Engineering Simulation, 1997, Vol. 14, pp. 325—332
Reprints available from the Publisher
Photocopying permitted by license only

Parallel Logic Simulation of MOS–Structures at the Switching Level

A. I. ANDRYUKHIN

Institute of Applied Mathematics and Mechanics, National Academy of Sciences of Ukraine, Donetsk

(Received April 5, 1994)

A computer-aided method for parallel logic simulation of switching circuits is suggested. The method uses the hardware features of the IBM PC video system. The algorithms and characteristics of their program realizations are given.

Digital devices manufactured by the MOS- and CMOS-technologies have some features of operation which complicate their simulation when being represented by the gate level models. The bidirectional character of signals presents no difficulties if healthy devices are under consideration but the fault simulation shows clearly the shortcommings of the conventional gate approach [1-3]. In [3] six shortages of the classical switching circuit theory are pointed out as applied to the analysis of modern VLSI MOS-circuits. The specific character of the MOS-circuit operation may be taken into consideration by simulation at the switching level [1-4] using various modifications of Bryant's algorithm [5]. The aim of this work is to speed up the simulation procedures by constructing a parallel simulation algorithm at the switching level which is equivalent to Bryant's algorithm, and prove its practical implementation on IBM PC.

The logic simulation carried out on the "von Neumann" architecture computers is an essentially slow process since the parallel alterations in the circuit are to be imitated in a consecutive order. Therefore, the acceleration of modelling under conditions of constantly growing dimensions of the investigated devices is a problem of high importance.

There are several ways to solve this problem: a) to design specialized computing devices for logical simulation [6] and to use to the full extent their hardware and software facilities; b) to raise the simulation level using some multilevel language (such as the VHDL [7] or IDEAL languages) to describe digital devices; c) to use the hardware features of instrumental computers in the simulation programs.

Specialized logic simulation computers are not always available and often exhibit a number of limitations. Simulation with the use of the VHDL language is promising for these problems but has its own disadvantages: considerable labour contribution for programming; a tiresome debugging of the simulating program which is prepared by the input-output characteristics of the real object and must show adequacy to the latter; qualitative evaluation of the test generated with the aid of these models. One of the well-known ways for logic simulation acceleration is parallel simulation based on the computer's ability to perform operations on digits of the computer word simultaneously and independently (usually 2, 4 bytes or over a field of 256 bytes if the EC computer is used).

In [8,9] it was suggested to use the structural features of the computer instruction list for an IBM 370 computer and compatibles to accelerate simulation of some logic circuits. The literature also gives the productivity estimates for this simulation method.

In a manner described in [8,9] we carry out the logic simulation on a personal computer of the IBM series using its hardware features, in particular, the potentialities of its video RAM. Recall that the PC display screen is an assembly of isolated dots forming a rectangular raster. The number of the dots defines the graphic system resolution. It may be expressed by a pair of numbers of which the first one indicates the number of dots in the line while the second - the number of lines in the screen. Every dot is described by a certain numbers of bits (the attribute) in the video RAM. The attribute consists of 1, 2, 4, and 8 bits depending on the graphic mode. The video RAM is a constituent of the video adapter (the device controlling the PC display operation).

The main prerequisite of the parallel simulation method is the availability of the hardware-performed logic bit-wise operations over the contents of the RAM field and the video RAM buffer, with the result being written in the buffer [10-13]. The list of the operations includes such entities as AND, OR, NOT, and "exclusive-OR". To use these facilities it is sufficient to know the action of functions "putimage" and "getimage" for programming in the C language [12]. The parameters of these functions are the rectangular domain coordinates at the display and the type of logic operation. So, using a single action of screening we can execute a logic operation over the computer memory fields having tens of kilobytes.

Going to the suggested method, note that the input data for the algorithm are, firstly, the description of the circuit having MOS- transistors differing in their high and low resistances, and the nodes characterized by high and low capacitance. For convenience this description is supposed to be equivalent to those in the "esim" language [4]. Secondly, the sequence of input signals is specified. The algorithm yields the signal values at each node of the device as responses to given input actions.

Let $\{Z, (C0, C1), CX, (SC0, SC1), SCX, (W0, W1), WX, (D0, D1), DX\}$ be the simplest signal set needed for n-MOS-circuit simulation which is ordered in reference to $\leq$. We use ordinary notations D, W, SC and C to describe the forces (controlled, weak, supercharged, and charged) and notations $0,1,X$ to describe the node state [4]. So, for any signal $S = (H, G)$, where $H(G)$ is the force value (state value).

A particular place is occupied by the signal $Z = (Z_H, Z_G)$, which is taken as node disconnection.

The signal value at node v in the MOS-circuit may be determined as the magnitude of the "strongest" signal among all signals caming to it by all acyclic paths passing through the conducting transistors from the device input nodes and the nodes where the capacity existed initially. There is a simple iterative technique for calculating the node response to the input actions for this rather complicated definition. Let the signal value at u (val(u)) be $S_u = (H_u, G_u)$. For each node v a new signal value $S^{(i)}$ is calculated in compliance with joining the previous value v and function values $F_{uv}($ val(u)) associated with the transistor between nodes u and v for all neighbours u of node v [4]. The calculations continue until $S^{(i+1)}$ is equal to $S^{(i)}$. Denote F_{uv} (val(u)) as $f(T, R, H_u, G_u)$, where u is the node connected directly to v through a transistor of the T type which has the value R at its gate. The function f of the signal transformation when the signal is passing through the transistor will be considered later. These iterative calculations give wrong results for some complex circuits. Bryant's algorithm modifies the calculations performed by this calculation scheme, which is a ground for a so-called simple algorithm using the distributive transistor functions f. At first we consider the simple algorithm with the distributive transistor transformation functions and then proceed to Bryant's parallel algorithm.

Recall the transformation rules for a signal passing through transistors [2,4]. First consider the circuits consisting of the n-MOS-transistors only. The switching-type n-MOS-transistor will be denoted as $T = 1$, while the load-type transistor (put instead of resistor in the MOS-circuits) $T = 0$. Denoting the transistor gate value by R, assume that $R = 0$ for the cut-off transistor and $R = 1$ for the conducting transistor. The first rule states that for the cut-off transistor, its source and drain have no effect on each other, i. e. are disconnected, and $f(T, R, H, G) = Z$ at $R = 0, T = 1$. It follows from the second rule that $f(T, R, H, G) = (H, G)$ at $T = 1$ and $R = 1$ which means full signal transition through the conducting switching transistor. The third rule defines the conducting load transistor operation ($T = 0$). It may be written in the form $f(T, R, H, G) = (W, G)$ at $H = D$ or as $f(T, R, H, G) = (H, G)$, otherwise. Extending the transformation rules for signals passing through transistors [4], we can define the transformation function f for the signal $S = (H, G)$ passing through the T type transistor with the state signal value R at the gate:

$$f(T, R, H, G) = \begin{cases} Z \text{ at } T = 1 \text{ and } R = 0; \\ (H, G) \text{ at } T = 1 \text{ and } R = 1; \\ (H, G) \text{ at } T = 0 \text{ and } H < D; \\ (W, G) \text{ at } T = 0 \text{ and } H = D. \end{cases}$$

Now represent f by the totality of Boolean functions which may be evaluated in parallel for the simple algorithm and Bryant's algorithm. The forms of these functions depend on the force and state value coding. Since the main memory capacity is occupied by the link description, the saving of memory hardly depends at all on the coding type of the force signal or node state. So, in order to raise the simulation speed we assume the following signal coding: $Z = (Z_h, Z_g)$, where $Z_h = (0,0,0,0)$ and $Z_g = (0,0,0)$, $D = (1,0,0,0)$, $W = (0,1,0,0)$, $SC = (0,0,1,0)$, $C = (0,0,0,1)$,$X = (1,0,0)$, $1 = (0,1,0)$ and $0 = (0,0,1)$.

Let $F = f(T,R,H,G)$, then for the bit components $F = (FH_1, FH_2, FH_3, FH_4, FG_1,FG_2, FG_3)$ for the chosen coding we obtain the following expressions, with the assumption that $R = (R_1, R_2, R_3)$, $H = (H_1, H_2, H_3, H_4)$ and $G = (G_1, G_2, G_3)$. For the node state values

$$FG_1 = G_1(\overline{T} \vee TR_2);$$

$$FG_2 = G_2(\overline{T} \vee TR_2);$$

$$FG_3 = G_3(\overline{T} \vee TR_2); \qquad (1)$$

while for the force values

$$FH_1 = H_1TR_2;\ FH_2 = H_2(TR_2 \vee \overline{T}\,\overline{H}_1) \vee \overline{T}\,\overline{H}_1;$$

$$FH_3 = H_3(TR_2 \vee \overline{T}\,\overline{H}_1);\ FH_4 = H_4(TR_2 \vee \overline{T}\,\overline{H}_1). \qquad (2)$$

These formulae which are symmetric about the source and drain, may be used for the simple simulation algorithm using the distributive transistor functions [4, algorithm 9.2]. The base of the parallel algorithm is formed by the following data structures.

Using device description in the "esim" language we must generate the structures Q_1, Q_2, Q_3, T, representing the arrays with the length $L = \sum k_i$, $i = 1, N$. Here k_i denotes the number of neighbours of the i-th node and N is the number of the circuit nodes. Generation of these arrays is reduced to ordering the records of the device expanded description in the "esim" language by their drain numbers. To get the expanded description, each node number must be supplied with the initial description record if the node number represents a drain or a source node. If the node number is a source node number we change the positions of the drain and source numbers in the record. The treatment scheme is symmetrical with respect to source and drain.

In the program implementation it is assumed that the nodes with numbers 1(2) have logic values 0(1) respectively, and information about external and some internal nodes is modified. Element $Q_2(k)$ is the node number of the gate in the transistor put between the nodes with numbers $Q_1(k)$ and $Q_3(k)$. To understand these data we need to consider circuit example 9.24 given in [4]. This circuit and its description in the "esim" language are shown in Figure 1. The corresponding structures Q_1, Q_2, Q_3, T will look as follows: $Q_1 = (1, 2, 3, 4, 5, 5, 5, 6, 6, 7, 8, 9)$, $Q_2 = (2, 2,2, 2, 5, 4, 2, 4, 3, 2, 7, 7)$, $Q_3 = (1, 2, 3, 4, 2, 6,7, 5, 1, 5, 9, 8)$ and $T = (1, 1, 1, 1, 0, 1, 0, 1, 1,0, 1, 1)$. Nodes 1 and 2 are the earth and power supply respectively, while nodes 3 and 4 are external inputs. Array T describes the transistor mode of operation: $0(d)$ is the depletion mode and $1(e)$ is the enrichment mode.

Denote by X the circuit node value field whose component length is $[N/8] + 1$ bytes. Completion of the algorithm operation gives the steady signal values at the circuit nodes. The working fields with the components length of $[L/8] + 1$ bytes are assumed to be selected: S^0 (S^1) are the fields of the force signal values and state values at the circuit nodes whose numbers are contained in array Q_1; R are the fields of the state signal values for the transistor gate nodes whose numbers are contained in array Q_2; (H, G) is the force (state) signal value field at the nodes corresponding to the connection array Q_3; F is the working field for the signal values transformed by function f. Note that the fields(X, S^0, S^1, F) consist of seven components.

Using the above notations we can write the algorithm in the following form:

1. If input actions exist, write them in X, otherwise go to exit. (Usually, the device input terminals take values $D0, D1$, while at the beginning of simulation the internal node values are CX).

2. Enter the values from X to the field S^0 according to array Q_1.

3. Enter the values from X to the fields R, (H, G) according to arrays Q_2 and Q_3, respectively.

4. Calculate values F according to Eqs. (1) and (2).

5. Determine in the loop for every node v, its new value by choosing the maximum value from the values of the neighbouring set (calculated in field F)) and its previous value according to Hasse lattice [3,4,] and enter this value into $S^1$and X.

6. If S^0 is equal to S^1, then write solution X and go to step 1.

7. Interchange the positions of S^0 and S^1 (in a code it may be realized by mutual changing of the corresponding address pointers) and go to step 3.

It is apparent that at step 4 the calculations of the transformation function values for the nodes of the whole device are carried out using the video memory functions. Step 5 determines $\text{lub}\{u_i\}$ for node v with neighbour set u_i [2,4]. It is a rather simple procedure determine by Hasse's lattice the least upper bound (lub), involving ordering of nodes in structure S^0 and the corresponding signals in F.

So, the determination of the value at node v as the least upper bound (lub) of a set of the root unblocked paths leading to the vertex v in the ternary switching graph [4,5] is reduced to an iteration process the code which is simple to construct.

Modified Bryant's algorithm is performed as following:

1. Set for all internal nodes the force value C or SC.

2. Set all the internal node state signals in X.

3. Calculate the force values by (1), (2) assuming that the transistor with the

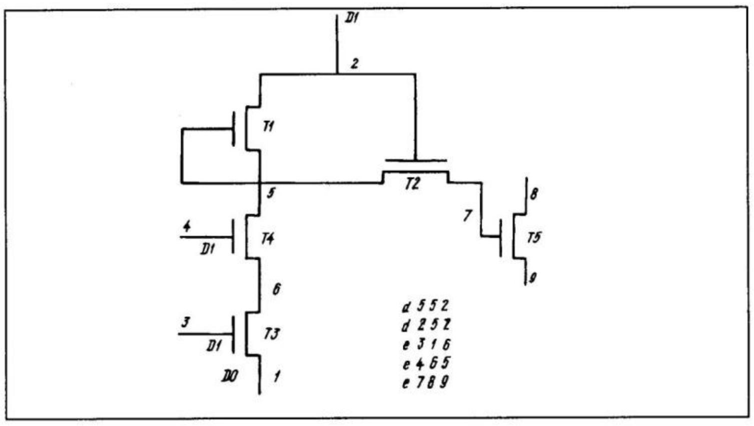

Figure 1

gate state X is in the conducting mode (instead of R_2 we use $\overline{R}_3$). Denote the solution as $S^x = (G_1^x, G_2^x, G_3^x, H_1^x, H_2^x, H_3^x, H_4^x)$.

4. Calculate the solution $S^1 = (G_1^1, G_2^1, G_3^1, H_1^1, H_2^1, H_3^1, H_4^1)$ by (1), (2). However, when lub is determined only those $f(T, R, H, G)$ are taken into account for which the variable of $Y = Y(f)$ is equal to unity. The value of Y is equal to zero when: a) state 1 passes through the gate with indeterminate value X ; b) if the signal force S^1 is not equal to the force S^x.

Condition "a" may be written as $\overline{R}_1 \vee \overline{G}_2^1$,, condition "b" — as $H_1^1 H_1^x \vee H_2^1 H_2^x \vee H_3^1 H_3^x \vee H_4^1 H_4^x$ and so Y is equal to the product of these expressions. Further, in the determination of lub we use $Yf(T, R, H, G)$.

5. This step is similar to step 4, but now the zero passing through the transistor with the gate X is forbidden, and the solution is $S^0 = (G_1^0, G_2^0, G_3^0, H_1^0, H_2^0, H_3^0, H_4^0)$. The conditions similar to step 4 may be written in the form $\overline{R}_1 \vee \overline{G}_3^0$ and $H_1^0 H_1^x \vee H_2^0 H_2^x \vee H_3^0 H_3^x \vee H_4^0 H_4^x$, respectively.

6. Replace the solutions in step 3 changing the states by the indication of the variation $Y = G_1^0 G_1^1 \vee G_2^0 G_2^1 \vee G_3^0 G_3^1$, and the complete solution is $S = (G_1, G_2, G_3, H_1^x, H_2^x, H_3^x, H_4^x)$, where $G_1 = YG^0 \vee \overline{Y}$, $G_2 = YG^0$, $G_3 = YG^0$.

The example of the simple parallel algorithm operation is shown in Table 1 for the device with the unity input actions at the external inputs 3 and 4 (see the Figure). In the circuits specially constructed for checking the possibilities of the suggested algorithms based on using the PC features mentioned above (the gates were replaced by appropriate transistor structures) and consisting of 1179, 2979, 7107, 12771, and 20938 transistors the processing times for 10 input sets were 10, 25, 60, 110, and 170 s, respectively. The average number of iterations was 15. The simulation was carried out on IBM PC AT/386 with the clock rate of 40 MHz.

The simulation of the MOS- and C-MOS-structures poses the need introduce the *p*-MOS-transistor, which implies certain complication of the equation for the transformation function *f*, because if a large number of transistors is described, bit variables T_1, T_2 used for their coding are needed. Let for the load transistor $T_1 = 0$ (for T_2 — indifferently); for the *n*-MOS-transistor we have $T_1 = 1, T_2 = 1$, and for the *p*-MOS-transistor $T_1 = 1,\ T_2 = 0$. Taking into account that the *p*-MOS-transistor, in the view of the logics, operates as a *n*-MOS-transistor with the inverse gate value, we get the equations similar to (1) and (2):

$$FG_1 = G_1T_1\overline{(R_2 \oplus T_2)} \vee G_1\overline{T_1};$$

$$FG_2 = G_2T_1\overline{(R_2 \oplus T_2)} \vee G_2\overline{T_1};$$

$$FG_3 = G_3T_1\overline{(R_2 \oplus T_2)} \vee G_3\overline{T_1};$$

Table 1.

Iteration index	Field name	Node value in the field											
1	S^0	$D0$	$D1$	$D1$	$D1$	CX	CX	CX	CX	CX	CX	CX	CX
	R	1	1	1	1	X	1	1	1	1	1	X	X
	H, G	$D0$	$D1$	$D1$	$D1$	$D1$	CX	CX	CX	$D0$	CX	CX	CX
	F	$D0$	$D1$	$D1$	$D1$	$W1$	CX	CX	CX	$D0$	CX	Z	Z
2	S^0	$D0$	$D1$	$D1$	$D1$	$W1$	$W1$	$W1$	$D0$	$D0$	CX	CX	CX
	R	1	1	1	1	1	1	1	1	1	1	X	X
	H, G	$D0$	$D1$	$D1$	$D1$	$D1$	$D0$	CX	$W1$	$D0$	$W1$	CX	CX
	F	$D0$	$D1$	$D1$	$D1$	$W1$	$D0$	CX	$W1$	$D0$	$W1$	Z	Z
3	S^0	$D0$	$D0$	$D0$	$D0$	$D0$	$D0$	$D0$	$D0$	$D0$	$W1$	CX	CX
	R	1	1	1	1	0	1	1	1	1	1	1	1
	H, G	$D0$	$D0$	$D0$	$D0$	$D1$	$D0$	$W1$	$D0$	$D0$	$D0$	CX	CX
	F	$D0$	$D0$	$D0$	$D0$	$W1$	$D0$	$W1$	$D0$	$D0$	$W0$	CX	CX
4	S^0	$D0$	$D1$	$D1$	$D1$	$D0$	$D0$	$D0$	$D0$	$D0$	$W0$	CX	CX
	R	1	1	1	1	0	1	1	1	1	1	0	0
	H, G	$D0$	$D1$	$D1$	$D1$	$D1$	$D0$	$W0$	$D0$	$D0$	$D0$	CX	CX
	F	$D0$	$D1$	$D1$	$D1$	$W1$	$D0$	$W0$	$D0$	$D0$	$W0$	Z	Z

$$FH_1 = T_1H_1\overline{(R_2 \oplus T_2)};$$
$$FH_2 = T_1H_2\overline{(R_2 \oplus T_2)} \vee \overline{T}_1H_2\overline{H}_1 \vee \overline{T}_1H_1;$$
$$FH_3 = T_1H_3\overline{(R_2 \oplus T_2)} \vee \overline{T}_1H_3\overline{H}_1;$$
$$FH_4 = T_1H_4\overline{(R_2 \oplus T_2)} \vee \overline{T}_1H_4\overline{H}_1,$$

where $\oplus$ is the "exclusive-OR" operation. For $T_2 = 1$ (only n-MOS-transistors) they pass to (1) and (2).

To summarize it the method of Boolean representation of component connections of multivalued signals describing discrete device operation may be used in parallel computers for simulation of the gate and neuron structures.

REFERENCES

1. K. Kinosita, K. Asada, O. Karatsu, *Logicheskoe proektirovanie SBIS* (VLSI logical design) (Moscow: Mir, 1988) (Russian translation).
2. E. Randel, A. Bryant, IEEE Trans. on Comp., **33**, N.2, 160-177 (1984).
3. Dj. P. Hayes, in *Trudy Instituta inzhenerov po elektrotekhnike i radioelektronike* (Proceedings of the IEEE), **70**, №.10, 5-19 (1982) (Russian translation).
4. Dj. Ulman, *Vychislitelnye aspekty SBIS* (Computational aspects in VLSI) (Moscow: Radio i svyaz, 1990) (Russian translation).
5. V. S. Lobunov, *Electronnoe modelirovanie,* №.2, 62-66 (1991) (in Russian).
6. A. V. Shmid, A.B. Svyatskii, in: *Itogi nauki i tekhniki. Ser. Tekhn. kibernetika* (Moscow: VINITI, 1986), 20, 136-204 (in Russian).
7. Dj. R. Armstrong, *Modelirovanie tsifrovykh sistem na yazyke VHDL* (Digital circuit simulation using the VHDL language) (Moscow: Mir, 1992) (Russian translation).
8. A. I. Andryukhin, *Avtomatika i vychislitelnaya tekhnika,* №.6,87-88 (1990) (in Russian).
9. A. I. Andryukhin, in: *Modelirovanie i diagnostika upravlyayushchikh sistem,* (Kiev: 1991), 5-8 (in Russian).
10. R. Joudain, *Spravochnik programmista personalnykh kompyuterov tipa IBM PC, XT i A*T (Programmer's Problem Solver for the IBM PC, XT & AT) (Moscow: Finansy i statistika, 1992) (Russian translation).
11. V. L. Grigoriev, *Videosistemy PK firmy IBM* (Video systems for the IBM PC) (Moscow: Radio i svyaz, 1993) (in Russian).
12. B. P. Prokofiev, N. N. Sukharev, Yu. E. Khramov, *Graficheskie sredstva Turbo C i Turbo C++* (Turbo C and Turbo C++ graphics) (G. V. Gens and Yu. E. Khramov, eds.) (Moscow: Finansy i statistika, 1992) (in Russian).
13. V. G. Chertkov, in: V mire PK,In the PC Kingdom №.1, 115-125 (1993) (in Russian).