magnitude faster than SPICE2.

It is evident that for the analysis of large digital systems which contain tightly coupled circuit blocks or critical paths a simulator is required which will combine the accuracy of circuit simulation (for critical parts of the network) with the speed and memory-saving advantages of timing and logic simulation for the remainder of the circuit [8]. SPLICE (Simulation Program with Large-scale Integrated Circuit Emphasis) has been written with this goal in mind.

2 PROGRAM STRUCTURE

The block structure of program SPLICE is shown in Fig.1. The program has been partitioned into three distinct blocks which communicate with each other via mass-storage files. This permits the input and output processing modules of SPLICE to be implemented on machines other than the main computer. The input processor for SPLICE processes data with a syntax which is very similar to that used by program SPICE2. It is responsible for circuit macro expansion, satisfying back-referenced models, identifying element types and reducing arithmetic expressions wherever possible. The input processor produces a binary file to be read by the Setup and Analysis segment of the program. (This binary file could also be produced by programs such as direct circuit extraction from integrated circuit artwork data.)

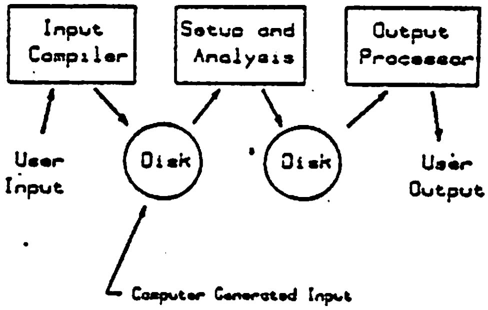


Fig. 1 - General Structure of SPLICE

Certain analysis constants and variables are initialized and the input file is read and processed. After reading the circuit description and analysis requests, the program performs a number of preprocessing operations to minimize the overhead which will occur during the analysis. This Setup Phase includes the generation of node fanin and fanout tables, the reduction of data required for each element to signal-path values and a model pointer, the compaction of

This work was sponsored by Hewlett Packard, Palo Alto, Ca.

node-to-ground capacitors [5], mock decomposition of the sparsely-stored circuit matricies to minimize and compute fill-ins, and the reduction of many indirect address chains to reduce the number of memory references performed during analysis. Most of the data used by SPLICE is stored in dynamically allocated arrays controlled by a memory management package included in the program. A number of temporary mass-storage files are used by SPLICE to reduce the maximum memory requirements of the program during the Setup Phase. These files are released prior to the analysis.

The circuit is then analyzed using an event scheduling scheme similar to that used in many modern logic simulators [9]. However, in contrast to a logic simulator the "events" scheduled in SPLICE may be any one of the three types: logic, timing or circuit. After the analysis of a scheduled block, any other blocks affected by its change of state are also scheduled for analysis. The storage of selected circuit variables on mass-storage as they change is also controlled by the schedular, as described in Section 3.1. If a circuit block exhibits no significant change in state over a period of time, it may be released from the event list just as a logic or timing block would be.

After the analysis the output file may be processed repeatedly by the output module of the program to produce waveform plots and logic output tables. This task may be performed on the host machine or the data may be shipped to an off-line intelligent terminal.

3 ANALYSIS ALGORITHMS

The program can perform a static analysis, which will allow for the propagation and solution of initial conditions, and a time-domain non-linear transient analysis for circuit node voltages and logic levels. The logic simulation uses four states: logic zero, logic one, undefined and high impedance. All logic elements may have different rise and fall delays, a necessity for MOS logic. As an event occurs, control is passed temporarily to the circuit, timing or logic analysis control module. Each block of the circuit uses a model control block (MCB) of data which contains either the information or the addresses of the information required during analysis.

To reduce the overhead imposed by the scheduling operation for a circuit analysis, the concept of minimum resolvable time (MRT) is used. One unit of MRT is both the minimum permitted non-zero gate delay of the logic simulation and the minimum time for which a circuit or timing block may be analyzed for the storage of changed values or propagation of events. For example, once a circuit analysis is scheduled, the circuit block is analyzed repeatedly until at least one unit of MRT has elapsed. In general, this should not require more than one solution unless one or more of the circuit variables is changing very rapidly at that time. Another advantage of MRT is that since piece-wise input sources are constrained to have their breakpoints at integer multiples of MRT a breakpoint table, such as is used in SPICE2, is not required.

One time consuming aspect of schedular operation is the searching of the event list to insert a new event. Since most events occur within 100 MRT units of the present event and because of the uniform discretization of time provided by the MRT algorithms, a selected number of events near the present event may be addressed directly. SPLICE provides direct addressing for 200 MRT units around the present event. This scheme implies a small penalty in storage but reduces the event access time during analysis [10].

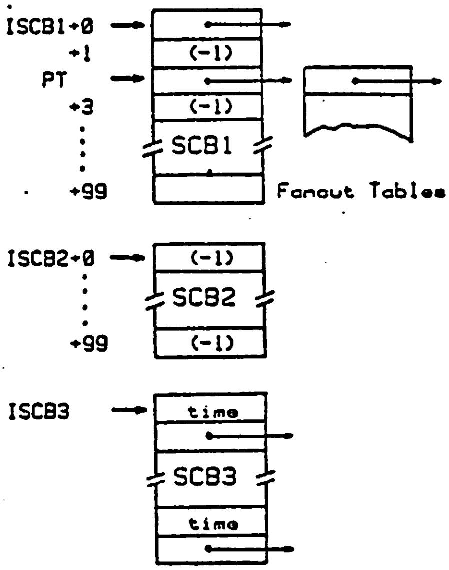


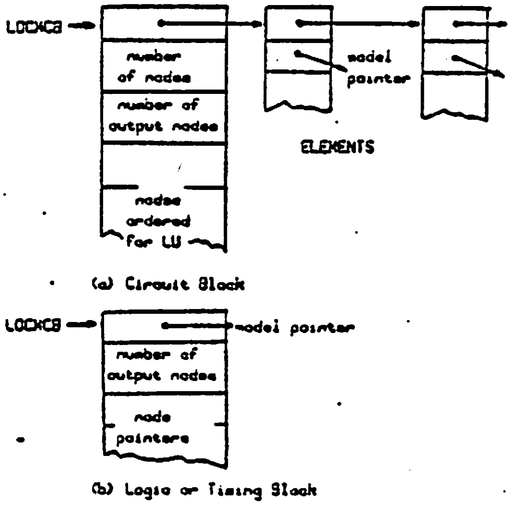
Fig. 2 - Structure of the Schedular Control Blocks

3.1 The Schedular

The structure of the schedular event list is shown in Fig.2. The event list is partitioned into three blocks. The first block SCB1 is addressed by ISCB1 and contains the event presently being processed as indicated by pointer PT (Present Time) in Fig.2. The entries in this block and in block SCB2 are separated in time by one unit of MRT. Multiple events at the same time are handled with a linked structure, as shown. Many of the MCB pointers in these blocks will be null (-1) for a relatively inactive network. Any events scheduled to occur outside the range of SCB1 and SCB2 are stored in block SCB3 and are tagged with th :hey are due to be processed. As PT moves down the list to ISCB1+99, pointers ISCB1 and ISCB2 are swapped, block ISCB2 is cleared and the entries in SCB3 are searched for inclusion in the new SCB2 block. Should the network become dormant, that is if SCB1 and SCB2 are empty, a provision is made to branch directly to the next change in input excitation which would be present in SCB3.

3.2 Model Control Block (MCB)

All network macro-models use an MCB during analysis. The structure of the MCB for a circuit block is shown in Fig.3(a). Examples of circuit blocks stored in this way are floating capacitors or resistors, operational amplifiers, or a number of MOS transfer gates connected to a single circuit node. The sparse circuit matrix is stored as a set of linked elements associated with the solution vector and the circuit MCB contains a set of pointers to these nodes, ordered such that the number of fillins generated during the matrix elimination processed is minimized. The circuit elements are stored as a linked list associated with the MCB. Elements stored in the element list contain pointers to directly address their matrix entry locations, as is done in SPICE2. These pointers are generated during the Setup Phase to minimize matrix load time during the circuit analysis. To further reduce the time overhead associated with finding the matrix locations for each element, the modified nodal analysis (MNA) matrix is only updated between Newton-Raphson iterations at a single timepoint if variable charge-storage elements are present in the block or the timestep has just been cut. Otherwise, the LU factors computed at the first iteration of the timepoint are used [11].



(a) Circuit Block

(b) Logic or Timing Block

Fig. 3 - Data Storage for Elements

The MCB for a typical timing or logic element is shown in Fig.3(b). The structure is very similar to the element storage structure for a circuit block. No parameters are stored with the elements, but rather a new model is generated for each different element type or device geometry. While this implies a storage penalty for small circuits, for large circuits where one type of device may be used many hundreds or thousands of times this approach results in significant storage savings. The device model block may contain a piece-wise linear look-up table, the parameters required for a simple analytical model, or the parameters required for a more complex model. For example an MOS transistor may be modeled using the table look-up technique employed in MOTIS-C, the simple Shichman-Hodges model of SPICE-1 [1] or the MOS2 model of SPICE2D [12]. These models are available for both the circuit and timing analyses.

3.3 Solution Vector and Fanout Tables

The structure of the solution vector and fanout tables is shown in Fig.4. For each node type, the output vector contains pointers to the node fanin and fanout tables, the node type (logic, timing, external circuit or internal circuit) and the last time at which a logic or voltage change occurred at that node. In all cases the fanout table contains the addresses of all elements which are driven by the node. Should the state of a node change significantly at any time, the fanout table is scheduled for processing and thus the effect of the change is propagated throughout the circuit.

For logic nodes the fanin pointer is only used if the logic node is a buss, that is if more than one logic element may determine the logical state of the node. In this case the fanin table is used to check for buss contention. The past two logic states of the node are packed into a single word at the end of the list.

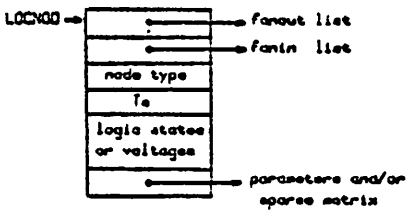


Fig. 4 - Node Data Structure

For timing nodes the fanin table contains the addresses of all elements which drive the node. These may include driver transistors, loads, transmission gates or floating capacitors. Grounded capacitors are treated as a special case to improve efficiency. The past two node voltages and the address of the node capacitance entry are stored at the end of the list. Timing node voltages are scaled and stored as integers with 30000 discrete levels of voltage spanning twice the range of the network power supplies.

Circuit blocks are treated as timing macromodels, communicating with the rest of the network via timing nodes. When a circuit block is scheduled, all timing nodes to which it is connected are processed to generate a Norton equivalent at the node. The circuit block, together with the current source and conductance equivalents at these external circuit nodes, is processed. Should internal or external circuit node voltages change significantly, the circuit block and its associated timing nodes are scheduled one unit of MRT ahead.

4 PROGRAM PERFORMANCE

The program has been used to simulate a number of circuits, so far the largest being a 600 MOSFET PLA circuit. For logic elements an average of 12 words per gate or 12 words per logic node are required. This includes the storage required for the element itself and its share of node, fanin and fanout table storage. The speed of a pure logic simulation is approximately 10us per gate per unit of MRT or 1ms per gate-event. For timing simulation approximately 8 words per device or 25 words per timing node are required. The speed of a pure timing simulation is typically 50us per MOS transfer device per unit of MRT. For both logic and timing simulations the execution times may vary up to an order of magnitude depending on the properties of the circuit being simulated. The time and memory requirements of a circuit simulation are variable but in general they are significantly lower than those of SPICE2D.

5 SUMMARY

SPLICE is a simulation program for large-scale digital electronic systems and can perform circuit, timing and logic analyses in parallel. While SPLICE is written for use on a CDC6400 computer, it was designed with use on a minicomputer or similar intelligent terminal in mind.

The program uses event-scheduling algorithms coupled with modern circuit analysis and timing simulation techniques to take advantage of some of the properties of large integrated circuits and enhance both program execution speed and data storage requirements. For logic analysis an average of 12 words per simple gate is required. Timing analysis results show an average of 8 words per element. For circuit blocks the storage requirements vary but are significantly less than those of SPICE2. Preliminary results indicate that SPLICE typically requires 10us per simple gate per unit of MRT for logic simulation and 50us per MOS device per unit of MRT for timing simulation on the CDC6400. A combined simulation would lie between these two limits.

6 REFERENCES

[1] L. W. Nagel, "SPICE2: A Computer Program to Simulate Semiconductor Circuits," ERL Memo No. ERL-M520, University of California, Berkeley, May 1975.

[2] G. R. Case, "SALOGS -- A CDC 6600 Program to Simulate Digital Logic Networks, Vol.1 - User's Manual," Sandia Lab Report SAND 74-0441, 1975.

[3] P. Wilcox and A. Rombeck, "F/LOGIC - An Interactive Fault and Logic Simulator for Digital Circuits," Proc. 13th ACM Design Automation Workshop, pp. 68-73, 1976.

[4] Several large-scale logic and fault analysis simulators are commercially available, such as CC-TEGAS3, D-LASAR, and LOGCAP.

[5] B. R. Chawla, H. K. Gummel, P. Kozak, "MOTIS - An MOS Timing Simulator," Trans. IEEE, Vol.CAS-22, No.13, pp.901-910, Dec.1975.

[6] S. P. Fan, M. Y. Hsueh, A. R. Newton and D. O. Pederson, "MOTIS-C: A New Circuit Simulator for MOS LSI Circuits," Proceedings, 1977 IEEE International Symposium on Circuits and Systems, Phoenix, Arizona, April 25-27, 1977.

[7] M. Y. Hsueh, A. R. Newton and D. O. Pederson, "New Approaches to Modeling and Electrical Simulation of LSI Logic Circuits," Journees d'Electronique, 1977, Lausanne, Switzerland.

[8] G. Arnout, H. De Man, "The Use of Threshold Functions and Boolean Controlled Network Elements for Macromodeling of LSI Circuits," Digest of Technical Papers, ESSCIRC'77, Ulm, F.R.G.

[9] S. A. Szygenda and E. W. Thompson, "Modeling and Digital Simulation for Design Verification and Diagnosis," Trans. IEEE, Vol.C-25, No.13, pp.1242-1253, Dec.1976.

[10] M. J. Flomenhoft, Private Communication

[11] A. R. Newton and G. L. Taylor, "BIASL.25 - An MOS Circuit Simulation Program for a Programmable Calculator," Proceedings Tenth Annual Conference on Circuits, Systems and Computers, Asilomar, California, 1976.

[12] "SPICE2 MOS Modeling Handbook," D. R. Alexander et al, Report BDM/A-77-071-TR, BDM Corporation, Albuquerque, New Mexico, 1977.

REFERENCES

[1] L. Nagel, "SPICE2: A computer program to simulate semiconductor circuits," Electronics Research Laboratory Report No. ERL-M520, University of California, Berkeley, May 1975.

[2] G. R. Case, "SALOGS — A CDC 6600 Program to Simulate Digital Logic Networks, Vol. 1 - User's Manual," Sandia Laboratory Report No. SAND 74-0441, 1975.

[3] G. Case, "The SALOGS Digital Logic Simulator," *Proc. 1978 IEEE International Symposium on Circuits and Systems,* New York, pp. 5-10, May 1978.

[4] P. Wilcox, A. Rombeck, "F/LOGIC — An Interactive Fault and Logic Simulator for Digital Circuits," *Proc. 13th ACM Design Automation Workshop,* pp. 68-73, 1976.

[5] S. A. Szygenda, "TEGAS2 — Anatomy of a General Purpose Test Generation and Simulation System for Digital Logic," *Proc. 9th Design Automation Workshop.*

[6] "LOGCAP-II User's Guide," National CSS, Inc., Norwalk, Conn.

[7] B. R. Chawla, H. K. Gummel, and P. Kozak, "MOTIS — An MOS Timing Simulator," *IEEE Trans. Circuits and Systems,* Vol. CAS-22, No. 12, pp. 901-

909, Dec. 1975.

[8] J. D. Crawford, M. Y. Hsueh, A. R. Newton and D. O. Pederson, "MOTIS-C User's Guide," Electronics Research Laboratory, University of California, Berkeley, June 1978.

[9] M. Y. Hsueh, A. R. Newton and D. O. Pederson, "New Approaches to Modeling and Electrical Simulation of LSI Logic Circuits," *Journees d'Electronique,* 1977, Lausanne, Switzerland.

[10] G. Arnout and H. J. DeMan, "The Use of Threshold Functions and Boolean-Controlled Network Elements for Macromodeling of LSI Circuits," *IEEE J. Solid-State Circuits,* vol. SC-13, pp. 326-332, June 1978.

[11] E. Cohen, "Program Reference for SPICE2," Electronics Research Laboratory Report No. ERL-M592, University of California, Berkeley, June 1976.

[12] SCEPTRE, Air Force Weapons Laboratory Technical Report AFWL-TR-69-77, Kirkland AFB, NM, 1969.

[13] W. T. Weeks *et al,* "Algorithms for ASTAP — A Network Analysis Program," *IEEE Trans. Circuit Theory,* vol. CT-20, pp. 628-634, Nov. 1973.

[14] G. R. Boyle, "Simulation of Integrated Injection Logic," Electronics Research Laboratory Report No. ERL-M78/13, University of California, Berkeley, March 1978.

[15] M. Y. Hsueh, A. R. Newton and D. O. Pederson, "The Development of Macromodels for MOS Timing Simulators," *Proc. 1978 IEEE International*

Symposium on Circuits and Systems, New York, pp. 345-349, May 1978.

[16] L. O. Chua, P. M. Lin, *Computer Aided Analysis of Electronics Circuits: algorithms and computational techniques,* Prentice Hall, New Jersey, 1975.

[17] C. W. Ho, A. E. Ruehli, P. A. Brennan, "The Modified Nodal Approach to Network Analysis," *Proc. 1974 IEEE International Symposium on Circuits and Systems,* San Francisco, pp. 505-509, April 1974.

[18] D. R. Alexander *et al,* "SPICE2 MOS Modeling Handbook," Report BDM/A-77-071-TR, BDM Corporation, Albuquerque, New Mexico, 1977.

[19] A. Vladimirescu, E. Cohen and D. O. Pederson, "SPICE2 User's Guide," Electronics Research Laboratory, University of California, Berkeley, April 1978.

[20] M. Y. Hsueh and D. O. Pederson, "An Improved Circuit Approach for Macromodeling Digital Circuits," *Proc. 1977 IEEE International Symposium on Circuits and Systems,* Phoenix, pp. 696-699, April 1977.

[21] L. O. Chua and L. K. Chen, "Diakoptik and Generalized Hybrid Analysis," *IEEE Trans. Circuits and Systems,* vol. CAS-23, No. 12, pp. 694-705, Dec. 1976.

[22] F. F. Wu, "Solution of Large-Scale Networks by Tearing," *IEEE Trans. Circuits and Systems,* vol. CAS-23, No. 12, pp. 706-713, Dec. 1976.

[23] G. Guardabassi and A. Sangiovanni-Vincentelli, "A Two Levels Algorithm for Tearing," *IEEE Trans. Circuits and Systems,* vol. CAS-23, No. 12, pp. 783-791,

Dec. 1976.

[24] A. Sangiovanni-Vincentelli, L. K. Chen and L. O. Chua, "An Efficient Heuristic Cluster Algorithm for Tearing Large-Scale Networks," *IEEE Trans. Circuits and Systems,* vol. CAS-24, No. 12, pp. 709-717.

[25] A. E. Ruehli, N. B. Rabbat and H. Y. Hsieh, "Macromodular Latent Solution of Digital Networks Including Interconnections," *Proc. 1978 IEEE International Symposium on Circuits and Systems,* New York, pp. 515-521, May 1978.

[26] A. E. Ruehli, R. B. Rabbat, H. Y. Hsieh, "Macromodelling - an approach for analysing large-scale circuits," *Computer Aided Design,* vol. 10, No. 2, pp. 121-129, March 1978.

[27] G. Kron, *Diakoptics — Piecewise Solution of Large-Scale Systems,* MacDonald, London, 1963.

[28] N. B. Rabbat and H. Y. Hsieh, "A Latent Macromodular Approach to Large-Scale Sparse Networks," *IEEE Trans. Circuits and Systems,* vol. CAS-23, No. 12, pp. 745-752, Dec. 1976.

[29] F. H. Branin, "The Relation between Kron's Method and the Classical Methods of Network Analysis," *Matrix and Tensor Quarterly,* vol. 12, No. 3, pp. 69-115, March 1962.

[30] A. Ralston, *A First Course in Numerical Analysis,* McGraw Hill, New York,

1965.

[31] S. Seshu and D. N. Freeman, "The Diagnosis of Asynchronous Sequential Swithching Systems," *IRE Trans. Elec. Comp.*, vol. EC-11, No. 4, pp. 459-465, August, 1962.

[32] S. A. Szygenda and E. W. Thompson, "Modeling and Digital Simulation for Design Verification and Diagnosis," *IEEE Trans. Computers,* vol. C-25, No. 12, pp. 1242-1253, Dec. 1976.

[33] M. J. Flomenhoft and B. M. Csensitis, "Minicomputer-Based Logic Circuit Fault Simulator," *Proc. 11th ACM Design Automation Workshop, 1974.*

[34] M. M. Markowitz, "The Elimination form of the Inverse and its Application to Linear Programming," *Management Science,* vol. 3, pp. 255-269, April 1957.

[35] C. N. Ahlquist, J. R. Breirogel, J. T. Koo, J. L. McCollum, W. G. Oldham and A. L. Renninger, "A 16384-Bit Dynamic RAM," *IEEE J. Solid-State Circuits,* vol. SC-11, No. 5, pp. 570-574, Oct. 1976.

[36] This program was developed by L. Scheffer, Corporate Engineering, Hewlett-Packard, Palo Alto, Ca.

[37] The version of MOTIS-C used here has been modified for efficient execution on the HP 3000 computer by L. Scheffer, Corporate Engineering, Hewlett-Packard, Palo Alto, Ca.

[38] H. Schichman and D. A. Hodges, "Modeling and Simulation of Insulated-Gate Field-Effect Transistor Switching Circuits," *IEEE J. Solid-State Circuits,* vol.

SC-3, pp. 285-289, Sept. 1968.

[39] A. S. Grove, *Physics and Technology of Semiconductor Devices,* Wiley, New York, 1967.

[40] E. B. Kosemchak, "Computer Analysis of Digital Integrated Circuits by Macromodeling", Ph.D. thesis, Columbia University, USA, 1971.

[41] N. B. Rabbat, "Macromodelling and Transient Simulation of Large Integrated Digital Systems," Ph.D. thesis, The Queen's University of Belfast, UK, 1971.