COMPUTER-AIDED LAYOUT INTERCONNECTION EXTRACTION OF CELL-STRUCTURED INTEGRATED CIRCUIT MASKS

V. SZÉKELY, P. BAJI, M. KERECSEN-RENCZ, M. KOLTAI*, I. KÓNYA and F. MASSZI

Department of Electron Devices,
Technical University, H-1521 Budapest

Received Oct. 22, 1984
Presented by Prof. Dr. K. Tarnay

Summary

The program CELLINEX presented in the paper finds the cellular interconnections from the layout of cell-structured integrated circuits. From this the logical description of the circuit is generated and it is checked whether the realized interconnections are permitted or not and whether there are trivial lacks or not. The paper describes the characteristics of the program and the most important algorithms. Some kinds of documentation of the results are presented.

Introduction

Faultless IC masks: one of the basic problems of integrated circuit fabrication. Debugging possible mask errors and updating the whole costly and time-consuming mask production unpleasantly spoils the effectiveness of design and construction efforts, especially by custom design circuits where decreased costs and short production time are basic conditions. Consequently, all possible methods should be used for checking masks during design work to avoid actual production of faulty masks.

Final result of computer-aided layout design work consists of several computer files defining the mask geometry, data of its shapes etc. The actual mask production is an automatic procedure:

— a computer-controlled pattern generation of master masks from the above-mentioned files,
— reduction of the master mask to the final size, and the step and repeat procedure,
— production of actual working mask sets.

* Department of Theoretical Electricity

All possible checking work should be done before the first step, i. e. on computer data files.

Checking possibilities of layout data can be divided into two parts:

Design-rule checking: "syntactical" control of the layout, whether all design rules are kept in the mask or not. This task is mostly done by special design-rule check (DRC) programs.

— *Layout recognition*: "semantical" control of the mask, whether the syntactically correct mask represents the wanted circuit or not. All layout shapes are functionally recognized: what kind of components and/or interconnections do they realize.

Layout recognition is classified by the smallest, but still recognized part of the circuit. If this part is a component (resistor, transistor etc.), recognition work is done on *component level*. If recognition is restricted to some bigger parts of the circuit, typically to cells of cell-structured integrated circuits, it is called *cell-level* recognition. In this case only cells and their interconnections are to be recognized.

This paper presents a possible realization of the computer-aided cell-level recognition, the program CELLINEX (CELL INterconnection EXtraction) developed by our staff during the spring term of 1983.

Characteristics of the program CELLINEX

The task of the program CELLINEX is layout checking of digital LSI integrated circuits [5]. This program has close relations to the cell-library maintenance program CELLIB developed by our staff earlier [4]. The program CELLINEX can recognize layouts consisting of cells and their interconnections. Layout data are handled in a general layout-oriented graphical language. The attention is focussed on custom- and semicustom-design integrated circuits. The limits are: maximum 400* cells, 2000 data-paths, 5000 shapes holding interconnection functions, 20 layers among which interconnection layers are at will. Both one-layer and multi-layer interconnected circuits can be recognized. A very effective partitioning technique and carefully designed algorithms provide unusually short running time of the program: 4–5 minutes of CPU-time in a mega-mini machine configuration for the maximal size of circuits.

Presentation of the results are planned to fulfill possibly all designer's requirements. Just a few examples: recognized circuits can be analyzed by logical simulation programs or data-pathes of the layout can be displayed on graphical screen.

* now it is already 1000

Algorithms

The first impression can be that the layout interconnection recognition is free from serious algorithmic problems. The main problems are solved by simple analytical geometric algorithms: polygon coincidences with windows etc. The essence of the problem is the tremendous set of data on which these simple algorithms operate. No straightforward methods can be applied: the data files must be kept on mass storage and the program should handle files of 0,5–1 Mbyte in case of the most complicated layouts.

As a consequence of big data sets, running time of so-called two-parameter geometrical operations (e. g. finding common parts of shapes) increases very much, because each shape should be compared to all others. If an examined mask layer consists of n shapes,

$$\binom{n}{2} = \frac{n(n-1)}{2} \approx \frac{n^2}{2} \qquad (1)$$

comparisons should be made, i. e. the quantity of required operations is proportional with n^2. Considering 2000 shapes, if a single comparison takes about 200 μsec CPU-time, performing such a two-parameter operation should last for 6–7 minutes, resulting a running-time for the whole program in the range of hours. These difficulties give reason to detailed discussion of algorithmic problems.

Partitioning technique

Problems caused by the big quantity of shapes can be reduced by partitioning the layout in order to separate operations. Let us consider a layout of n shapes partitioned to k partitions (Fig. 1). If the distribution of shapes is equal among partitions and the problem of shapes cutted by partition boundaries is not considered, all partitions will consist of n/k shapes, and the quantity of needed operations:

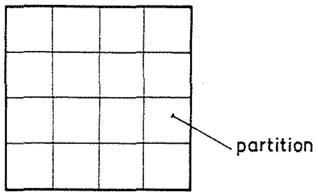


Fig. 1. Layout partitioning

$$k\left(\frac{n}{k}\right)^2 \cdot \frac{1}{2} = \frac{n^2}{2k} \tag{2}$$

i. e. k times less than in case of no partitioning. Considering our example, the mentioned 6-7 minutes of CPU time should decrease to a few seconds.

To increase unlimitedly the quantity of partitions is a thoughtless conclusion, however. If partitions become smaller than mean shape's size, partition boundaries cut more and more shapes to two or even more parts, consequently the total number of shapes will increase, and more partitions also require more computer time for administration. There is an optimum after which more partitioning means no more advantage [2]. (Detailed experimental experiences of this problem are discussed in the next part.)

Practical problems of partitioning

A one- or two-dimensional, fixed boundary, cutting partitioning method is used in the program CELLINEX. The quantity of partitions is given by the user, being maximum 16 in both of x and y directions (maximum 256 partitions). The "fixed boundary" term means the positions of partition boundaries are independent from mask pattern. The term "cutting" means the partition boundaries cut into parts the intersecting shapes.

The algorithm of partitioning is presented on Fig. 2. All operations are performed first for vertical partition boundaries. Steps of the algorithm are as follows:

— In all points, where a polygon edge cuts a partition boundary, two excess polygon vertices are marked out with same coordinate data. One of the new vertices belongs to the partition in one side of the boundary, while the other one to the other (Fig. 2.b).

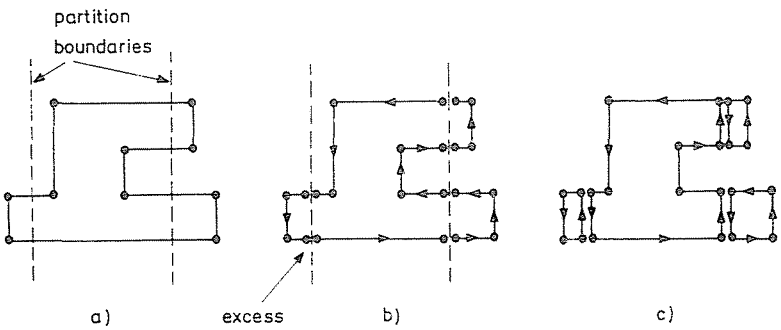


Fig. 2. Explanation of the partitioning algorithm. Pairs of excess vertices are actually situated at same coordinate points, here they are separated only for demonstrational purposes

— Linked lists are built from all polygon vertex coordinates.
— Y-direction order of excess vertices are determined along all partition boundaries.
— Forward pointers of back excess neighbouring vertices are exchanged (Fig. 2.c).

Horizontal partition boundaries are treated the same way.

Special attention is paid to shapes having vertices or edges on exact partition boundaries, making the partition algorithm more complex. An additional problem: if edges are permitted on exact partition boundaries,

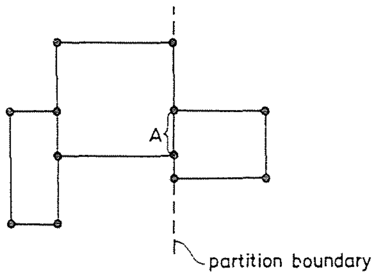


Fig. 3. If the contiguity of shapes are examined only inside the partitions, the contact A cannot be recognized

recognition of adjoining shapes is no more partition-local problem (see the case in Fig. 3). To avoid difficulties the following solution was chosen. All shape coordinates are integer numbers (graphical mask-description languages use grid for the possible coordinate values). Internal data structures of the program CELLINEX double the density of this grid — all vertex coordinates are multiplied by two, consequently all values will be even numbers. If partition boundaries are put to odd coordinate values, no shapes will have edges or vertices on partition boundaries.

Sorting

It is advisable to sort partitioned shapes, based upon the order of partitions. Result of this sorting is the file of ordered partitioned shapes consisting of the shapes and contact windows data of the ordered partitions. While processing partition data the records of this file are sequentially read to a big (80 kbyte) main memory buffer. Linked lists are generated to chain different mask levels before further processing.

The algorithm of "point in a polygon"

This algorithm is very important. Its task: to determine whether a point is inside (or on the border) of a polygon, or outside. The algorithm is of double importance: first of all this is the key of recognizing electrical contacts of two polygons via contact windows. (All windows are represented with their centre points: two polygons are considered as electrically contacted if both include the centre point of the same window.) Secondly, this algorithm is responsible for the labelling of polygons: a label text belongs to that polygon in which the starting point of the text is included.

The essence of the algorithm is shown on Fig. 4. Its steps:

— An *y* direction imaginary straight line is drawn through the point.

— Intersections of this straight line and the polygon contour are determined.

— If there are odd number of intersections below the point, the point is inside, otherwise (even intersections) outside.

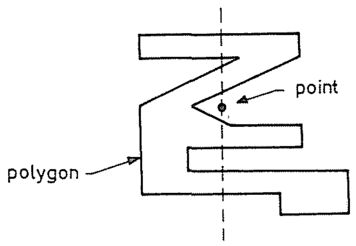


Fig. 4. Explanation of the algorithm "point in a polygon"

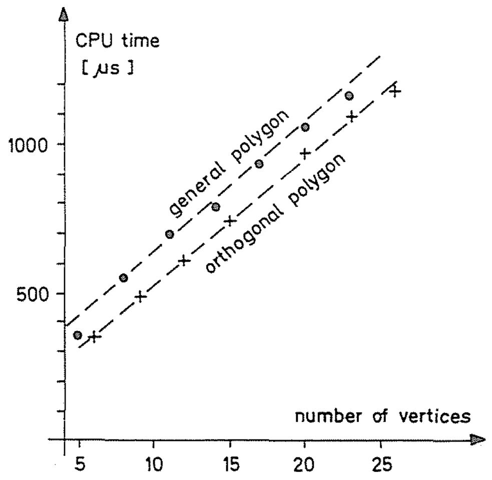


Fig. 5. CPU times of the algorithm "point in a polygon"

(Excess steps processing points situated exactly on polygon contours are not detailed here.)

CPU time statistics of a subroutine realizing this algorithm is presented in Fig. 5.* Trivial cases are not included: running time of these is 50 μsec.

Possibilities for simplifying the operations

In order to quicken processing all possibilities for simplifying the operations are used; in some trivial cases even no operations are made. To enable simplifications, inner representation of polygons include

— vertex coordinates of closing rectangles of all polygons,

— a code-number flagging specific polygons (e. g. rectangles, orthogonal polygons).

It is worth to store these few redundant data because operations can be simplified on their basis. Algorithms of "point in a polygon" or "adjoining polygons" start with the examination of the enclosing rectangle. Negative decisions (point being outside, separate polygons) can be done in many cases in this first step saving computer time etc. The program utilises most complex algorithmic formulae only in case of general polygons.

Method of interconnection recognition

The first step is to discover electrically connected polygons. If polygons are on separate layers, electrical contacts can only be realized via windows, thus the "point in a polygon" algorithm is used for the contact recognition. If polygons are on the same layer, the "adjoining polygons" algorithm is used to determine if they have a contact or not. A special interconnection file is maintained to store discovered contacts: all contacts are represented with serial numbers of the two polygons in question.

The next step is to search equipotential groups of shapes. The algorithm is as follows:

— a pointer is assigned to each shape to locate the next shape in the same equipotential group,

— at the beginning all shapes point to themselves (i. e. all groups consist of a single shape).

— contacted shape pairs are taken from the interconnection file one after the other, and pointers of each pairs are exchanged.

* All CPU time data was measured on the SEMCON configuration of Microelectronics Works. Budapest.

The algorithm contains some excess steps for handling multiple interconnections, loops, etc., which are not given in details here.

After the above procedure all equipotential groups of shapes are discovered: the layout is recognized, only results are to be documented.

4. Partitioning and CPU time

Experiments were made to examine the relation between circuit size and CPU time needed, and to examine how the quantity of partitions influences the CPU time of some algorithms of the program. Basic element of the examined circuit was a small subcircuit consisting of three cells, 24 interconnection shapes, 7 windows and 20 shape labels. This subcircuit was then multiplied in both directions in a matrix manner to produce circuits of different sizes. At first, total CPU time versus circuit size was measured with (1) constant partition quantity, and (2) proportionally increased partition quantity with the size of the circuit examined. Results are plotted in Fig. 6. In case of given quantity of

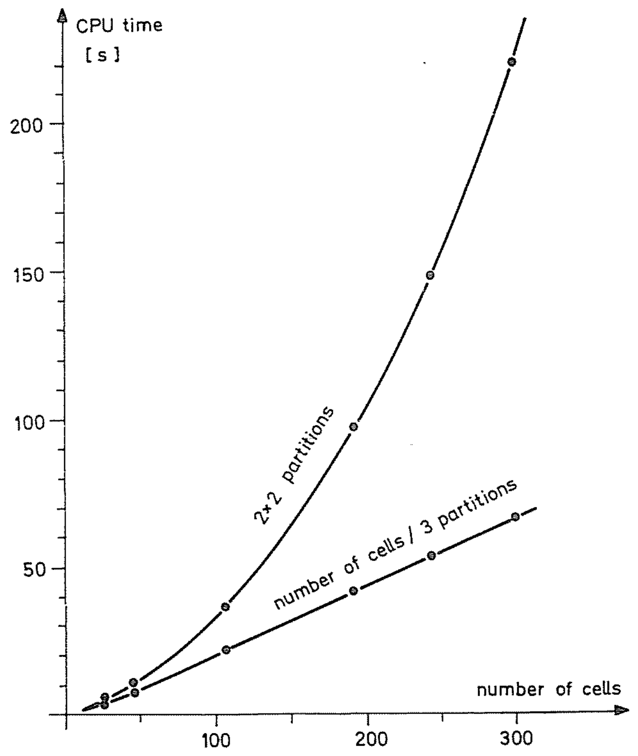


Fig. 6. CPU time versus cell quantity (merge operation not included)

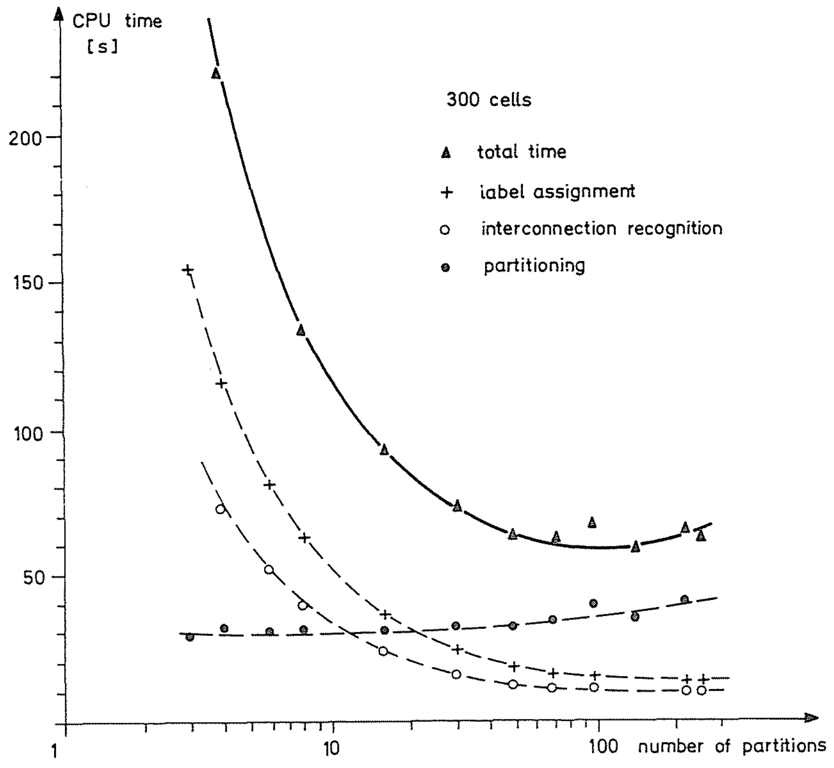


Fig. 7. CPU times of algorithms versus partition quantity

partitions there is a quadratic relation between CPU time and circuit size. If the quantity of partitions is increased, the relation is more linear.

The next examination was made on a big circuit of 300 cells: CPU-times of some algorithms were examined versus increase of partition quantity. Results are shown in Fig. 7. It can be seen that the algorithms of label assignement and interconnection recognition require monotonously decreasing computer time if the quantity of partitions is increased. This advantage is partly spoiled by increasing time consumption of the partitioning algorithm itself. The total CPU time has an optimal, minimal value at about the partition quantity of 100. Note the importance of distribution of a given partition quantity between x and y direction partitioning: best results were obtained if partition quantity was about the same in both directions, in conformity with earlier theoretical discussion of the problem [3].

5. Docun entation of results

Results of layout recognition can be used in a wide variety of forms to check the layout. The program CELLINEX supports as much checking methods as possible.

Errors, warnings

The recognition can discover specific layout errors which are definitely due to designer's mistakes. These errors are flagged with error messages. In other cases there is only a probability of an error: then warnings are printed. Typical irregularities flagged by the program:
— unused window,
— open cell-output,
— loop in an equipotential group of shapes,
— short-circuit (supply voltage to cell output etc.)
— functional irregularities in the interconnection of cells (no cell input connected to signal path, more — not tri-state — outputs connected),
— useless parts in layout (shapes connected to a single cell or to no cells).

Cell listing

Printout is shown on Fig. 8 (for a single cell). Cell outputs with labels and a character code referring to their function are listed, together with their interconnections.

```
 1-JUL-83          CELL INTERCONNECTION EXTRACTION PROGRAM          PAGE=  10
==============================================================================

                                   CELL LISTING
                                   ============

                          DESIGN FILE NAME=  CELL

CELL NO.=   1,   NAME=WAND2   ,   RELATED INTERCONNECTIONS=

                        CELL NODES          I    INTERCONNECTION
                 NO.    TYPE   IDENTIFIER   I    NO.  IDENTIFIER
                 ----------------------------------------------
                  1.    INP+       BE1      I     1.    MASODIK
                  2.    INP+       BE2      I     2.    ELSO BEM
                  3.    OUT+       OUTP     I     3.    -
                  4.    UGG-       TAPF     I    40.    -
                  5.    UDD-       TAPF     I    41.    -
                  6.    GND-       FOLD     I    42.    -
                 ----------------------------------------------
```

Fig. 8. Cell listing printout detail

Interconnection listing

This is the inverse of cell listing (Fig. 9). All interconnections are listed: which are the cells and how they are connected to them.

```
                              INTERCONNECTION LISTING
                              =======================

                 DESIGN FILE NAME=  CELL


INTERCONNECTION NO.=  31   IDENTIFIER=KIMENET ,   CONNECTED CELLS=

                   C E L L         I          N O D E
                 NO.     NAME      I    NO.     TYPE    IDENTIFIER
                 ---------------------------------------------------
                 21.    NNOR2      I     3.     OUT+       OUTP
                 ---------------------------------------------------


            ****  ERROR  ****     NO INPUT OR TRI-STATE I/O CONNECTED TO
                                  THE TREE BEING NOT UDD,UGG,USS,GND
            ****  ERROR  ****     ONLY A SINGLE CELL PIN IS CONNECTED
                                  TO THIS TREE
```

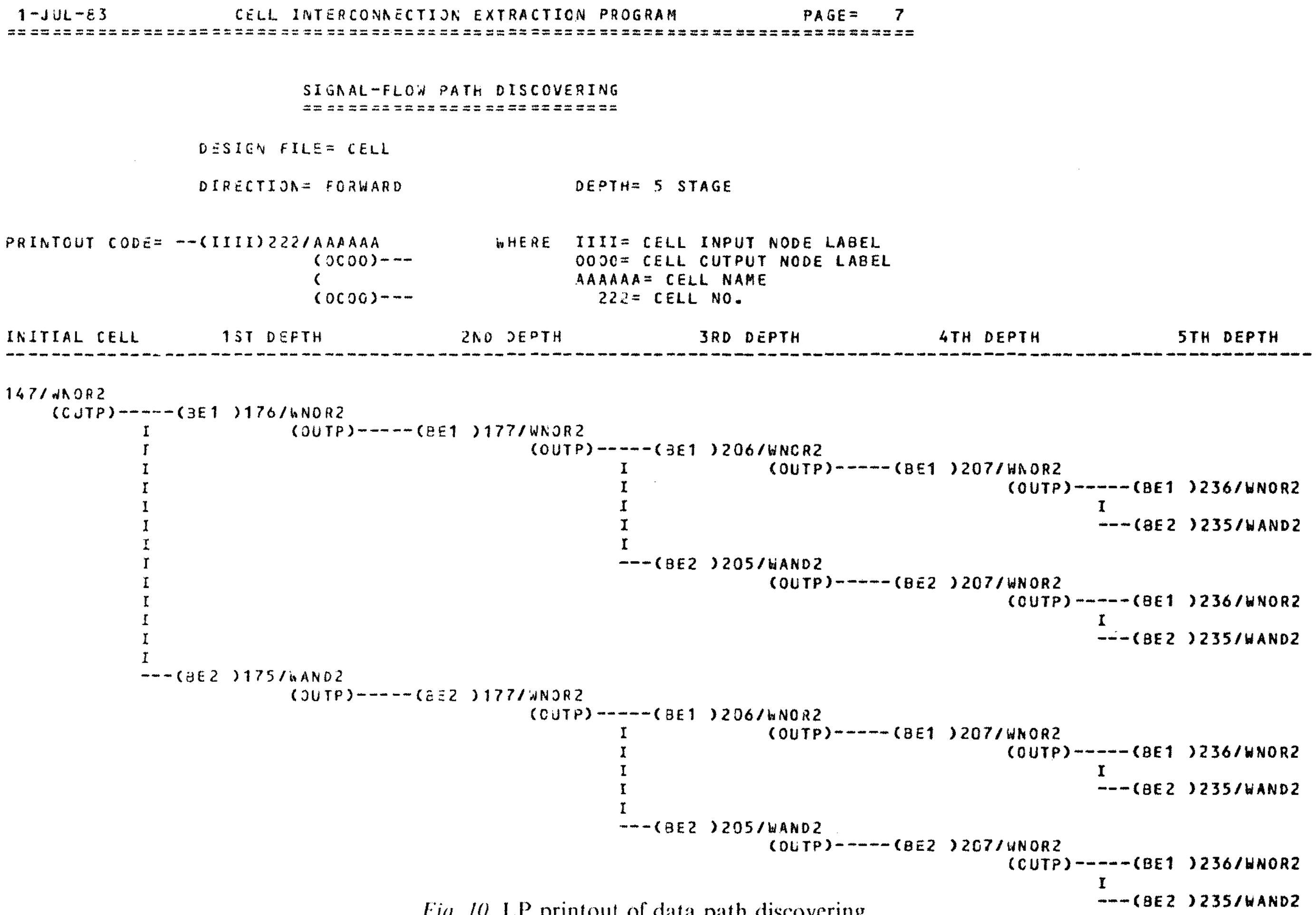
Fig. 9. Interconnection listing printout detail

Data-path discovering

The user has a possibility to follow a logical signal through succeeding cells. Starting cell and depth of discovering are to be given by the user. Results of documentation are shown in Fig. 10. The subroutine realizing this kind of documentation searches the outputs of the starting cell, then searches cells connected with their inputs to the starting cell's outputs, etc. The procedure can be done both in forward and reverse directions (relative to the signal-flow).

Graphical interactive data-path discovering

The former procedure can be done on a graphical display. The layout, or part of it, but only outlines of the cells are displayed. By putting the cursor to the starting cell, all of its forward (or reverse) interconnection paths will be graphically shown. Repeating the procedure any data paths can be traced (Fig. 11).



```
 1-JUL-83          CELL INTERCONNECTION EXTRACTION PROGRAM         PAGE=   7
============================================================================


                         SIGNAL-FLOW PATH DISCOVERING
                         ============================

                DESIGN FILE= CELL

                DIRECTION= FORWARD                      DEPTH= 5 STAGE


PRINTOUT CODE= --(IIII)222/AAAAAA        WHERE  IIII= CELL INPUT NODE LABEL
                         (OC00)---              OO00= CELL OUTPUT NODE LABEL
                         (                      AAAAAA= CELL NAME
                         (OC00)---                222= CELL NO.

INITIAL CELL      1ST DEPTH           2ND DEPTH           3RD DEPTH           4TH DEPTH           5TH DEPTH
-------------------------------------------------------------------------------------------------------------

147/WNOR2
    (OUTP)-----(BE1 )176/WNOR2
           I            (OUTP)-----(BE1 )177/WNOR2
           I                                (OUTP)-----(BE1 )206/WNOR2
           I                                       I            (OUTP)-----(BE1 )207/WNOR2
           I                                       I                                (OUTP)-----(BE1 )236/WNOR2
           I                                       I                                       I
           I                                       I                                       ---(BE2 )235/WAND2
           I                                       I
           I                                       ---(BE2 )205/WAND2
           I                                                    (OUTP)-----(BE2 )207/WNOR2
           I                                                                        (OUTP)-----(BE1 )236/WNOR2
           I                                                                               I
           I                                                                               ---(BE2 )235/WAND2
           I
           ---(BE2 )175/WAND2
                        (OUTP)-----(BE2 )177/WNOR2
                                            (OUTP)-----(BE1 )206/WNOR2
                                                   I            (OUTP)-----(BE1 )207/WNOR2
                                                   I                                (OUTP)-----(BE1 )236/WNOR2
                                                   I                                       I
                                                   I                                       ---(BE2 )235/WAND2
                                                   I
                                                   ---(BE2 )205/WAND2
                                                                (OUTP)-----(BE2 )207/WNOR2
                                                                                    (OUTP)-----(BE1 )236/WNOR2
                                                                                           I
                                                                                           ---(BE2 )235/WAND2
```

Fig. 10. LP printout of data path discovering

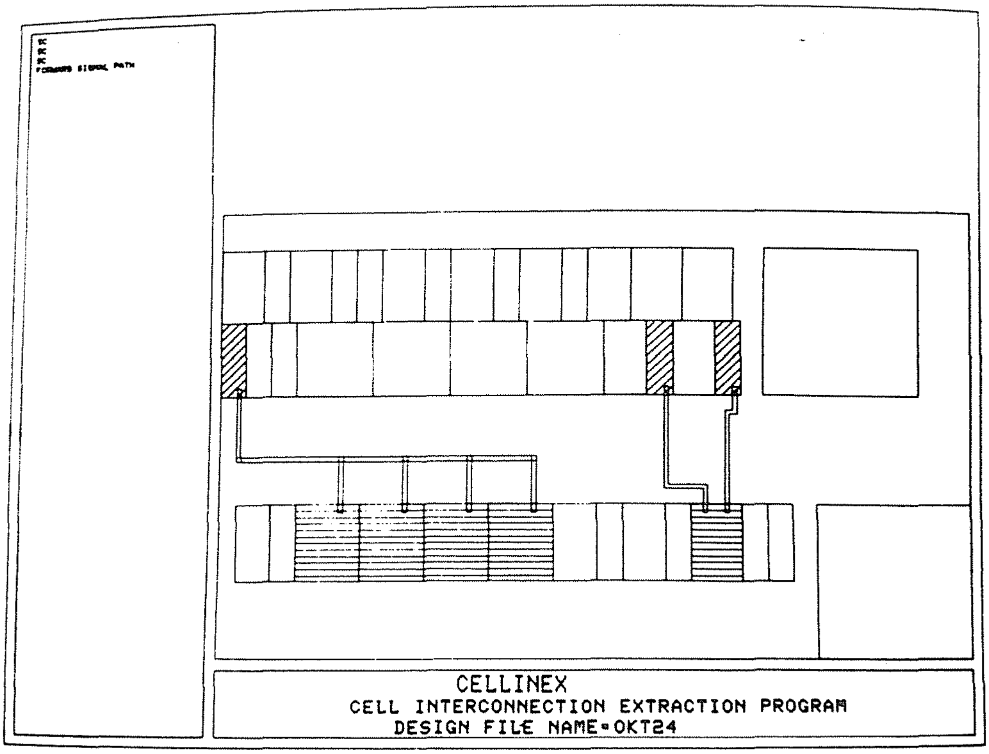


Fig. 11. Data path discovering on graphical screen

Generation of logical circuit description

The program generates logical circuit descriptions of recognized circuits according to the requirements of logical simulation program LOBSTER [6]. Needed logical descriptions of present cells are extracted from the cell library, maintained by the program CELLIB [4]. The logical description

— can be compared to that given by the user,

— can be simulated by the program LOBSTER.

Generation of layout statistics

All essential statistics of examined layouts are printed out, i.e.

— quantities of shapes, windows, signal-paths,

— quantities of cells and cell types,

— maximal and average number of shape vertices,
— maximal and average values of x and y lengths of shapes,
— distribution of shapes among the partitions etc.

Improvements

The program CELLINEX has been used since July 1983 at the Microelectronics Works, Budapest. Possible planned improvements:
— expansion to bigger quantities of cells (max. about 1000, instead of the present 400).
— Calculation of parasitic capacitances. If parasitic capacitances of interconnection areas are calculated, timing verification can be done for the circuits designed.

Of course, further experiences may advise more improvements to be realized in the program.

References

1. SZÉKELY, V.–BAJI, P.–RENCZ, M.–KOLTAI, M.: Computer-aided methods in the design of integrated circuit masks. Report made by the Department of Electron Devices, Budapest 1981. (in Hungarian)
2. BAIRD: Fast algorithms for LSI artwork analysis. *IEEE DATC* pp. 303–311. 1977.
3. FARKAS, G.: Determination of integrated circuit models starting from their topology. Dr. techn. thesis, Technical University, Budapest, 1980 (in Hungarian).
4. SZÉKELY, V.–BAJI, P.–Mrs. KERECSEN-RENCZ, M.–KÓNYA, I.–MASSZI, F.: The program CELLIB—Computer-aided cell library maintenance for the microelectronical design. *Híradástechnika*, Nov. 1983. (in Hungarian)
5. SZÉKELY, V.–BAJI, P.–Mrs. KERECSEN-RENCZ, M.–KÓNYA, I.–KOLTAI, M.–MASSZI, F.: The program CELLINEX—computer-aided cell interconnection recognition. User's manuel, Budapest, 1983. (in Hungarian)
6. JÁVOR, A.–MRS. BENKŐ, M.: The program LOBSTER. User's manuel, Central Research Institute for Physics, Budapest, 1980. (in Hungarian)

Dr. Vladimir SZÉKELY
Dr. Pál BAJI
Dr. Márta KERECSEN–RENCZ
Dr. Mihály KOLTAI
Ilona KÓNYA
Dr. Ferenc MASSZI

H-1521 Budapest