Int. J. Elect. Enging. Educ., Vol. 30, pp. 216–223. Manchester U.P., 1993. Printed in Great Britain

MATRIX MODEL OF LOGICAL SIMULATOR WITHIN SPREADSHEET

ILYA LEVIN
School of Education, Tel Aviv University, Israel

1 INTRODUCTION

Spreadsheets have been gaining ground in education both in support and in new fields of application. This trend is most obvious in the fields of mathematical education[1–3] and science education[4–6].

Another potential explorative and prospective field of application for spreadsheets is in technological education. This paper emphasizes two major directions for using spreadsheets in technological education — both as a means of simulation and as a means of control.

A number of recent studies[7–12] deal with the development and application of simulations in the technological curriculum. The use of spreadsheets as a means of simulation in teaching computer engineering was first studied by Huelsman in 1984[13]. This work contains samples of the usage of the logical capacity of spreadsheets. The spreadsheet-oriented method of logical gates networks simulation was proposed in another study[14].

The present paper aims at studying the use of spreadsheets for the simulation of a particular type of integrated circuits — Programmable Logic Arrays (PLA). This simulation is especially interesting, since it allows the simulation of integrated circuits as well as anything lending itself to description by logical functions systems.

Another possible direction of using spreadsheets in technological education is as a tool for the implementation of intelligent control. Intelligent control constitutes the contemporary base of technological education[15]. The present study will also test the logical capacity of the spreadsheet as a base for building logical control systems.

The use of spreadsheets for simulation of PLA and building logical control systems can be achieved by using the same *matrix model* of logical simulator proposed in the present paper.

The fragment of a spreadsheet which simulates the functioning of a circuit is the *spreadsheet-model* of the circuit. Every cell (i, j), placed on intersection of i-row and j-column of the spreadsheet-model is programmed for the implementation of a model-oriented function Fij. This function is the *indicative function* of the spreadsheet-model. The representation of the circuit in the form of the spreadsheet-model is the *simulation of the circuit*. The spreadsheet simulations that are based on the matrix model are the *matrix simulations*.

2 MATRIX IMPLEMENTATION OF LOGICAL FUNCTIONS

Let us consider the example of a matrix implementation[16] of a system of logical functions, using a system presented in a disjunctive normal form (DNF):

$$y_1 = x_1 x_2' + x_1' x_2 x_4 + x_1' x_3' x_4';$$
$$y_2 = x_1 x_2' + x_1 x_3 x_4' + x_1 x_2 + x_2 x_4; \qquad (1)$$
$$y_3 = x_1' x_2 x_4 + x_1 x_2 + x_1' x_3'.$$

In equations (1) x' denotes *not* (x).

Any logical function system $y_1, \ldots y_N$ of variables $x_1, \ldots x_M$ can be presented in a DNF and implemented in the form of a two level structure: conjunctions $X_1, \ldots X_H$ are implemented on the first level of this structure and disjunctions $y_1, \ldots y_N$ of these conjunctions on the second level.

Logical circuits are often implemented in the form of matrix integrated circuits. The matrix structure and matrix implementation of our example are shown in Figs. 1 and 2, respectively. In these figures, *AND array* M_1 and *OR array* M_2 are the systems of orthogonal buses with semiconductors at their crosspoints. Every horizontal bus of M_1 represents the conjunction X_h of the logical functions systems. Every vertical bus of M_2 represents the disjunction y_n of the system. In other words matrix M_2 implements N different disjunctions of H different conjunctions formed in matrix M_1.

Every horizontal bus of the *AND* array on Fig. 2 corresponds to one term (conjunction) of the DNF. The transistor appears at the intersection of horizontal X_h and vertical x_m if a variable is included in the term X_h in the negative form. The transistor appears in the intersection of the X_h and x_m', if x_m is included in X_h in direct form (without inversion).

Matrix descriptions of the DNF are usually represented in the form of tables. Columns appearing in the tables are marked by variables $x_1, \ldots x_M$ and functions $y_1, \ldots y_N$. Every conjunction is put into accordance with the row of the table. Character *1* appears at the intersection of row m and column h, if the

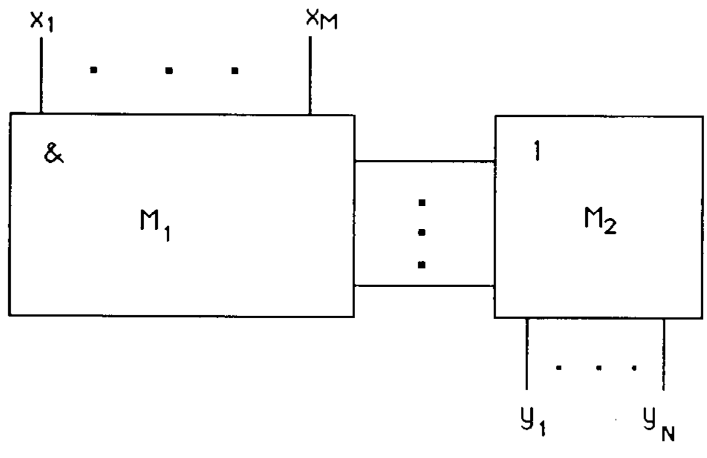

FIG. 1 The matrix structure.

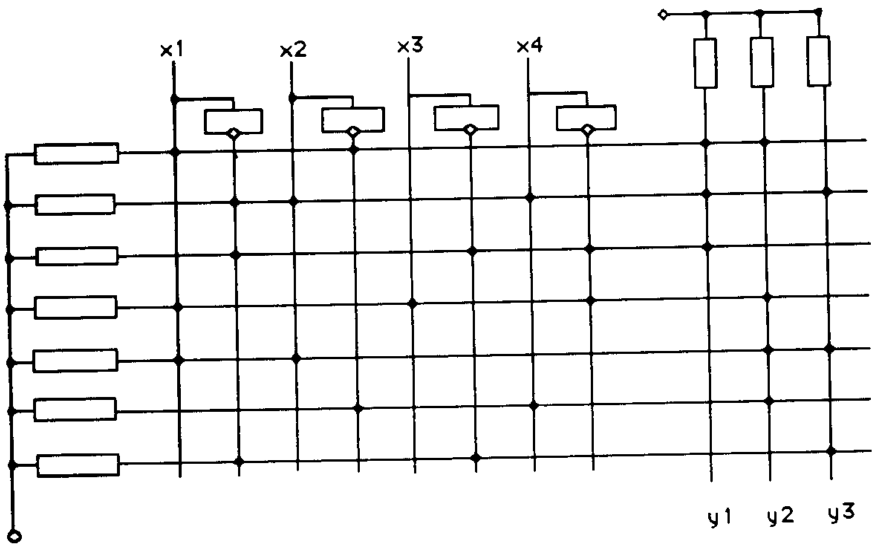

FIG. 2 The matrix implementation.

variable x_m is contained in the term X_h in the direct form. Character *0* appears at the intersection of row *m* and column *h*, if the variable x_m is contained in the term X_h in the negative form. Character ® appears at the intersection of row *m* and column *h* if the variable x_m is absent in the term X_h.

Character *1* appears at the point of the intersection of row X_h and column y_n if term X_h is contained in the function y_n $(n = 1 \dots N)$, and character • in the opposite case.

The arrays of our example, corresponding to system (1) and to Fig. 2 are shown in Table 1.

An integral circuit implemented in the form of a two level matrix structure which has a fixed number of inputs *M*, conjunctions *H* and disjunctions *N*, is called a Programmable Logic Array. We are dealing with a PLA, programmable by the user. In this case we shall assume that the PLA implements the DNF of the system of logical functions of interest.

TABLE 1 *Table description of the matrix structure*

x1	x2	x3	x4	y1	y2	y3
1	0	®	®	1	1	•
0	1	®	1	1	•	1
0	®	0	0	1	•	•
1	®	1	0	•	1	•
1	1	®	®	•	1	1
®	0	®	0	•	1	•
0	®	0	®	•	•	1

3 MATRIX SIMULATION MODEL

Let us employ the matrix structure as a model for simulation of the DNF within the spreadsheet. We shall now implement DNF (see equation (1)) in the spreadsheet. The suggested method of simulation involves two main ideas.

First, we shall construct the spreadsheet-model of the DNF of a system of logical functions which represents the table of the system (Table 1).

Second, we proceed to construct the spreadsheet model of the PLA making every intersection point of the matrix structure correspond with a cell of the spreadsheet-model.

We propose to use two spreadsheets T_1 and T_2 for our purposes. One of them T_1 is a spreadsheet-model of the DNF, or *the installation* matrix, the second T_2 is the spreadsheet-model of the PLA (Fig. 2). Every cell of T_2 is programmed by a universal indicative function, which will be presented below. This function's value depends on the value of the corresponding cell of the spreadsheet-model T_1 which is a copy of Table 1.

The T_1 spreadsheet-model for our example is exhibited in Table 2.

Let us now construct the matrix T_2. Variables $x_1, \dots x_4$ and $y_1, \dots y_3$ are copied on to the spreadsheet T_2 from Table 1 without change.

In accordance with the matrix structure, any cell of the spreadsheet-model of the matrix of conjunctions can implement one of three different functions: reference to the contents of the left hand cell, conjunction of the variable and contents of left hand cell, conjunction of the inversion of the variable and contents of left hand cell. The spreadsheet-model of the disjunction matrix can implement one of two functions: reference to the value of the higher preceding cell or the disjunction of the higher preceding cell and corresponding output of the matrix of conjunction.

Indicative functions of the spreadsheet-model T_2 are defined by the following rules.

For the AND array

If *1* appears in the cell (m, h) of T_1, then the conjunction of the value of the corresponding variable and that of preceding left hand cells will be updated

TABLE 2 *The spreadsheet model of the DNF*

	1	2	3	4	5	6	7	8
1		x 1	x 2	x 3	x 4	y 1	y 2	y 3
2		1	0	®	®	1	1	•
3		0	1	®	1	1	•	1
4		0	®	0	0	1	•	•
5		1	®	1	0	•	1	0
6		1	1	®	®	•	1	1
7		®	0	®	0	•	1	•
8		0	®	0	®	•	•	1

into the correspondent cell $F_{m,h}$. This can be expressed by the following indicative function:

$$F_{m,h} = F_{m-1,h} \& F_{m,1}.$$

If *0* appears in a cell (m, h) of T_1, then the conjunction of the inversion of value of the corresponding variable and of the value of the preceding left hand cells will be updated into the corresponding cell $F_{m,h}$. This can be expressed in a form of the indicative function:

$$F_{m,h} = F_{m-1,h} \& NOT(F_{m,1}).$$

If ® appears in a cell (m, h) of T_1, then the value of the corresponding variable will be updated into the corresponding cell $F_{m,h}$. This can be expressed by indicative function:

$$F_{m,h} = F_{m-1,h}.$$

Formally, all these conditions can be expressed as one function:

$$F_{m,h} = \begin{cases} F_{m-1,h} \& F_{m,1}, \text{ if } 1 \text{ appears in the cell } (m, h) \text{ of } T_1, \\ F_{m,h} = F_{m-1,h} \& NOT(F_{m,1}) \text{ if } 0 \text{ appears in cell } (m, h) \text{ of } T_1, \\ F_{m-1,h}, \text{ if ® appears in the cell } (m, h) \text{ of } T_1. \end{cases} \quad (2)$$

For the OR array

If character *1* appears in the cell (m, h) of T_1, then the disjunction of the value of the corresponding variable and the value of the higher preceding cell will be updated into the corresponding cell $F_{m,h}$. The indicative function in this case is:

$$F_{m,h} = OR(F_{M,h}, F_{m,h-1}).$$

If character • appears in a cell (m, h) of T_1, then the value of the higher preceding cell will be updated into the corresponding cell $F_{m,h}$. This can be expressed by the indicative function:

$$F_{m,h} = F_{m,h-1}.$$

Formally both of previous conditions can be expressed in the form of universal indicative function:

$$F_{m,h} = \begin{cases} OR(F_{M,h}, F_{m,h-1}), \text{ if } 1 \text{ appears in the cell } (m, h) \text{ of } T_1, \\ F_{m,h} = F_{m,h-1}, \text{ if } \bullet \text{ appears in the cell } (m, h) \text{ of } T_1. \end{cases} \quad (3)$$

Functions (2) and (3) are indicative functions of the corresponding spreadsheet-models. The spreadsheet-model of AND and OR arrays are filled in by the functions (2) and (3) correspondingly.

In our example the spreadsheet-model of the AND array is the fragment of the spreadsheet R2C3:R5C9. This fragment is programmed for the implementation of the function (2). Namely, in the cells of this fragment the following spreadsheet function is written:

$$=\text{IF(T1!RC=1,AND(R2C,RC[-1]),}$$
$$\text{IF(T1!RC="®",RC[-1],}$$
$$\text{IF(T1!RC=0,AND(NOT(R2C),RC[-1]),1))).}$$

In our example the spreadsheet-model of the OR array is the fragment of the spreadsheet R6C3 : R8C9. This fragment is programmed for the implementation of the function (3). Namely, in cells of this fragment the following spreadsheet function is written:

$$=\text{IF(T1!RC=1,OR(R[-1]C,RC5),IF(T1!RC="•",R[-1]C,1)).}$$

Cells R2C2 : R5C2 included an input vector of variables $x_1, \dots x_4$. Cells R6C9 : R8C9 included an output vector of functions $y_1, \dots y_3$.

Spreadsheet-model T_2 is exhibited on Table 3.

TABLE 3 *The spreadsheet model of the PLA*

	1	2	3	4	5	6	7	8
1		x1	x2	x3	x4	y1	y2	y3
2		*TRUE*	*TRUE*	*TRUE*	*FALSE*	*FALSE*	*FALSE*	*FALSE*
3	=*TRUE()*	=IF(T1!RC=1,AND(R	=IF(T1!RC=1,AND(R	=IF(T1!RC=1,AND(R	=IF(T1!RC=1,AND(	=IF(T1!RC=1,OR(	=IF(T1!RC=1,OR(	=IF(T1!RC=1,OR(
4	=*TRUE()*	=IF(T1!RC=1,AND(R	=IF(T1!RC=1,AND(R	=IF(T1!RC=1,AND(R	=IF(T1!RC=1,AND(	=IF(T1!RC=1,OR(	=IF(T1!RC=1,OR(	=IF(T1!RC=1,OR(
5	=*TRUE()*	=IF(T1!RC=1,AND(R	=IF(T1!RC=1,AND(R	=IF(T1!RC=1,AND(R	=IF(T1!RC=1,AND(	=IF(T1!RC=1,OR(	=IF(T1!RC=1,OR(	=IF(T1!RC=1,OR(
6	=*TRUE()*	=IF(T1!RC=1,AND(R	=IF(T1!RC=1,AND(R	=IF(T1!RC=1,AND(R	=IF(T1!RC=1,AND(	=IF(T1!RC=1,OR(	=IF(T1!RC=1,OR(	=IF(T1!RC=1,OR(
7	=*TRUE()*	=IF(T1!RC=1,AND(R	=IF(T1!RC=1,AND(R	=IF(T1!RC=1,AND(R	=IF(T1!RC=1,AND(	=IF(T1!RC=1,OR(	=IF(T1!RC=1,OR(	=IF(T1!RC=1,OR(
8	=*TRUE()*	=IF(T1!RC=1,AND(R	=IF(T1!RC=1,AND(R	=IF(T1!RC=1,AND(R	=IF(T1!RC=1,AND(	=IF(T1!RC=1,OR(	=IF(T1!RC=1,OR(	=IF(T1!RC=1,OR(
9	=*TRUE()*	=IF(T1!RC=1,AND(R	=IF(T1!RC=1,AND(R	=IF(T1!RC=1,AND(R	=IF(T1!RC=1,AND(	=IF(T1!RC=1,OR(	=IF(T1!RC=1,OR(	=IF(T1!RC=1,OR(

4 CONCLUSION

The pair T_1–T_2 constitutes the convenient basis for simulation of any logical function system. After any changes made in the spreadsheet-model T_1, spreadsheet-model T_2 begins to implement a new system of logical functions.

Thus user can work only with T_1, debugging its contents for purposes of control system design in a process of learning. The spreadsheet-model T_2 is an universal spreadsheet-model and an analog of the user-installed PLA. This type of integrated circuits is an accepted digital design practice at present.

The proposed logical simulator can be constructed by students individually. They can program the installation matrix for certain logical function system implementations, to carry out an experiment, to debug the system, and to explore the different properties of logical functions of the system and its matrix structure. The student can be both a designer of the matrix structure and a programmer of the applied logical system in the process of developing the matrix structure.

The matrix model proposed in this paper can be used both in learning and teaching logical functions and also, for example, in the following parts of the technology, electronics engineering or computer engineering curriculum:

- canonical method of structural implementation of control devices;
- synthesis of automaton on a PLA basis;
- different types of the matrix integrated circuits;
- logic and technology of homogeneous structures.

The other possible way of using the proposed matrix model is the logical control of learning environments. The spreadsheet based matrix structure is actually a flexible intelligent controller which with good perspectives in technological education.

REFERENCES

[1] Healy, L. and Sutherland, R., 'The use of spreadsheet within the mathematics classroom', *International Journal of Mathematical Education and Science Technology*, **21**(6), pp. 847–862 (1990)

[2] Arad, O. S., 'The spreadsheet: solving word problems', *Computing Teacher*, 14, 13–15, 45 (1986–87)

[3] MacDonald, J., 'Integrating spreadsheets into the mathematics classroom', *Mathematics Teacher*, **81**(11), pp. 616–622 (1988)

[4] Finnemore, Dj., 'More spreadsheets in science teaching', *School Science Review*, June, **71**(257), pp. 92–98 (1990)

[5] Kary, R., 'Spreadsheets in advanced physics chemistry', *Journal of Computers in Mathematics and Science Teaching*, **10**(1), pp. 39–49 (1990)

[6] Brosnan, T., 'Teaching chemistry, using spreadsheets', *School Science Review*, **70**(252), pp. 40–45 (1989)

[7] Roth, Gene L., 'A bright future for computer simulations', *Technology Teacher*, pp. 13–16 (November 1987)

[8] Corsini, P. and Rizzo, L. SSCSSC, 'A Tool for the teaching of digital circuits', *IEEE Trans. Educ.*, **34**, No. 1, pp. 70–75 (1991)

[9] Fuchs, W. K., Page, W., Patel, J. and Tobin Paul., 'Workstation-based logic animation and microarchitecture emulation for teaching introduction to computer engineering', *IEEE Trans. Educ.*, **34**, No. 1, pp. 218–225 (1989)

[10] Reid, R. J., 'Computer-aided engineering for computer architecture laboratories', *IEEE Trans. Educ.*, **34**, No. 1, pp. 56–61 (1991)

[11] Singh, M., 'Role of circuit and logic simulators in EE curriculum', *IEEE Trans. Educ.*, **32**, No. 3, pp. 411–414 (1989)

[12] Williams, R., 'Logic simulator to support design instruction', *IEEE Trans. Educ.*, **E–30**, No. 4, pp. 244–247 (1987)

[13] Huelsman, L. P., 'Electrical engineering application of microcomputer spreadsheet analysis programs', *IEEE Trans. Educ.*, **E–27**, pp. 86–92 (1984)

[14] El-Hajj, A. and Kabalan, K. Y., 'A spreadsheet simulation of logic networks', *IEEE Trans. Educ.*, **34**, No. 1, pp. 43–46 (1991)

[15] DeLuca, V. William, 'Intelligent technologies. A new basis for technology education', *Technology Teacher* (7–10 April 1988)

[16] Baranov, S. and Sinev, V., 'Automaton and programmable arrays', *Minsk: Vysheishaya Shkola* (in Russian) (1980)

ABSTRACTS – ENGLISH, FRENCH, GERMAN, SPANISH

Matrix model of logical simulator within spreadsheet

This paper examines the use of spreadsheets for simulation of logical control units. A matrix model is proposed for this aim. The use of this model is appropriate both for teaching and learning of simulation of a specific type of integrated circuit — Programmable Logic Arrays — and also for any type of control unit representable in the form of a logical function system.

Modèle matriciel d'un simulateur logique dans un tableur

Cet article envisage l'utilisation de tableurs pour la simulation d'unités de contrôle logiques. Un modèle matriciel est proposé dans ce but. L'usage du modèle est approprié à l'enseignement et l'apprentissage d'un type spécifique de circuits intégrés- des Programmable Logic Arrays (PLA) et aussi pour tout type d'unité de contrôle représentable sous forme de systèmes à fonction logique.

Matrixmodell des logischen Simulators innerhalb des Tabellenkalkulationsprogramms
Diese Arbeit untersucht die Verwendung von Tabellenkalkulationsprogrammen für die Simulierung logischer Steuerungseinheiten. Ein Matrixmodell wird für diesen Zweck vorgeschlagen. Die Verwendung dieses Modells ist sowohl zur Unterrichtung als auch zum Lernen von Simulierungen einer spezifischen Art integrierter Schaltung — programmierbares logisches Feld — sowie für jede Art Steuerungseinheit in Form eines logischen Funktionssystems geeignet.

Modelo matricial de simulador lógico dentro de hoja de cálculo
Este artículo examina la utilización de hojas de cálculo para la simulación de unidades de control lógicas. Se propone un modelo matricial con este propósito. El uso de este modelo resulta apropiado tanto para el aprendizaje como para la enseñanza de simulaciones de un tipo específico de circuito integrado — Arreglos Lógicos Programables — y asimismo para cualquier tipo de controlador representable en forma de función lógica.