

EMOTA: an event-driven MOS timing simulator for VLSI circuits

W.-Z. Shen
S.-J. Jou
Y.-S. Tao

Indexing terms: Computer simulation, Algorithms, Very large scale integration

Abstract: A novel new event-driven MOS timing simulator for VLSI circuits, EMOTA, is presented. The traditional event-driven simulation scheme used in logic simulation is modified for use in this circuit level timing simulator. The fast performance is due to the result of mixing the new derived event-driven algorithm and its associated time-wheel control mechanism, together with the unidirectional nonlinear Gauss-Seidel relaxation technique to decouple the circuit equations. EMOTA allows both interactive and batch simulation modes and its input format is the same as SPICE. The simulation speed of EMOTA is shown to be more than 300 times faster than SPICE2G.5 for circuits of several hundred transistors and the simulated waveforms have acceptable accuracy.

1 Introduction

General purpose circuit simulators, such as SPICE2 [1] and ASTAP [2], have been widely used for the circuit analysis of integrated circuits. These simulators can provide DC, AC and time-domain *transient* analyses of a wide range of circuits. Among these, time-domain transient analysis is the most commonly used analysis performed by circuit simulators, but it is also the most time-consuming one. Since the formation and solution of the circuit equations are most expensive in terms of the computer time, especially when the circuit size exceeds a certain range. Therefore, it is very inefficient to simulate large circuits using these traditional circuit simulators.

To enhance the speed of simulation when large circuits are simulated, many timing simulators such as MOTIS [3, 4], MOTIS-C [5], SPLICE [6], RELAX [7], ADEPT [8] and MOTA [9] have been developed based on *relaxation electrical simulation technology* (REST) [10]. These are special purpose simulators that deal with MOS circuit transient analysis and, basically, the relaxation techniques were used to decouple nodal equations to avoid solving the large sparse matrix, while still maintaining acceptable waveform accuracy. Because of the decoupling of the model equations, for logic gate circuits the time required at the solving step grows only linear with the number of nodes.

For most circuits, the fraction of nodes that are changing their voltage value at a given point in time decreases as the circuit size increases. For logic or circuit simulators, many schemes have been used to exploit this *time sparsity* or *latency* to further enhance the simulation of large electrical circuits [11–22].

In this paper, we describe EMOTA, an event-driven MOS timing simulator. The *Gauss-Seidel-Newton* relaxation simulation technique is used in EMOTA because of its better convergence properties [10]. Considering the property of convergence speed and the behaviour of MOS circuits, we partition the circuit into simple logic gate blocks and tightly-coupled and/or bidirectional circuit blocks. Thus, the signal propagation between logic gate and subcircuit block is unidirectional and, hence, we call this the unidirectional *Gauss-Seidel* relaxation simulation technique. For completeness of this paper, the whole simulation technique will be shortly discussed in Section 2.

The employment of an *event-driven, selective trace* technique that is usually used in logic simulators [13–15] can provide a major time saving in logic simulation. This technique views a whole state changing as event and processes the events dynamically without the need to check or simulate each node for activity. Because EMOTA uses the unidirectional Gauss-Seidel relaxation technique, the signal propagation between circuit blocks is like that of logic simulation. Thus EMOTA combines the unidirectional Gauss-Seidel relaxation technique and the concept of an event-driven technique into a novel new timing event-driven technique. This new method is quite different from the traditional one because in a timing simulator, not only the state, but the whole transient waveform, must be calculated. A special time-wheel control mechanism used to deal with dynamic events can simplify the control process. The event-driven algorithm and the time-wheel mechanism will be discussed in Sections 3 and 4, respectively.

Although there has been a need for theoretical and practical investigation in all of the algorithmic aspects used in the simulator, there is also a need to concentrate on the overall software system. So the whole program structure, data structure, methodology and I/O processing will be discussed in Section 5. Finally, the simulation results and comparisons with other simulators will be discussed.

2 Unidirectional Gauss-Seidel relaxation technique

To perform the transient analysis of MOS digital integrated circuits, the circuit equations can be written in the

Paper 7248G (E10), first received 30th May 1989 and in revised form 18th October 1989

The authors are with the Institute of Electronics, National Chiao Tung University, 45 Po-Ai Street, Hsinchu, Taiwan, Republic of China

following form using the modified nodal analysis [16] with the assumption that the inductance effect in the circuit is ignored and each node has a capacitor connected to reference node:

$$C(V(t), U(t))\dot{V}(t) = -f(V(t), U(t)) \quad 0 \leq t \leq T$$

with initial condition,

$$V(0) = V_0 \quad (1)$$

where $V(t)$ and $\dot{V}(t) \in R^n$ are the vectors of node voltage and its time derivative, respectively, and $U(t) \in R^n$ is the vector of the input voltage source at time t . $C(V(t), U(t)) \in R^n * R^n$ represents the nodal capacitance matrix. $f: R^n * R^n \rightarrow R^n$ can be expressed as

$$f(V(t), U(t)) = [f_1(V(t), U(t)), f_2(V(t), U(t)), \dots, f_n(V(t), U(t))]^T$$

where $f_i(V(t), U(t))$ is the sum of the currents charging the capacitors connected to node i .

To solve eqn. 1, a simulatable MOS timing model is derived [12] which uses the following techniques:

- (a) A trapezoidal algorithm to discretise the derivative operators,
- (b) The nonlinear Gauss-Seidel relaxation technique to decouple the circuit equation,
- (c) The Newton-Raphson method with one unknown to linearise the nonlinear elements.

After the above techniques are applied, the capacitor is replaced by its companion model and the MOS device is replaced by a Norton equivalent model. The interested reader may refer to Reference 12 for a detailed description of the Gauss-Seidel-Newton properties.

In MOS digital circuits, a commonly used circuit structure is a complex driver-load logic gate. This consists of a collection of MOS transistors that are connected in series-parallel combinations. For MOS devices, if the gate-drain and the gate-source capacitances are assumed to be ignored, the gate can represent an almost unidirectional node. Thus driver-load logic gates are unidirectional elements with one output whose voltage is controlled by several inputs. The behaviour of the output is a function of the topology of the circuit and the input voltages that control the transistors in this topology. For this type of circuit, the convergence speed is fast and accurate voltage solutions can be obtained by one Gauss-Seidel-Newton iteration. But, for circuits containing tightly coupled feedback loops or bidirectional elements, such as pass transistors, transmission gates and floating capacitors, strong coupling can cause severe inaccuracy and even instability during nonlinear relaxation analysis [9,10]. Thus, from the simulation point of view, for a circuit containing tightly coupled feedback loops and/or bidirectional elements, the relaxation method is not a robust technique.

A subcircuit technique [9] is provided by EMOTA to simulate circuits containing tightly coupled feedback loops and bidirectional elements without the occurrence of severe inaccuracies. The circuit block is treated with a direct matrix solving technique using the same simulatable timing model as above. After the tightly coupled circuits are all grouped into subcircuit blocks, the whole circuit is partitioned into sets of logic gates and subcircuit blocks only. The combination of the subcircuit technique and the nonlinear Gauss-Seidel relaxation technique is referred to as the unidirectional Gauss-Seidel relaxation technique. In this technique, signal propaga-

tion is unidirectional, i.e. from input nodes to output nodes of logic gates and subcircuit blocks like the one-way macromodelling concept in Reference 17. This technique is well suited to the event-driven control mechanism because the circuit is partitioned into blocks and thus the signal propagation between blocks in the circuit simulation is the same as that in the logic simulation. Thus, to enhance the simulation speed, the implementation and evaluation of these blocks must be simple and easy to store and move so that the processing overhead of the event is small and the time saving is large when simulating large circuits.

3 Time-wheel, event-driven control scheme

3.1 Event and selective trace

For logic simulation, *selective trace* is a technique based on the observation that if the output of an element does not change when the inputs change, then the fanouts of that element are not affected by the excitation of the input signals. Hence, the output of the current element is not evaluated and the signals stop propagating along the trace. Based on this technique, an *event* is said to have occurred when an output line of an element takes on a new value, or logic value. The new value may or may not be the same as the value already held on the output line [13]. Thus an element is triggered only when its inputs are ready to evaluate the outputs of the element. If the new value is the same as the value already held by the output line, the selective trace technique comes into play and the event is cancelled. However, if the new value is different from the value already held by the output line, the event will be propagated to the fanout elements.

Although the time-wheel, event-driven control scheme has long been used in logic simulators [13], it is not used in traditional circuit simulators because the whole circuit is solved in a matrix structure. On the other hand, the unidirectional Gauss-Seidel relaxation technique has the potential to use this control scheme because of the decoupling of circuit equations and because the signal propagation between blocks is unidirectional. To incorporate this scheme into EMOTA, the events at the circuit level must be carefully defined, and the propagation of events is much more complex than that in a logic simulator.

In the transient analysis of circuit simulation, we need to calculate the nodal voltage waveform instead of the logic state only, so the event cannot be defined to be the same as that in logic simulation. In EMOTA, an *event* is the change of node voltages between two successive simulation time steps of an element during the transient. So an event associated with an element can occur not only with change of voltages on the input nodes, but also with changes of internal and output nodal voltages of the element at any transient time step. For each element, whenever the voltage at any one of the input nodes starts changing, the effect of this change at the element must be traced as it propagates to its fanout elements. Only the elements that are affected directly by this change will be processed; this technique is *selective* and, hence, this is the selective trace technique for EMOTA.

3.2 New event-driven control scheme

SPLICE [6] and SAMSON [18] have proposed the event-driven technique, but the method proposed in SAMSON is at the matrix solving level which is different from the relaxation level used in EMOTA. SPLICE may be the first relaxation based simulator to propose the

event algorithm at relaxation level. This algorithm, shown in Fig. 1, has the drawback that, the elements are not only triggered by events created by itself, *self events* (determined by local time step control), but also by events created by the elements connected to its input nodes, *input events*. Therefore the simulation time step is limited not only by the local truncation error (LTE) of the element, but also by the LTE of the elements connected to its input nodes. So the events of the element may be created twice during a simulation time step, or created more often than is necessary. This drawback can be eliminated if once the element is initially activated by the *input events*, it is then triggered by itself, that is, triggered by the *self events* and ignores successive *input events*. Thus the simulation time step during the transient is determined only by the LTE of the element. The new timing event-driven algorithm used in EMOTA is listed in Fig. 2.

```

/*E(tn): event list at time step tn */
tn = 0;
WHILE(tn < TSPOP){
  k ← 0;
  WHILE(event list Ek(tn) is not empty){
    FOREACH(i in Ek(tn)){
      Vik+1 = Vik -  $\frac{g_i \tilde{V}^{k+1,i}}{g_i \tilde{V}^{k+1,i}}$ 
      where  $\tilde{V}^{k+1,i} = [V_1^{k+1}, \dots, V_i^{k+1}, V_{i+1}^k, \dots, V_n^k]^T$ 
      IF(|Vik+1 - Vik| ≤ ε; i.e. convergence is achieved){
        use LTE to decide the next time ts for processing node i;
        add node i to event list Ek(ts);
      }
      ELSE{
        add node i to event list Ek+1(ts);
        add the fanout nodes of node i to event list Ek(ts) if they
        are not already on Ek(ts);
      }
    }
    Ek(tn) ← Ek+1(ts); Ek+1(ts) ← empty;
    k ← k + 1;
  }
  tn ← tn+1
}

```

Fig. 1 SPLICE event-driven algorithm

```

/*element :single gate block and subcircuit block*/
/*E(tn) :event list at time step tn */
tn = 0
1 WHILE (event list E(tn) is not empty){
  FOREACH (element i in E(tn)){
  2 solve element i;
  Set element i to inactive;
  FOREACH (output node j of i){
  3 IF (|Vj(tn) - Vj(tn-1)| > doc){
  Set element i to active;
  4 IF (j is not active) {
  Add each inactive fanout element k
  of node j to E(tn);
  Set k and j to active;
  }
  }
  5 IF (element i is not active) {
  Set nodes of element i to inactive;
  }
  ELSE {
  6 Use LTE to determine next time step, ts;
  7 add element i to E(tn + ts);
  }
  }
  tn = tn+1
}

```

Fig. 2 Unidirectional nonlinear Gauss-Seidel relaxation event-driven algorithm

In this algorithm, the event list $E(t_n)$ is used to schedule the events of the elements that are to be processed in time t_n (step 1) and an element is initially activated only by the input events (step 4). Once an element is to be activated, at time t_n , the event for this element will be scheduled into the event list $E(t_n)$. Then this element will be solved at time t_n and if any one of the output node voltages changes after calculation (including the internal nodes of subcircuit), this element is then activated by itself (step 3) and the LTE is used to determine the next time step t_s (step 6). The old event is cancelled and a new event associated with this element will be added into the event list $E(t_n + t_s)$ (step 7). Thus events are propagated and the elements that are already activated will not be activated again.

If all the output nodes and internal nodes of the active element stay unchanged, the old event is cancelled and the element is set to be inactive. Thus, no events propagate through this element and unnecessary computation and checks are avoided. In EMOTA the voltage change is recognised if it is greater than a specified value d_{oc} . The value of d_{oc} must be carefully and properly chosen to obtain an accurate simulation result and a fast simulation speed. It has been found experimentally that the value of 0.0005 V for d_{oc} gives the best result.

During a change of logic level, there may be tens of events occurring depending on the limitation of the LTE instead of one, as in logic simulation. To successfully use this new timing event-driven control scheme with the unidirectional Gauss-Seidel relaxation technique, some critical points must be investigated.

(i) Before the transient analysis, *initial events* must be put into the time-wheel.

Initial events are the events triggered by the excitation of input voltage sources. For example, a circuit, input sources and the associated initial events are shown in Fig. 3a. Before simulation, the *initial events* are all put into the time wheel to avoid checking the input source for the initial events at each simulation time step. For logic simulation, the above arrangement is satisfactory because each change of logic level needs only one event evaluation, but for circuit simulation, there is one problem. When the initial event occurs, some input may change very slowly so after one time step calculation, the output voltage may not change or the change may be below d_{oc} . In this case, the event will not propagate and the block will not activate itself while actually this block should activate itself and the event should propagate after the input has completely changed. To solve this problem we use the second scheme.

(ii) For initial events, it is divided into ten events, each one is separated by one tenth of the duration.

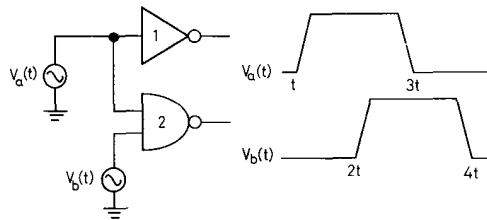
To have a safe simulation, EMOTA does not allow stepped input voltage sources. For inputs that are connected to the outputs of other blocks, the events will be missed only when a fast changing stage is driving a slow changing state. This occurs when the input voltage is stable and the change of output voltage during the whole input transient is less than d_{oc} (0.0005 V). In this case, the transient time of the second stage is almost 10000(5 V/0.0005 V) times longer than the first stage. In practical circuit designs, this is not advisable and some buffer stages are inserted.

(iii) There are three conditions under which a block may have more than one event associated with it during a transient:

(a) The ten *initial events* and the *self events* activated by the block,

(b) When the input source has been stable after a transient and then changes its value again, but the block is still in the activated condition due to previous changes as illustrated at time $2t$ in Fig. 3b.

(c) A block has more than one input changing at the same time, as in Fig. 3c.



Initial events	
Time	Block
t	1 and 2
$2t$	2
$3t$	1 and 2
$4t$	2

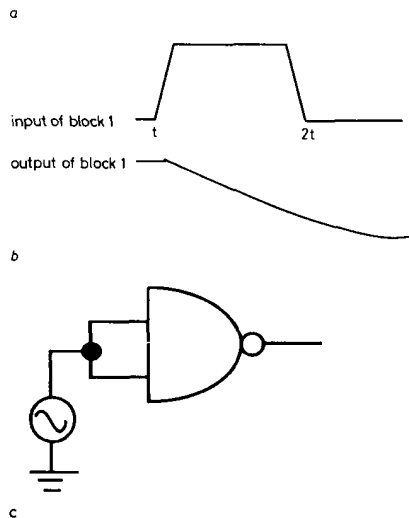


Fig. 3 Some special conditions of event-driven control

When a block has more than one source of events scheduled in the time wheel during a transient, if each one activates itself then there will be too many events associated with this block, thus wasting simulation time. For conditions (a) and (b), we use an active flag ($active_f$) to check whether there are too many events. For a block that is inactive, $active_f = 0$, and if it is activated by the input sources, $active_f = 1$; otherwise it is activated by itself and $active_f = 2$. Thus, when the program meets an event with $active_f = 1$, but the block is already activated ($active_f = 2$) the program skips the initial events. For condition (c), there are two events for the same block in a single time step. To avoid an erroneous simulation (calculating the output voltage twice in a time step), we add a $local_count$ variable to each block and a global counter is increased by one when the time-wheel grid advances one step. If a block has been simulated in this time-wheel grid, the program will set $local_count$ of this

block to the value of the global counter. Thus, when another event in the same time grid is encountered, because the global counter equals $local_count$, the program will skip this event.

This new timing event-driven algorithm is listed in Fig. 2. The three schemes that efficiently and correctly control the signal propagation in an MOS circuit as the simulation proceeds will be discussed in Section 6. As we have mentioned, the number of events in a circuit simulation are greater than in logic simulation. So the scheduling and controlling of events must be well arranged to reduce event operations. EMOTA uses a new time-wheel control mechanism to process the scheduling of the events in an efficient manner. This time flow mechanism will be discussed in next Section.

4 Time-wheel mechanism

Three types of event control schemes have been proposed, namely, fixed time increment, next event and hybrid time flow mechanism [13]. The hybrid time flow mechanism is adopted in most modern simulators. In this mechanism, a double-size time queue TQ and one overflow event list (sometimes called a macro time event list, MTEL) are used. A detailed description can be found in Reference 13. In EMOTA, a multiloop time wheel it utilised instead of the TQ and MTEL to achieve efficiency in operation and storage.

The time wheel is like a circle with 50 grids in size, as shown in Fig. 4a. Each grid in the wheel represents a minimum time unit (TI) used in EMOTA and the wheel is moved forward by this fixed time increment. TI is chosen to be 0.02 ns for accurate simulation results. The variable $WI(0 \leq WI < 50)$ indicates which grid in the wheel is currently being processed. Each grid in the time wheel contains a pointer to a list of events that occur at this instant of time. For example, if an event is determined to occur five time units later by current evaluation, this event will be linked to the event list in grid $WI + 5$.

As for wheel size, a large wheel size will occupy too much space, but if a small wheel size is used, too many events will be scheduled to a grid number that is larger than the wheel size. Because in MOS integrated circuits,

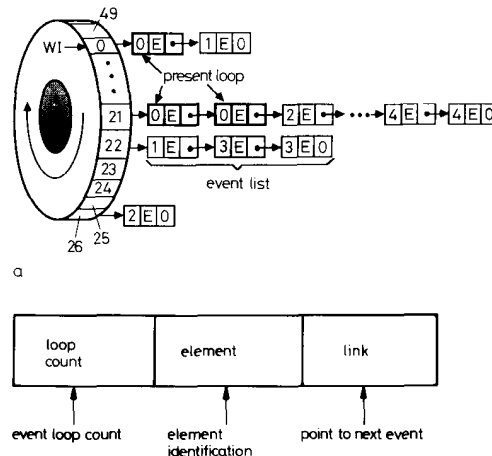


Fig. 4 The time wheel
a Time-wheel mechanism total wheel size = 50; time increment = 0.02 ns
b Data structure of an event

for simulation technique used (LTE limitation and convergence condition requirement), during the transient, the time step is usually less than 1 ns. Thus, the wheel size is chosen to be 50 to save memory space. Since the wheel size is 50, some events (like initial events, etc.) will be scheduled to a grid number (NO) which is larger than the wheel size. Instead of using one overflow event list, to solve this problem, the number $NO + WI$ is divided by 50, the residue is the grid number where the event is to be added to. The quotient is recorded in the first entry of the event data structure as a loop count. The data structure and the event of this multiloop time wheel is shown in Fig. 4b. The second term of the data structure indicates the element associated with this event. The last term is a link that points to the other events in the same time unit to form the event list.

Let WLC be a variable denoting the number of cycles that have been made through the time wheel. When each grid of the event list is traversed, the events whose loop count is zero are processed and then deleted. As for events whose loop count is not zero, the loop count is decreased by one and the events remain in this time grid. When the *Link* points to *NULL*, it means all the events in the current time grid have been traversed. Then the variable WI is increased by one, i.e. the next grid is now

to be processed. Whenever the variable WI equals 50, it is reset to zero and the variable WLC will be increased by one. Thus the simulation time is always equal to the expression $((50 * WLC) + WI) * T1$.

5 Overall program and data structure

EMOTA is implemented on a SUN workstation and written in C. The overall program flowchart is shown in Fig. 5. The input format of EMOTA is the same as for SPICE, except for two differences. In EMOTA, the user must define the global nodes, such as the reference node and power node, by using the *. Global card; numbers and letters can be used to specify a node while in SPICE, a node must be specified as a number. As for the MOS model, the level 1 and 2 models are the same as for SPICE2G.5 and level 4 is an I&G table lookup model [21]. The data structure for a logic gate and subcircuit, and the partitioning scheme, will be discussed in more detail in the following Section.

5.1 Logic gate data structure and evaluation

In EMOTA, the user can define the driver-load CMOS gate as shown in Fig. 6 using a .SUBCKT card to define a logic gate block. Such a transistor structure can be

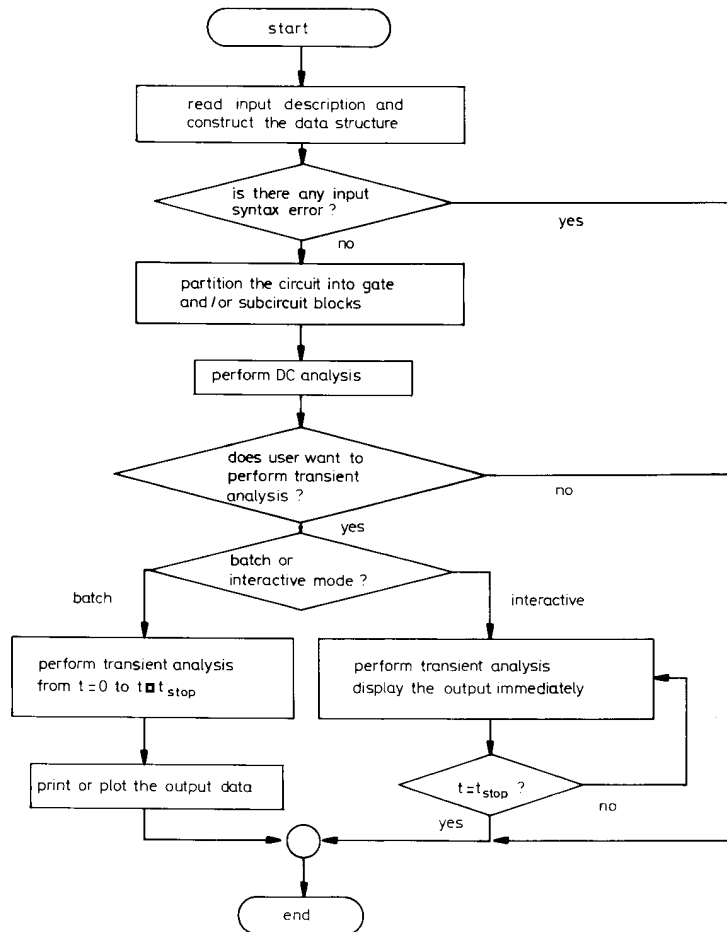


Fig. 5 Main program flowchart of EMOTA

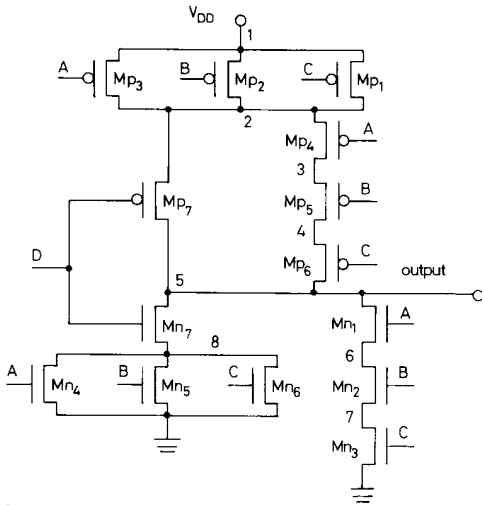
stored as an evaluation tree. This tree structure is suitable for this series/parallel configuration [19]. The internal tree data structure, used in EMOTA, for the gate in Fig. 6 is shown in Fig. 7.

```

.SUBCKT GEXAMPLE A B C D OUTPUT 1
Mp1 2 C 1 1 pMOS
Mp2 2 B 1 1 pMOS
Mp3 2 A 1 1 pMOS
Mp4 3 A 2 1 pMOS
Mp5 4 B 3 1 pMOS
Mp6 5 C 4 1 pMOS
Mp7 OUTPUT D 2 1 pMOS
Mn1 OUTPUT A 6 0 nMOS
Mn2 6 B 7 0 nMOS
Mn3 7 C 0 0 nMOS
Mn4 8 A 0 0 nMOS
Mn5 8 B 0 0 nMOS
Mn6 8 C 0 0 nMOS
Mn7 5 D 8 0 nMOS
.ENDS GEXAMPLE

```

a

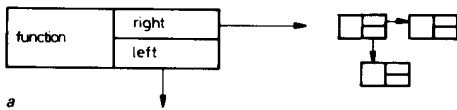


b

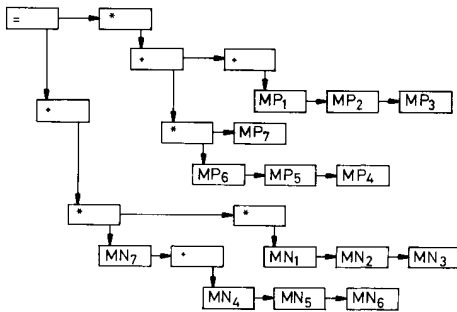
Fig. 6 Defining a driver-load gate

a Circuit description

b Circuit diagram



a



b

Fig. 7 Representation of data in EMOTA

a Basic node structure

b Tree data structure

As shown in Fig. 7, each node structure consists of three parts: the function of the node, the left node pointer and the right node pointer. All the nodes are linked to the tree structure according to the circuit connections. The root node, indicated by '=', consolidates the results from both the driver and the load subtrees. Each subtree can be further broken down into series/parallel connections. The function of the parent node shows that the child nodes linked to the left pointer are in parallel or in series by the '+' and '*' operators, respectively. The MOS transistors are always at the leaf node indicated by 'M'. For example, Mn1, Mn2 and Mn3 are in series; Mp1, Mp2, and Mp3 are in parallel. Using this data structure, once an event occurs at the input nodes of the logic gate, only the pointer of '=' needs to be scheduled and processed.

The logic gates are simulated using macromodels for those sections of the circuit whose connectivity is well understood in the tree data structure. The macromodel, which is based on a large-signal linear approximation of the transistors in the driver-load gate [20], enables us to compute the output voltage of the gate in an efficient manner. For example, in EMOTA, to evaluate the output voltage waveform of a driver-load gate, first, the equivalent models are applied to each MOS device and output capacitor. According to the macromodel, as the driver or load subtree is traversed, the current (conductance) for a parallel transistor section is computed as the sum of its constituent currents (conductances); for a series section, the reciprocal of the current (conductance) is obtained as the sum of the reciprocal of the constituent currents (conductances) [20]. To traverse the tree data structure, a depth-first traversal algorithm is used, as shown in Fig. 8. Each node in the tree data structure will be traversed to yield the equivalent current and conductance via the technique described previously. Finally, the driver (or load) transistor of the final equivalent circuit is viewed as the only transistor between the output node and V_{DD} (or

```

GLOBAL VARIABLE :level_stack
LOCAL VARIABLE :t_level,p

```

/* at initial, level = 0, p = root node */

TRAVERSE (p)

```

{
  t_level = level;
  while (p -> left != NULL){
    PUSH p -> stack_B;
    level = level + 1;
    p = p -> left;
  }
  CALCULATE current I and conductance G;
  PUSH I and G -> stack[level];
  REPEAT:
  while (p -> right != NULL){
    p = p -> right;
    if (p -> function == "M"){
      CALCULATE current I and conductance G;
      PUSH I and G -> stack[level];
    }
    else
      CALL TRAVERSE(p);
  }
  POP stack_b -> p;
  level = level - 1;
  PERFORM p -> function;
  PUSH the resultant I and G -> stack[level];
  if (level == t_level)
    return;
  else
    goto REPEAT;
}

```

Fig. 8 Depth first traversal algorithm

GND) node. The complexity of this algorithm is $O(L)$, where L is the depth of the tree (the number of series and parallel operations). Although the example used here is a full CMOS logic gate, domino or other dynamic logic gates can also use this data structure and traversal method.

5.2 The subcircuit scheme

The subcircuit block is a group of elements that have their nodes connected by the source and drain nodes of pass transistors, and/or the two terminal nodes of resistors or capacitors. So the boundaries of the subcircuit are the primary input nodes, primary output nodes, gate nodes of MOS devices, and power supply rails (i.e., V_{DD} and GND). All the element models for subcircuit blocks are listed in Fig. 9. In the subcircuit, the direct matrix solving technique is used to solve this tightly coupled block [9].

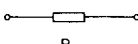
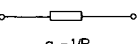
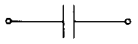
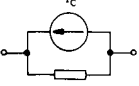
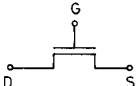
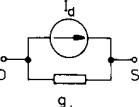
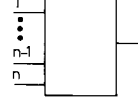
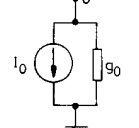
element	symbol	model
resistor		
capacitor		
pass transistor		
logic gate		

Fig. 9 Element models of subcircuit used in EMOTA

5.3 The strategies for partitioning

The objective of partitioning in EMOTA is to decouple the input circuit into a set of loosely coupled blocks. The drain-source path of pass transistors and RC elements are recognised as strongly coupled paths. Therefore, the strategies adopted by EMOTA are, first, connect all elements passing along the strongly coupled path to form a block, and secondly, stop only at the gates of pass transistors or inputs to logic gates. If there is only one logic gate in the block, it is viewed as a single logic gate block and is separated from the subcircuit block. This is due to the fact that the single logic gate block can be solved easily by using the method discussed in Section 2.1.

In the partitioning stage of EMOTA, all the parasitic capacitances connected to one node are lumped as one

capacitance and it is lumped from the node to ground. The parasitic capacitance is calculated approximately by averaging the values through all the working voltage regions, and a good coefficient for the Miller effect is calculated. Therefore, the loading effect of each node has already been included in the block so the coupling effect is avoided. In addition, since each node has a capacitance to ground, the resultant matrix has the characteristics of diagonal dominance. In this condition, there is no pivot problem during the matrix solving process.

A detailed flowchart of the partitioning algorithm is shown in Fig. 10. First, a node is selected as the current node, then, a search is made of all its connectivity to see whether strongly coupled paths exist or not. If so, the element in each path is included in this block and the other nodes of the elements are pushed into the stack. After doing this, the current node is flagged, the nodes in the stack are popped as current nodes and the above process is repeated. Block formation is continuous until the stack is empty. Block attributes, such as input lists, output list, matrix ordering, etc. can now be constructed well. EMOTA then continues to search for an unflagged node as a current node to continue forming the second block. When all nodes are flagged, the job of partitioning comes to an end. EMOTA goes to the subsequent module.

To illustrate the partitioning algorithm, a simple circuit example is shown in Fig. 11a. Initially, node 2 is chosen as the current node, all the elements enclosed by the broken line are connected by a strongly coupled path, hence one block is formed. The input list of this block is nodes 3, 4, 5 and 6 and the output list is nodes 2 and 7. EMOTA continues to search for an unflagged node, that is node 4, as the current node. It is found that only one strongly coupled path, gate A, is connected to it. So, gate A is grouped by itself as a single gate. The only unflagged node now is node 8 and, similarly, only gate B is connected to it, so gate B itself is also grouped as a single gate. Hence, the circuit has been partitioned and grouped into one subcircuit block and two single gate blocks, as shown in Fig. 11b. For convenience of application of the event-driven algorithm, the data structure of the circuit is arranged as shown in Fig. 11c where the left column array denotes the total node number. All the subcircuit and single gate blocks are linked to the array by their input node numbers.

6 Simulation results and discussions

In this Section, several circuits and their simulation results are discussed. These circuits are simulated by using a SUN-3/160C (68020) workstation with a floating point coprocessor.

To illustrate that EMOTA can provide accurate timing waveforms similar to the traditional circuit simulator, a 11-stage cascade inverter is used as an example circuit. In this circuit, each node has an interconnection line capacitance as shown in Fig. 12a. This inverter chain is simulated by EMOTA, and the output waveforms of each odd stage are shown in Fig. 12b. For this case, the interconnection line capacitances are 0.2pf. Fig. 12c shows the comparison of the simulation waveform between EMOTA and SPICE2G.5. It is seen that these two waveforms are in excellent agreement. Table 1a is a summary for the pair delay between EMOTA waveforms and SPICE2G.5 waveforms with various interconnection line capacitances. The difference between these two wave-

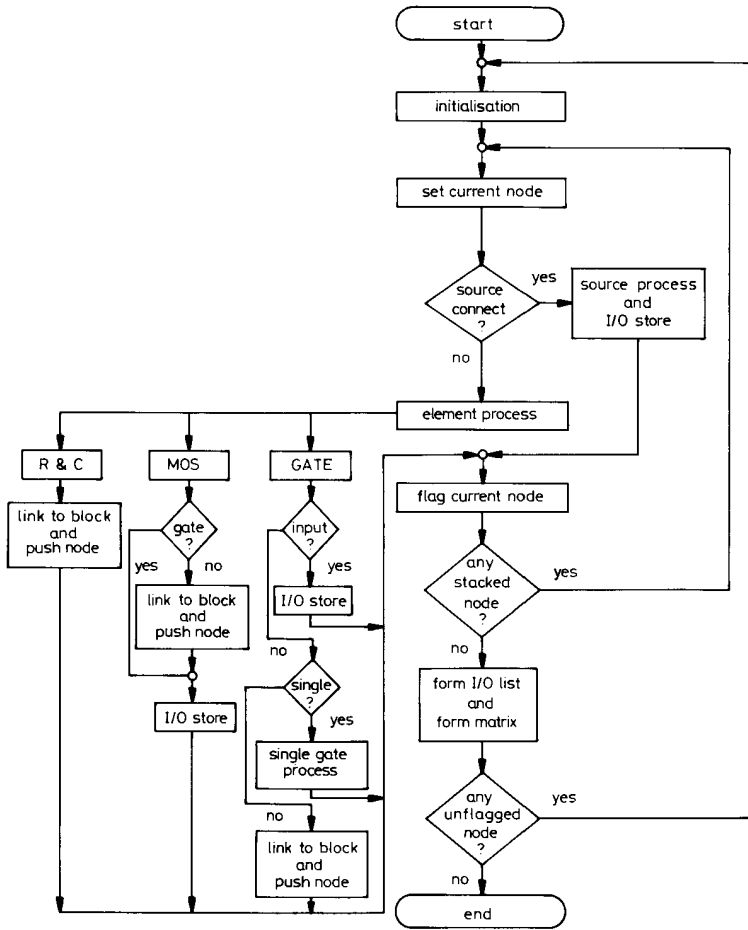


Fig. 10 Partitioning flowchart

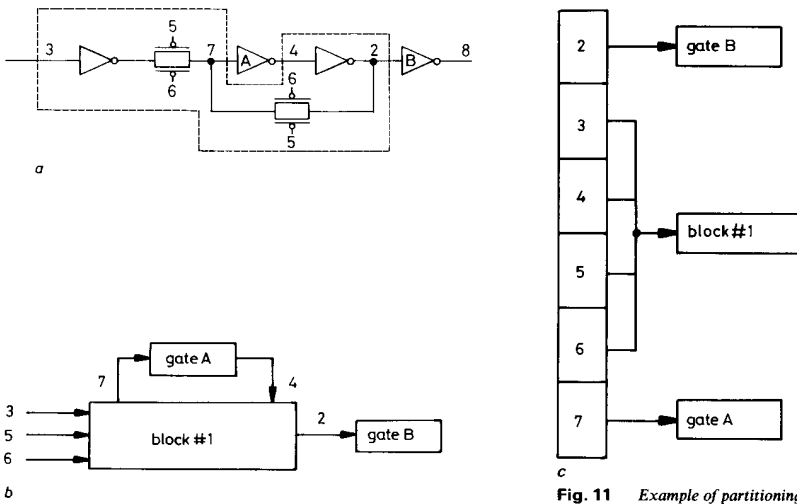


Fig. 11 Example of partitioning algorithm

forms is less than 3%. For the delay of the total 11-stage inverter as shown in Table 1b, the errors are all within 4%. These results are fairly good.

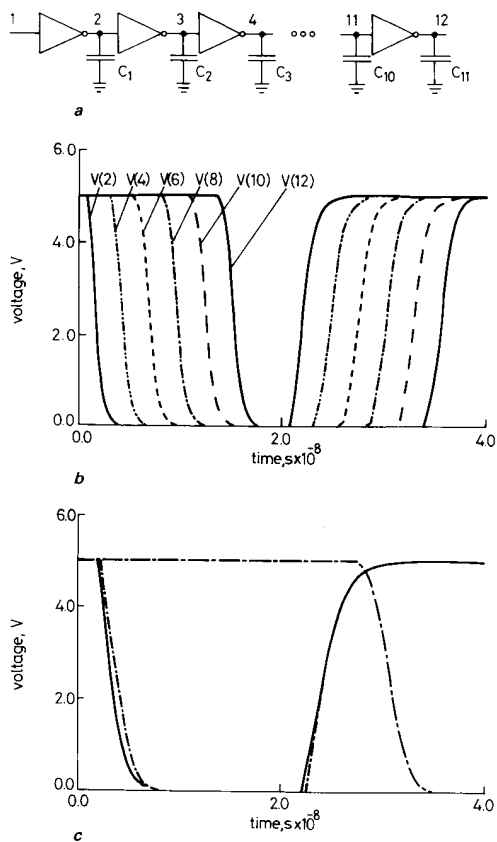


Fig. 12 Simulation of 11-stage cascade inverter

- a Circuit diagram
- b Simulation result of odd stage of inverter
- c Output waveform
- EMOTA for V(2);
- - - SPICE2G.5 for V(2);
- EMOTA for V(12);
- - - SPICE2G.5 for V(12)

Table 1: Accuracy comparison between EMOTA and SPICE2G.5

Node capacitance	EMOTA, ns	SPICE2G.5, ns	Error, %
0pf	0.396	0.388	2.06
0.1pf	1.583	1.557	1.67
0.2pf	2.754	2.722	1.17
0.5pf	6.333	6.218	1.85

a

Node capacitance	EMOTA, ns	SPICE2G.5, ns	Error, %
0pf	1.929	1.871	3.09
0.1pf	7.788	7.704	1.09
0.2pf	13.73	13.54	1.40
0.5pf	31.35	31.16	0.61

- b
- a Pair delay
- b Delay between first and 11th stages

The second example illustrates the logic gate and DC analysis ability. This is a 2-bit ripple counter composed of T flip-flops. The T flip-flop is implemented by using the master-slave J-K flip-flop, as shown in Fig. 13a. This circuit is partitioned into single logic gate blocks. The input clock waveform and the simulation results are illustrated in Fig. 13b which shows that the DC analysis sets up the correct DC state and the event-driven control scheme can correctly simulate the sequential circuits.

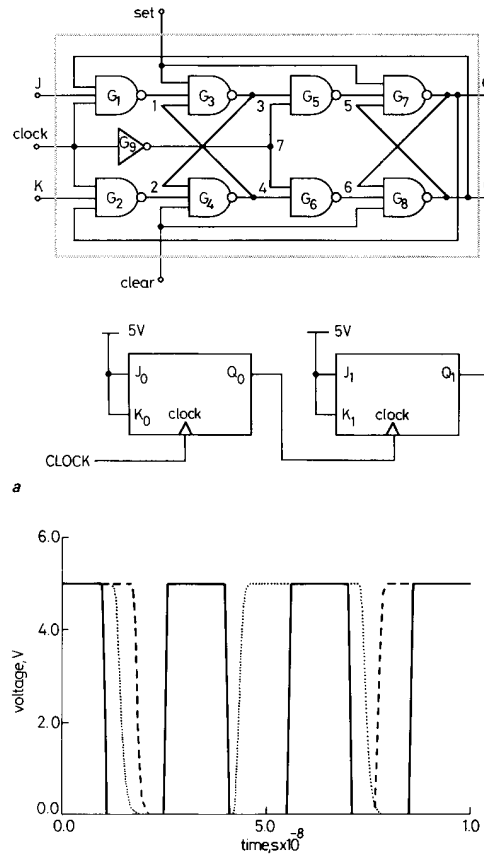
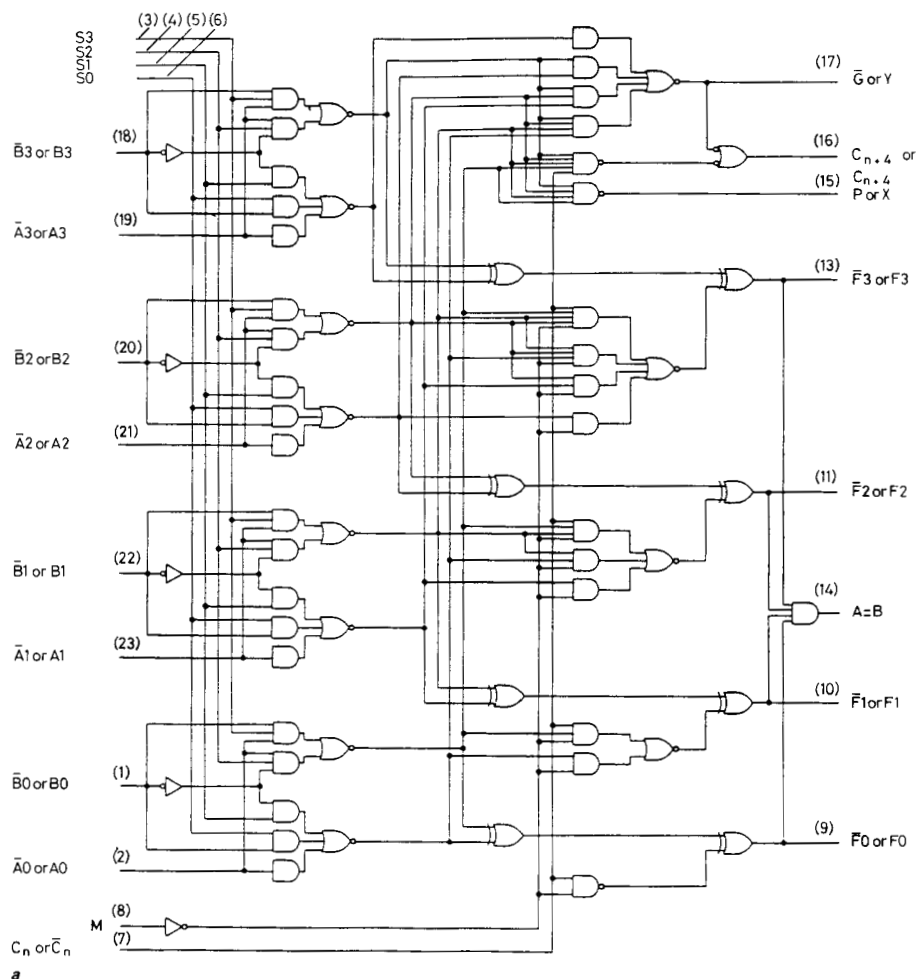


Fig. 13 Simulation of a 2-bit ripple counter

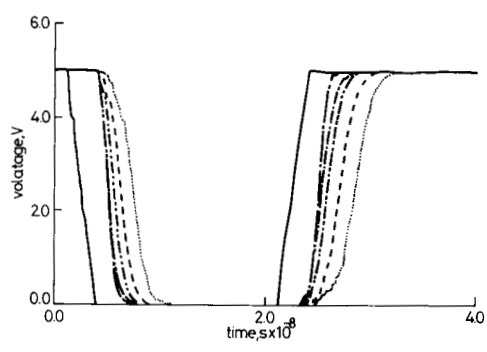
- a Arrangement of gates in flip-flop and circuit diagram of counter
- b Output waveform of counter simulated by EMOTA
- V(CLOCK)
- - - V(Q₁)
- (Q₀)

The third example is a 4-bit ALU converted from a TTL 74LS181. The logic gate diagram of the 74LS181 is shown in Fig. 14a and this circuit is implemented using a driver-load CMOS circuit instead of TTL. This circuit contains 290 MOSFET transistors. If $S_3 S_2 S_1 S_0 = 0001$ and $M = 0$, the ALU performs addition and the simulation results are shown in Fig. 14b. If $S_3 S_2 S_1 S_0 = 0110$ and $M = 1$, the ALU performs an exclusive OR and the simulation results are shown in Fig. 14c.

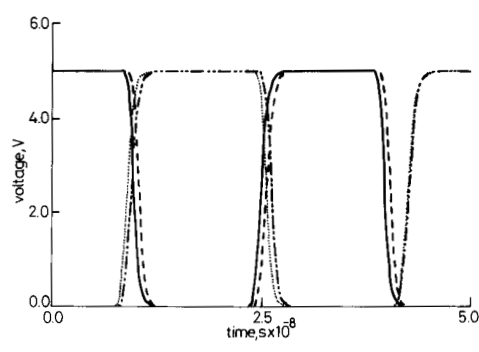
Several circuit examples have been used to demonstrate the time saved by using an event-driven scheme. These examples are simulated with and without an event-driven scheme for comparison. The simulation times of



a



b



c

Fig. 14 4-bit ALU simulation

a Circuit diagram
 b Output waveforms simulated by EMOTA
 — V(7)
 - - - V(9)
 - · - V(10)
 - · - V(11)
 - · - V(13)
 · · · V(16)
 $V(1) = V(22) = V(20) = V(18) = 5\text{ V}; V(2) = V(23) = V(21) = V(19) = 0\text{ V}$

c Output waveforms simulated by EMOTA
 - - - V(10)
 · · · V(11)
 - · - V(13)
 - - - V(9)

these examples are listed in Table 2. We can see from the table that the event-driven scheme indeed saves simulation time, and for large circuits and slow clock rate circuits, the time saving is more significant. The data in Table 2 indicate that the event-driven scheme can increase the simulation speed by almost an order of magnitude.

Table 2: Comparisons of the simulation speed with and without an event-driven scheme

Event control Circuits	With event-driven scheme	Without event-driven scheme
Two inverters ($T_x = 4$)	1.88 s	9.07 s
11 inverters ($T_x = 22$)	8.92 s	47.35 s
20 inverters ($T_x = 40$, $CP = 20$ sn) (Time = 40 ns)	13.37 s	84.07 s
20 inverters ($T_x = 20$, $CP = 20$ sn) (Time = 80 ns)	30.45 s	258.87 s
4 bit full adder ($T_x = 104$)	70.58 s	691.37 s
74181 4-bit ALU ($T_x = 290$)	200.33 s	2221.75 s

T_x = Transistor number; CP = input clock period; time = total time

If circuit size grows, the latency may be increased, i.e. the event-driven scheme will save more simulation time. To illustrate this, the ripple counter of the second example is simulated using EMOTA and SPICE2G.5. This counter circuit is simulated for various sizes (from 2-bit to 32-bit) and the simulation times are listed in Table 3. SPICE2G.5 cannot simulate counters larger than 16 bits due to convergence and memory limitation. The simulation time of a 32-bit counter using SPICE2G.5 is estimated by using extrapolation. The relationship between circuit size and simulation time is shown in Fig. 15. The simulation time for EMOTA is

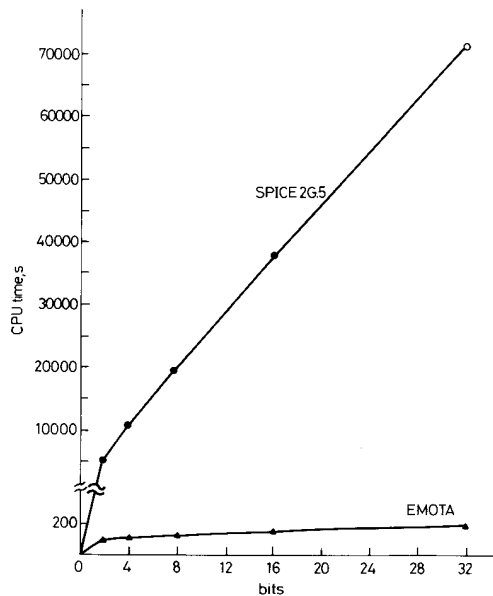


Fig. 15 Simulation time of EMOTA and SPICE2G.5 for increasing circuit size

approximately linear with node activity, which is typically less than linear with circuit size. At 32-bits, the EMOTA is 300 times faster than SPICE2G.5 (extrapolated). These time savings are the result of using the unidirectional Gauss-Seidel relaxation technique, an I&G table lookup model and an event-driven simulation technique. SPLICE [6] simulates a counter-decoder-encoder circuit with 553 MOSFETs about 66 times faster than SPICE2G. Compared with this example, EMOTA is more efficient than SPLICE.

Table 3: Comparison of simulation speed between EMOTA and SPICE2G.5

Bit number	SPICE2G.5, s	EMOTA, s	SPICE2G.5/EMOTA
2 bit	5276.02	101.38	52.04
4 bit	10309.02	134.02	76.92
8 bit	19610.75	153.00	128.17
16 bit	37403.75	180.15	207.63
32 bit	*71000.00	219.97	322.77

* Estimated value

7 Conclusion

In this paper, a relaxation-based event-driven timing simulator for VLSI circuits is presented. Because the input format is very similar to SPICE, users can describe the circuit very easily. According to the circuit connectivity described in an input file, the circuit will be partitioned and grouped into single logic gate blocks and subcircuit blocks automatically. To increase the simulation speed, tree data structures, simulatable timing models, relaxation technique and macromodels are used to solve the driver-load logic gate. Each subcircuit block is solved using the direct matrix solving method and node ordering to overcome the drawback of using relaxation technique.

The analytical models of SPICE2G.5 level 1 and 2 and table models are all implemented in EMOTA to calculate current and conductance. Instead of the bypass scheme, a new selective trace event-driven scheme is employed to exploit the circuit latency. Once the element is activated by the elements connected to its inputs, it is then only triggered by itself. Therefore, the local variable time step is only limited by its own LTE. A special time-wheel mechanism is used to control the time flow and event scheduling.

The results for simulation examples using EMOTA have shown that it can be two or three orders of magnitude faster than SPICE2G.5 when the circuit size becomes large. The errors between EMOTA and SPICE2G.5 are within 5% for the simulation examples. These errors are mainly due to the fact that drain-gate and source-gate capacitances are ignored. Finally, a special purpose hardware simulation engine for EMOTA is currently under investigation.

8 References

- NAGEL, L.W.: 'SPICE2, a computer program to simulate semiconductor circuits'. University of California, Berkeley, USA, Memorandum ERLM520, May 1975
- WEEKS, W.T., JIMENEZ, A.J., MAHONEY, G.W., METHTA, D., QUASSEMZADEH, H., and SCOTT, T.R.: 'Algorithm for ASTAP—a network analysis program', *IEEE Trans.*, 1973, **CT-20**, pp. 628–643
- CHAWLA, B.R., GUMMEL, H.K., and KOZAK, P.: 'MOTIS — An MOS timing simulator', *IEEE Trans.*, 1975, **CAS-22**, pp. 901–909

- 4 CHEN, C.F., and SUBRAMANIAM, P.: 'The second generation MOTIS timing simulator — an efficient and accurate approach for general MOS circuits'. Proceedings of International Symposium on Circuits and Systems, Montreal, Canada, May 1984, pp. 538–542
- 5 FAN, S.P., HSUEH, M.Y., NEWTON, A.R., and PEDERSON, D.O.: 'MOTIS-C: a new circuit simulator for MOS LSI circuit'. Proceedings IEEE International Symposium on Circuits Systems, 1977, pp. 700–703
- 6 SALEH, R.A., KLECKNER, J.E., and NEWTON, A.R.: 'Iterated timing analysis in SPLICE1'. IEEE International Conference on Computer-Aided Design, Santa Clara, California, USA, Nov. 1983, pp. 139–140
- 7 LELARASMEE, E., and SANGIOVANNI-VINCENTELLI, A.: 'RELAX: a new circuit simulator for large scale MOS integrated circuits'. Proceedings 19th Design Automation Conference, Las Vegas, Nevada, USA, June 1982, pp. 682–690
- 8 ODRYNA, P., and NASSIF, S.: 'The ADEPT timing simulation algorithm', *VLSI Syst. Des.*, 1986, 7, (3), pp. 24–34
- 9 JOU, S.-J., SHEN, W.-Z., JEN, C.-W., and LEE, C.-L.: 'MOTA: a MOSFET timing simulator', *IEE Proc. I, Solid-State & Electron Dev.*, 1986, 133, (5), pp. 193–199
- 10 NEWTON, A.R., and SANGIOVANNI-VINCENTELLI, A.: 'Relaxation-based electrical simulation', *IEEE Trans.*, 1983, ED-30, (9), pp. 1184–1207
- 11 VLADIMIRESCU, A., and PEDERSON, D.O.: 'Performance limits of the CLASSIE circuit simulation program', in Proceedings of International Symposium on Circuits and Systems, Rome, Italy, May 1982
- 12 JOU, S.-J., SHEN, W.-Z., JEN, C.-W., and LEE, C.-L.: 'Simulatable timing model of MOS logic circuit', *IEE Proc. Pt. G, Electron Circuits & Syst.*, 1987, 134, (6), pp. 193–199
- 13 SZYGENDA, S.A., and THOMPSON, E.W.: 'Digital logic simulation in a time-based, table-driven environment. Part 1. Design verification', *Computer*, 1975, 18, pp. 24–36
- 14 BRYANT, R.E.: 'An algorithm for MOS logic simulation', *Lambda*, 1980, (4), pp. 46–53
- 15 BAKER, C.M., and TERMAN, C.: 'Tools for verifying integrated circuit designs', *ibid.*, 1980, pp. 22–30
- 16 HO, C.-W., RUEHLI, A.E., and BRENNAN, P.A.: 'The modified nodal approach to network analysis', *IEEE Trans.*, 1975, CAS-22, (6), pp. 504–509
- 17 RUEHLI, A.E., SANGIOVANNI-VINCENTELLI, A.L., and RABBAT, G.: 'Timing analysis of large scale circuits containing one-way macromodels', *IEEE Trans.*, 1982, CAS-29, (3), pp. 185–189
- 18 SAKALLAH, K.A., and DIRECTOR, S.W.: 'SAMSON2: an event driven VLSI circuit simulator', *IEEE Trans.*, 1985, CAD-4, (4), pp. 668–684
- 19 LO, C.-Y., NHAM, H.N., and BOSE, A.K.: 'A data structure for mos circuits'. Proceedings 20th Design Automation Conference, Miami Beach, Florida, USA, June 1983, pp. 619–624
- 20 SUBRAMANIAM, P.: 'Modelling MOS VLSI circuits for transient analysis', *IEEE J.*, 1986, SC-21, (2), pp. 276–285
- 21 JOU, S.-J., SHEN, W.-Z., JEN, C.W., CHANG, C.-C., and TAO, Y.-S.: 'Table lookup MOSFET model and the automatic measure system'. EDMS, Taipei, Taiwan, Republic of China, Sept. 1986, pp. 190–195
- 22 RABBAT, N.G., SANGIOVANNI-VINCENTELLI, A.L., and HSIEH, H.Y.: 'A multilevel network algorithm with macromodeling and latency for the analysis of large scale nonlinear circuits in the time domain', *IEEE Trans.*, 1979, CS-26-9, pp. 733–741