GRAPHICS SIMULATING LOGIC CIRCUITS

Rashpal Ahluwalia* and Irene Gargantini

University of Western Ontario

ABSTRACT

This paper presents a system developed for interactive circuit design by the use of a graphic display unit. Such a system was conceived primarily as a teaching aid for the tutorial sessions of a course on Computer Logic. It provides the student with the possibility of testing the design of a sequential circuit of medium complexity.

The system description emphasizes the structure and generation of display formats for graphic output, more than the applications themselves.

Interaction may take place by using the light-pen for designating the location of a gate and/or through the push-buttons of the display console. A special feature of the system is the display of the output or state value inside each logical element.

*Supported in part by the National Research Council of Canada (Grant No. 1304T91)

ABRÉGÉ

Dans cette communication, on décrit un système interactif, avec console de visualisation pour faire des plans de circuits électroniques. Ce système graphique a été conçu comme aide pédagogique utilisable par ceux qui donnent un cours sur les circuits numériques d'ordinateurs et par leurs élèves. On donne à l'étudiant la possibilité d'essayer un circuit de logique séquentielle d'une complexité moyenne.

On met plus l'accent sur la structure et la génération de formats pour le rendement graphique que sur les applications elles-mêmes.

Pour un action réciproque l'étudiant peut désigner la position d'une porte avec le crayon électronique ou avec les boutons-poussoirs de la console de visualisation. Une caractéristique spéciale de ce système est sa capacité de présenter sur l'ecran le valeur logique, de tout élément intérieur d'un circuit numérique.

GRAPHICS SIMULATING LOGIC CIRCUITS

Rashpal Ahluwalia* & Irene Gargantini
University of Western Ontario

1. Introduction

The present paper is a description of the Interactive Logic Circuit Design system (ILCD) developed at the Department of Computer Science at Western.

ILCD was mainly designed to offer laboratory sessions on a man-machine interactive basis, in order to complement the lectures of the course on Computer Logic. Our facilities consisted of a PDP-15/20, equipped with 16K core memory, a graphic processor and a VT04 display unit. Our main goal was to develop a descriptive language easy to understand and use. The most recent version of ILCD consists now of several service routines, part of which are written in assembly language, and part in Fortran IV.

The natural flow in presenting a Computer Logic course is first to introduce the combinational gates, and then some sort of memory device, so that the students can combine them in order to obtain circuits closely related to the familiar world of arithmetic. Thus, subpicture files were created for AND-, NOT-, OR-, NAND-elements, J-K flip-flops and shift-

*Supported in part by the National Research Council of Canada (grant No. 1304T91).

registers. Each of the above elements was simulated as an independent unit in the sense that a corresponding directory was created for its display. Sequential circuits can then be designed by assembling the logical elements mentioned above.

Man-machine interaction takes place through a combination of the following devices: light-pen, teletype, LK-35 keyboard and pushbuttons of the display unit. For the display of a logical element or of a connecting line, the user can either supply the coordinates of the desired location via teletype or he can point the light-pen directly to the required position on the screen.

ILCD basically performs two operations, namely the placement of elements on the graphic display unit and the logical evaluation of gates. We correspondingly can distinguish two categories of data structures- the topological and the functional data structures. The first set is involved in the creation of the image of a circuit, the second establishes a connection between some truth tables. The distinction between the two categories appears to be particularly important in our case, because of the extension of the system to fault detection simulation, when we want to be able to change the logical value at some terminals while retaining the remaining structure.

The dimensions of the graphic display (9-1/4" by 9-1/4") have posed a fan-in problem in the sense that the number of inputs (and, in general, of connecting lines) had to be kept low enough to allow a good visibility on the screen.

Finally, just to give an idea of the capabilities of the system, we mention some of the circuits which have been implemented: a two-step algebraic adder, a circuit for multiplication using single length registers, a circuit for division using the principle of comparison (three digits), and a circuit for correcting single errors in the Hamming code with three checking digits. Slides will illustrate some of the circuits.

2. General Characteristics

The logical elements consist of AND-, OR-, NOT-, NAND-gates, full adders, J-K flip-flops and shift-registers. The geometrical elements, which create the connection between any two gates, consist of the input and output elements, joining lines and branching points (nodes). If the light-pen is used to designate a location, some adjustment takes place in order to guarantee straight lines as connections. All gates and the input component are characterized by the coordinates of their outputs, while the output element is defined through the coordinates of its input.

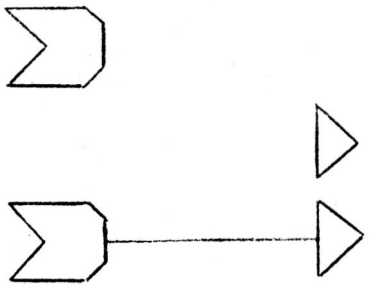
Similar gates are serially numbered according to the order of their appearance on the screen; all combinational elements (other than the inverter) are assumed to have three input terminals numbered 1, 2 and 3.

As an example we list below the statements relative to the display of the OR-gate numbered "1", followed by the inverter numbered "2".

(1) OR X1 Y1

(2) INV X2 Y2

(3) CON OR 1 INV 2



(4) CALC OR 1

(5) CALC INV 2

The specification of the coordinates in (1) and (2) are not necessary if the light-pen is used. The statements "CALC(ULATE) OR 1" and "CALC(ULATE) INV 2" do not have any effect on the screen until some values are entered as inputs; (for instance, through the pushbuttons). Then all output values of the gates appear inside the elements themselves. Computation is possible only when the circuit is logically complete.

For the convenience of the user, the images of all logical elements are continuously displayed in the offset area of the screen during the production of a display file. Also, all user commands are checked for syntax errors and appropriate error messages are printed out.

3. Topological Data Structure

When the user requests a logical component to be displayed at a certain location, the coordinates of that position are stored in an appropriate table. Such a table has two entries, one for the X-coordinate and one for the Y-coordinate. Since similar logical elements are sequentially numbered, so also are their corresponding tables.

The main display file created by ILCD contains three special features dealing with:

(i) branches to subpicture files for drawing a logical element,

(ii) production of lines connecting any two logical components,

(iii) branch to the subpicture file containing the logical values of all the elements.

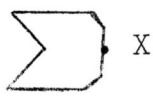
(i) The shape of each logical component is, of course, predefined. Thus, when an element has to be displayed, the corresponding subpicture file is called for. For example, if the commands "OR X1 Y1" and "INV X2 Y2" are typed in, the interaction between the main display file and the required subpicture files would take place as follows:

Subpicture File

SET POINT AT X1 Y1

Branch to OR Subpicture

```
CALL LINE (0,12,1,POR(1))
CALL LINE (-8,8,1)
CALL LINE (40,0,1)
CALL LINE (20,-20,1)
CALL LINE (-20,-20,1)
CALL LINE (40,0,1)
CALL LINE (8,8,1)
CALL LINE (0,12,1)
```

The corresponding drawing would be  X1 Y1

Subpicture File

SET POINT AT X2 Y2

Branch to INV Subpicture

```
CALL LINE (-20,20,1,PINV(1))
CALL LINE (0,-40,1)
CALL LINE (20,20,1)
```

The corresponding drawing would be  X2 Y2

(ii) The production of connecting lines is a part of the main display file. When a line is displayed, its coordinates are stored in an appropriate four entries table. Two of the entries are used to specify the coordinates of the source and the remaining two are used to specify the coordinates of the

destination.

(iii) The logical values of all elements are displayed through a subpicture file, by branching from and back to the main display file. Thus the functional values can be computed without changing or re-defining the topological structure of the circuit.

4. Functional Data Structure

Since our final goal is to simulate the design of logical circuits, ILCD must also create directories for the evaluation of Boolean functions. This part is graphical in nature too, since the logical values are displayed on the screen inside the corresponding logical elements. This requires linkages among:

(1) the directory which produces the Boolean values,
(2) the main display file.

Directories performing such linkages consist of:

(i) an array for the logical connections,

(ii) arrays for the input-output values of a logical element,

(iii) an array for the control of the time sequence of the circuit.

The array mentioned in (i) performs a dual function, namely:

(a) the computation of values at the terminals,

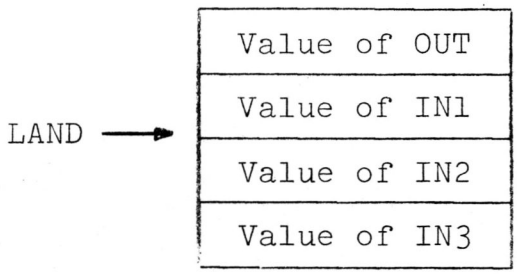
(b) the evaluation of the state of an element.

Such an array has four entries as shown below:

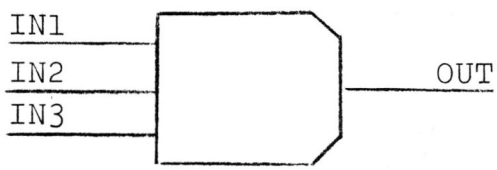
Word 1	Code for Source
Word 2	Source Number
Word 3	Code for Destination
Word 4	Destination Number

If Word 3 is different from zero, then Word 1 and Word 3 specify the source and the destination in coded form, while Word 2 and Word 4 specify the source and destination number. If Word 3 is zero, then the logical value of the element, defined by Word 1 and Word 2, is calculated.

Arrays mentioned in (ii) are basically logical tables generated for the computation of the input and output values of the logical elements. They vary in size and format according to which logical component they refer. Since it would take too much space to describe all the arrays, we simply give an example. The logical table associated with an AND-gate (LAND) has four entries:



LAND supplies the values at the terminals denoted by OUT, IN1, IN2, IN3 as it is shown in the diagram below:



Finally, a short comment on the array mentioned in (iii). Such an array is generated for the purpose of synchronizing the various parts of a sequential circuit. The control of the time sequence is achieved by specifying all possible values of the input elements at integral times.

Acknowledgment. The authors wish to thank Mr. Peter Gredley for his initial contribution to the simulation of logical circuits on a non-interactive basis.