Optimal Layout of CMOS Functional Arrays

TAKAO UEHARA, MEMBER, IEEE, AND WILLIAM M. VANCLEEMPUT, MEMBER, IEEE

***Abstract*—Designers of MOS LSI circuits can take advantage of complex functional cells in order to achieve better performance. This paper discusses the implementation of a random logic function on an array of CMOS transistors. A graph-theoretical algorithm which minimizes the size of an array is presented. This method is useful for the design of cells used in conventional design automation systems.**

***Index Terms*—CMOS circuit design, CMOS functional arrays, computer-aided design, design automation, LSI design automation, LSI layout.**

I. INTRODUCTION

IN integrated circuit design it is possible to implement a logic function by means of a circuit consisting of one or more elementary cells, such as NAND or NOR gates, or by means of a single functional cell.

The basic advantages of functional cells, such as smaller size and better performance, are well known to designers of MOS LSI [1]. Theoretical results about network synthesis with complex functional cells have been reported in [2]–[4]. Some commercial products also take advantage of these properties [5]. However, most designers still use a limited library of cells. For example, NAND gates are often used as the only primitive cell. More details on the physical implementation of complex functional cells have been reported in [6]–[10].

Designers often have no confidence in the performance and merit of more complex cells. In order to overcome these problems, systematic enumeration, layout, and verification of all possible functional cells are required. The useful functional cells are enumerated in [13]. Their number is so large that a systematic layout method is necessary. An array of CMOS FET's is used as the basic implementation strategy and a graph-theoretical algorithm which minimizes the size of the array is presented. This type of array is very useful as a basic building block for conventional design automation systems [11], [12] because it has a rectangular shape with the same height as the other cells. Several examples show the significant merit of functional cells in reducing the space required.

Manuscript received August 14, 1978; revised June 13, 1980. This work was supported by the Joint Services Electronics Program. This paper was presented at the 16th Design Automation Conference, San Diego, CA, June 1979.

T. Uehara was with the Computer Systems Laboratory, Departments of Electrical Engineering and Computer Science, Stanford University, Stanford, CA. He is now with the Computer Science Laboratory, Fujitsu Laboratories, Ltd., Kawasaki, Japan.

W. M. vanCleemput is with the Computer Systems Laboratory, Departments of Electrical Engineering and Computer Science, Stanford University, Stanford, CA 94305.

II. CMOS FUNCTIONAL CELLS

In CMOS it is possible to implement complex Boolean functions by means of n-MOS and p-MOS transistors, rather than by conventional logic elements such as AND, OR, NOT, NAND, and NOR.

An implementation of the EXCLUSIVE-OR function $XY + \overline{X}\overline{Y}$ is shown in Fig. 1, where the designer was required to use NAND gates throughout. An alternative implementation of the same function is shown in Fig. 2 [1], where the designer took advantage of the functional cell which realizes the function $XY + Z$. This approach results in better performance and smaller size than the design of Fig. 1.

It is often difficult for a designer to synthesize such an efficient implementation in the most general case of a network of n-MOS and p-MOS transistors.

III. ENUMERATION

In this paper we will limit ourselves to AND–OR networks realized in CMOS by means of series/parallel connections of transistors. Furthermore, we will require that the topology of the p-MOS and n-MOS sides of the circuit are each other's dual. This restriction in topology is commonly used by designers of CMOS circuits.

The number of functional cells which has series/parallel topology is shown in Table I, where the maximum number of series FET's between the power and the output is designated as the level of a cell. The details of the enumeration are shown in [13].

The delay of a cell mainly depends on the number of levels since it corresponds to the longest path to charge the capacitance. Generally, cells with less than four levels are desirable. To use all of the cells with three levels and some with four levels seems to be a reasonable compromise, although the decision about the usefulness of cells is beyond the scope of this paper. In any case, a systematic design procedure is required to generate a minimal-area CMOS layout for a given Boolean function. The remainder of this paper will discuss a synthesis procedure for Boolean functions in the restricted CMOS technological environment.

IV. BASIC LAYOUT STRATEGY

The basic layout scheme for an arbitrary logic function is given in this section, starting from the corresponding AND/OR (sum of products) specification.

A cell is an array of CMOS transistors, as shown in Fig. 3. It consists of a row of p-MOS transistors and a row of n-MOS

0018-9340/81/0500-0305$00.75

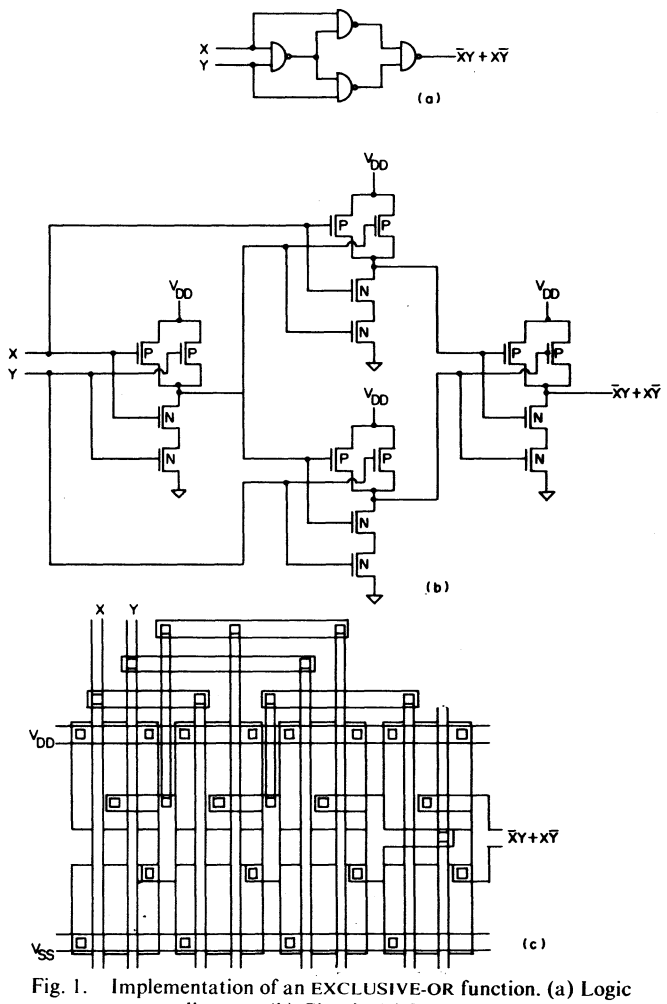


Fig. 1. Implementation of an EXCLUSIVE-OR function. (a) Logic diagram. (b) Circuit. (c) Layout.

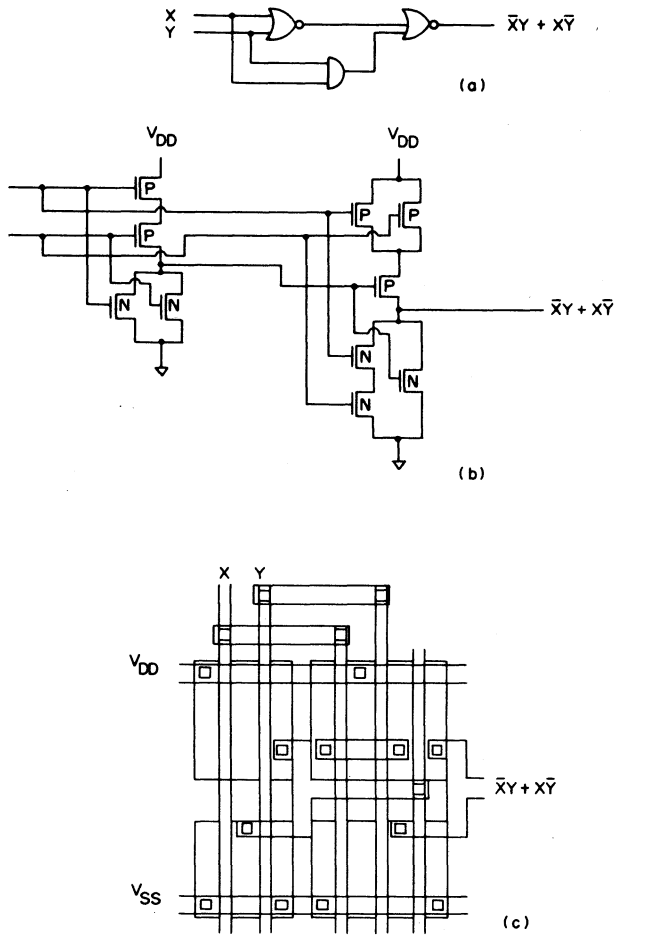


Fig. 2. An alternative implementation of the EXCLUSIVE-OR function. (a) Logic diagram. (b) Circuit. (c) Layout.

transistors corresponding to the p-MOS and n-MOS sides of the circuit, respectively. Because of the requirement that the p-MOS and n-MOS sides are each other's dual, the number of transistors is the same in both rows. We will further assume that the transistors are aligned vertically. AND/OR gates in the logic diagram correspond to the series/parallel connections in the circuit diagram. It is quite clear that for every AND/OR specification of a Boolean function, one can obtain a series-parallel implementation in CMOS technology, in which the p-MOS side and n-MOS sides are each others dual. The number of series/parallel transistors for every AND/OR element is equal to the number of inputs to that element. The dual topology of the p-MOS side and of the n-MOS side are as shown in Fig. 3(c).

A more general topology other than series/parallel can be used in a MOS circuit as in the case of a relay network. The topology of the p-MOS side and the n-MOS side need not be dual in the strict sense. However, the series-parallel connection and the duality are assumed here in order to simplify the problem.

In addition to functional correctness, a designer needs to be concerned with maximizing circuit performance and layout efficiency. Given a Boolean function to be implemented in the restricted CMOS technology under consideration, how can one obtain an optimal layout? In the next section it will become clear that for a given n-MOS/p-MOS transistor network many layout possibilities exist.

TABLE I

NUMBER OF CELLS WITH A GIVEN LEVEL

Number of levels	Number of cells
1	1
2	6
3	80
4	3434

V. Optimal Layout

A graph theoretical algorithm for minimizing the size of a functional array is developed in this section.

A. Preliminary Considerations

Physically adjacent gates can be connected by a diffusion area, although designers should be careful of its performance in the case of high-speed logic. The aluminum connections between neighbors, as in Fig. 3(d), are replaced by diffusion areas, as shown in Fig. 4(a), but the size of the array was not changed. Even in a more sophisticated layout arrangement, the alignment between p-MOS side and n-MOS side is required. The layout can be further improved, as shown in Fig. 4(b), by judicious pairing of sources and drains.

However, the best result is obtained from the alternative circuit of Fig. 5(b), which is logically equivalent to the circuit in Fig. 3(b).

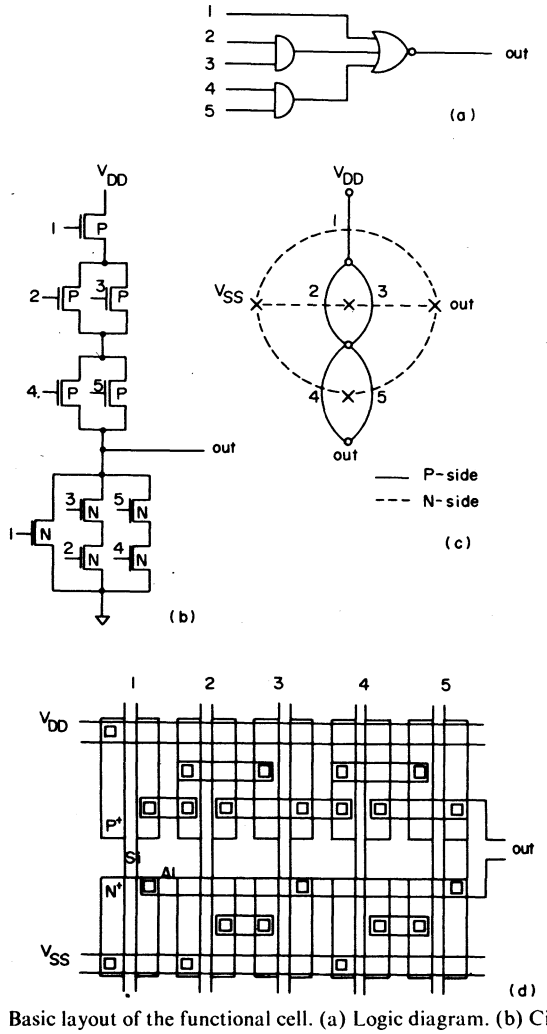


Fig. 3. Basic layout of the functional cell. (a) Logic diagram. (b) Circuit. (c) Graph model. (d) Layout.

Finally, the layout of the functional cell can be optimized, as shown in Fig. 5(d), and the size of this array is almost one half that of the basic layout shown in Fig. 3(d).

In general, the area of a functional cell is calculated as follows: area = width * height, where: height = constant width = basic grid size * (number of inputs + number of separations + 1).

A separation is required when there is no connection between physically adjacent transistors, as illustrated in Fig. 4(b). Since both the cell height and the basic grid size are a function of the technology employed, an optimal layout is obtained by minimizing the number of separations.

B. Graph-Theoretical Algorithm

The graph model of a circuit is defined as follows. A p-side graph and an n-side graph are models of the p-MOS side and the n-MOS side of a circuit, respectively. The p-MOS side graph is defined as follows:

1) every gate/drain potential is represented by a vertex;

2) every transistor is represented by an edge, connecting the vertices representing the source and drain.

The n-side graph can be defined in a similar way. An example of such a graph is shown in Fig. 5.

Because of the restriction on the CMOS circuits under consideration, both the n-side and p-side graphs are series-parallel graphs.

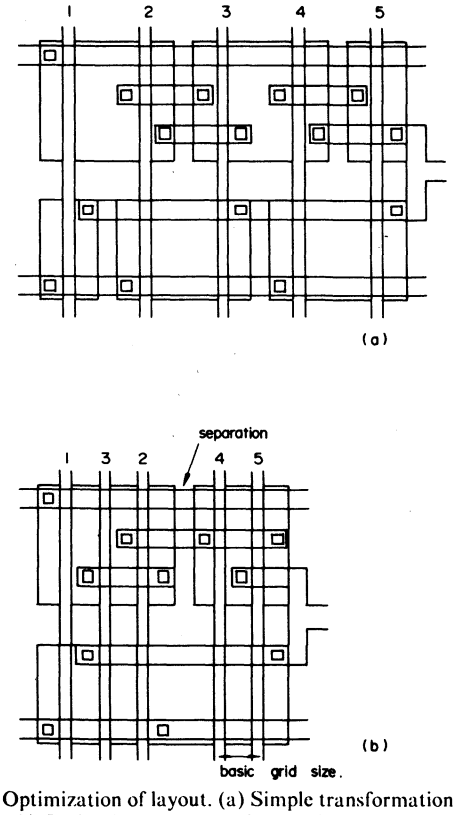


Fig. 4. Optimization of layout. (a) Simple transformation of Fig. 3(c). (b) Optimal arrangement for the circuit in Fig. 3(b).

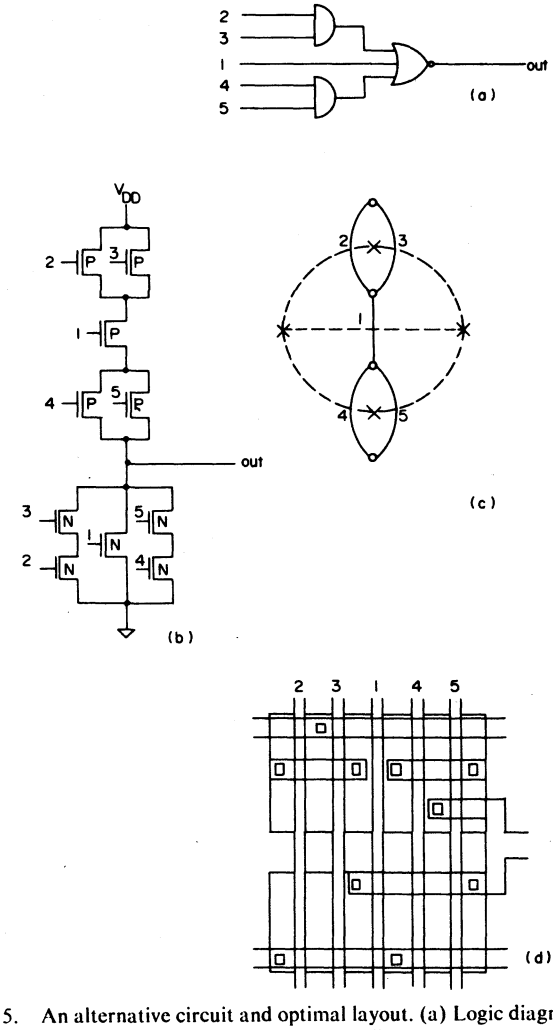


Fig. 5. An alternative circuit and optimal layout. (a) Logic diagram. (b) Circuit. (c) Graph model. (d) Layout.

Edges correspond to transistors in both graphs and they are connected in a series/parallel manner according to the series/parallel connections of transistors in the circuit. The names of input signals are used to label those edges. The p-side graph and the n-side graph are dual by the assumption of Section III and each corresponding pair of edges has a common label.

The following property of the graph model is of interest for the optimal layout of CMOS circuits.

If two edges x and y are adjacent in the graph model, then it is possible to place the corresponding gates in a physically adjacent position of an array and hence, connect them by a diffusion area. In order to minimize the number of separation areas, it is necessary to find a set of minimum-size paths which correspond to chains of transistors in the array. As indicated in Section V-A, such a set will result in a minimal area layout.

If there exists an Euler path [14], i.e., a sequence of edges that contains all the edges of the graph model, then all gates can be chained by diffusion areas. If there is no Euler path, then the graph can be decomposed into several subgraphs which have Euler paths. In the latter case, each Euler path corresponds to a chain of transistors that is separated from another such chains by a separation area.

In order to reduce the size of an array, it is necessary to find a pair of paths on the dual graph models with the same sequence of labels because p-type and n-type gates corresponding to the same input signal have the same horizontal position in the CMOS array. For example, the path $\langle 1, 3, 2, 4, 5\rangle$ of the n-side graph in Fig. 3(c) produces a chain of gates on the n-MOS side, as shown in Fig. 4(b). There is, however, no corresponding Euler path in the p-side graph. Therefore, the gates on the p-MOS side are separated between gate 2 and gate 4, as shown in Fig. 4(b).

On the other hand, path $\langle 2, 3, 1, 4, 5\rangle$ is an Euler path in both the p-side and the n-side graph of Fig. 5(c). Therefore, all gates can be chained together by diffusion areas without any separation areas, as shown in Fig. 5(d).

The general algorithm is shown below:

1) enumerate all possible decompositions of the graph model to find the minimum number of Euler paths that cover the graph;

2) chain the gates by means of a diffusion area according to the order of the edges in each Euler path; and

3) if more than two Euler paths are necessary to cover the graph model, then provide a separation area between each pair of chains.

The problem of finding an optimal layout of a Boolean function using the restricted CMOS design style is then reduced to decomposing the corresponding graph model into a minimum number of Euler paths that cover the graph model.

VI. Reduction of the Problem

In order to find the minimum number of Euler paths, it is possible to take advantage of the reduction method which is illustrated in Fig. 6: an odd number of series or parallel edges can be reduced to a single edge.

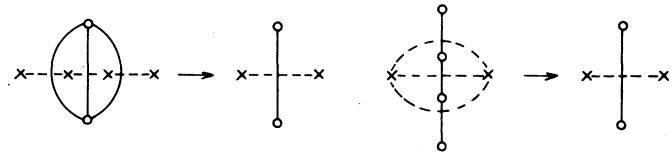


Fig. 6. Reduction of odd number of edges.

Definition: The reduced graph is obtained by replacing an odd number of series (parallel) edges by a single edge, until no further reduction is possible.

Theorem 1: If there is an Euler path in the reduced graph, then there exists an Euler path in the original graph.

Proof: It is possible to reconstruct an Euler path in the original graph by replacing each edge of the Euler path in the reduced graph by a sequence of the original odd number of edges.

Sometimes this reduction makes the problem trivial. For example, the graph model of Fig. 8 is reduced to a single edge and the existence of an Euler path in the graph model is obvious.

Theorem 2: If the number of inputs to every AND/OR element is odd, then

1) the corresponding graph model has a single Euler path;

2) there exists a graph model such that the sequence of edges on an Euler path corresponds to the vertical order of inputs on a planar representation of the logic diagram.

Proof:

1) The CMOS implementation of an AND/OR function has a number of series/parallel transistors that is equal to the number of variables of that function (see Section IV). Since the number of edges in series or in parallel is always odd, the graph model can be reduced to a single edge which is an Euler path itself. So there exists an Euler path on the original path according to Theorem 1.

2) It is possible to construct the graph as follows [see the example in Fig. 7(c)].

a) Start with an edge corresponding to the circuit's output.

b) Select an edge corresponding to the output of a gate and replace it by the series-parallel graph for that gate.

c) Reorganize the sequence of new edges on the Euler path being constructed such that it corresponds to the vertical order of the inputs on the planar representation of the logic diagram. Such a rearrangement of edges in the Euler path is always possible when the number of inputs to an AND/OR element and hence, the number of edges in series or in parallel is odd.

It should be noted that this algorithm assumes that every gate has an odd number of inputs. This is obviously not the case for most AND–OR networks in actual practice.

VII. Heuristic Algorithm

Since the graph-theoretical algorithm of Section V is exhaustive in nature, a heuristic algorithm which takes advantage of Theorem 2 is proposed. Additional inputs called "pseudo" inputs are introduced and the original problem is modified so that every gate in a logic diagram has an odd number of inputs.

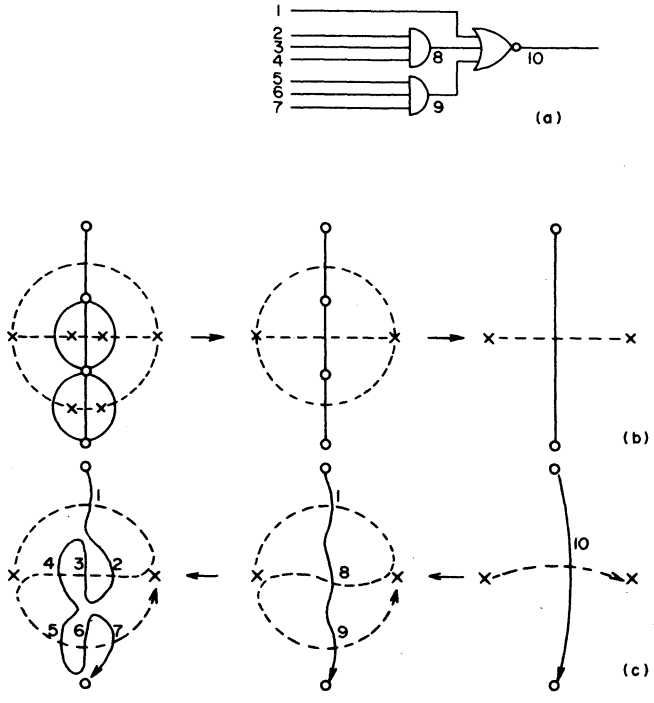


Fig. 7. Application of reduction rule. (a) Logic diagram. (b) Graph model and its reduction. (c) Reconstruction of an Euler path.

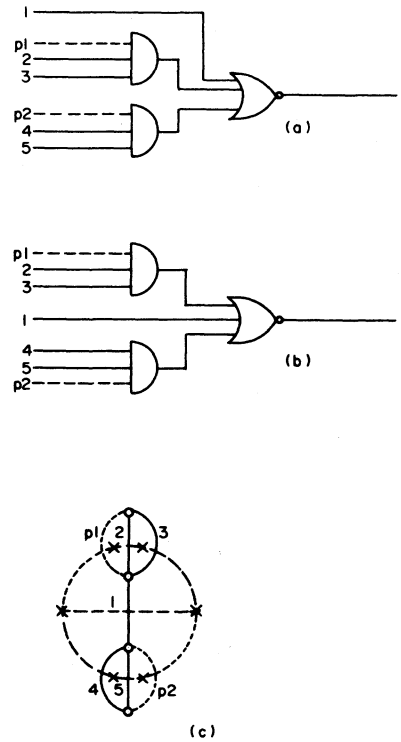


Fig. 8. Application of the heuristic algorithm. (a) New inputs $p1$ and $p2$ are added. (b) Optimal sequence of inputs without the interlace of $p1$ or $p2$. (c) Circuit with the dual path $\{p1, 2, 3, 1, 4, 5, p2\}$.

It is guaranteed by Theorem 2 that there exists an Euler path for this modified problem. This Euler path contains edges corresponding to the original inputs and also edges corresponding to the new "pseudo" inputs, which are possible separation areas. The topology of the circuit should be selected such that the number of separation areas is minimized.

The heuristic algorithm consists of the following steps.

1) To every gate with an even number of inputs a "pseudo" input is added.

2) Add this new input to the gate such that the planar representation of the logic diagram shows a minimal interlace of "pseudo" and real inputs. It should be noted that a "pseudo" input at the top or at the bottom of the logic diagram does not contribute to the separation areas, as illustrated in Fig. 7(b) and (c).

3) Construct the graph model such that the sequence of edges corresponds to the vertical order of inputs on the planar logic diagram.

4) Chain together the gates by means of diffusion areas, as indicated by the sequence of edges on the Euler path. "Pseudo" edges indicate separation areas.

5) The final circuit topology can be derived by deleting "pseudo" edges in parallel with other edges and by contracting "pseudo" edges in series with other edges.

The minimization of the separation areas can be performed on a logic diagram which nicely shows the structure of the series/parallel graph.

Fig. 8 shows the application of this heuristic algorithm to the problem of Fig. 3. The same result as in Fig. 5 is found easily. In general, new additional inputs correspond to separation areas, but in this case they do not actually separate the chain of gates because they are on both ends. An algorithm to construct the minimal interlace is shown in the Appendix.

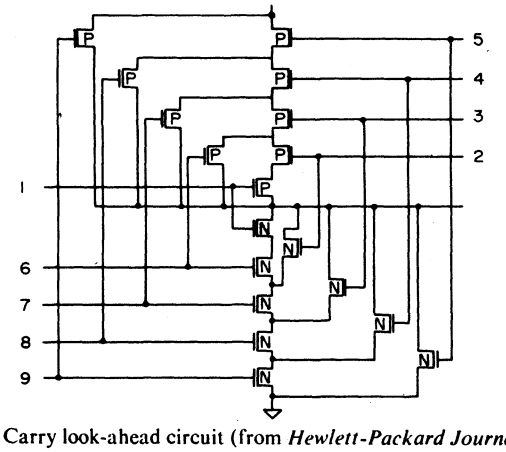


Fig. 9. Carry look-ahead circuit (from *Hewlett-Packard Journal*, April 1977).

This heuristic algorithm does not necessarily give the optimal layout. However, if the resulting sequence has no separation areas, it is the real optimal solution.

Fig. 9 is a 4-bit carry look-ahead circuit from Hewlett-Packard's processor MC2 [5]. The circuit has no Euler path. But the alternative circuit in Fig. 10(c) has an Euler path on the dual graphs. This optimal solution is found easily by the heuristic algorithm, as shown in Fig. 10. Fig. 11 shows that the space for the functional cell is less than one-third of the conventional gate realization.

VIII. Overall Layout Scheme

The layout of a complete integrated circuit involves a collection of functional cells and the optimal bussing of power and logic signals. Several design automation systems have been developed to solve these problems [11], [12].

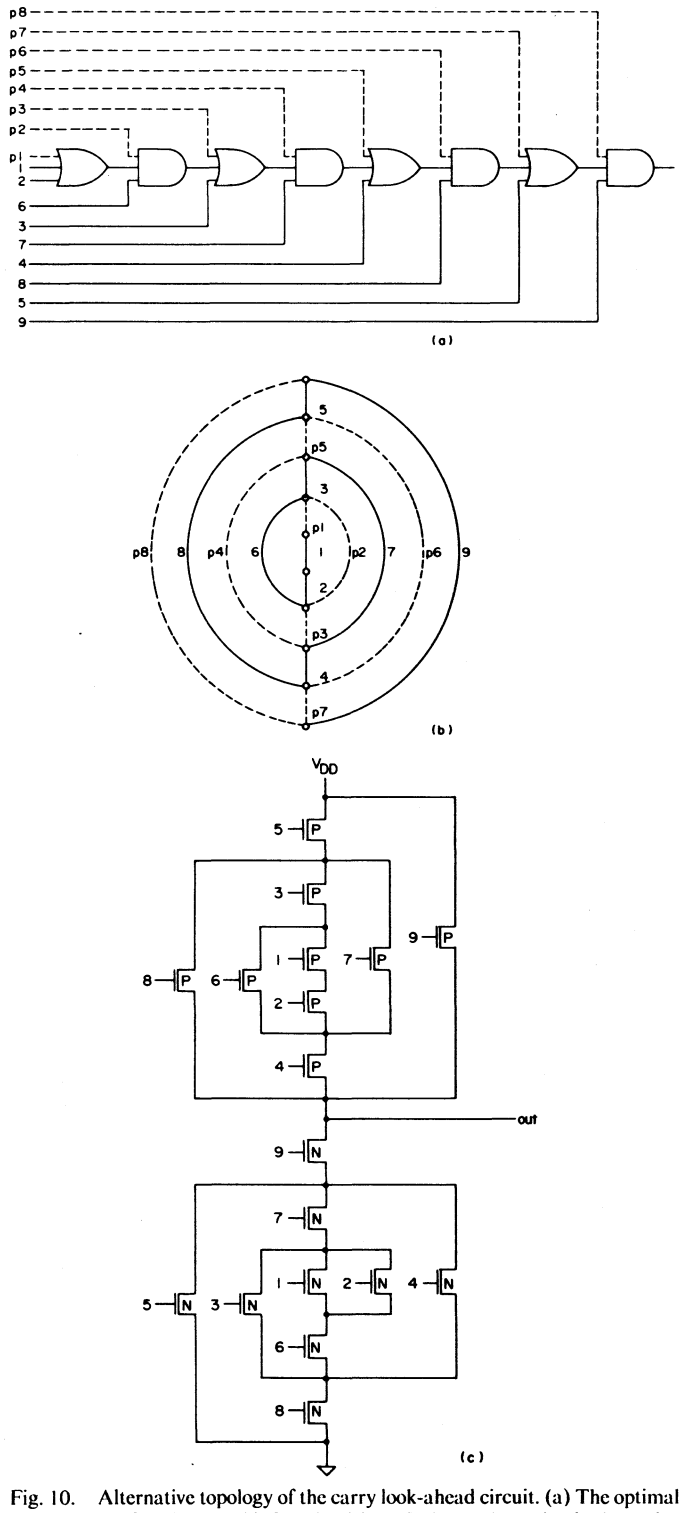


Fig. 10. Alternative topology of the carry look-ahead circuit. (a) The optimal sequence of variables. (b) Graph with an Euler path on the dual graphs. (c) Circuit diagrams.

The functional cells proposed in this paper are potentially useful as the basic primitives to be used in these design automation systems. Because the cells have the same height, the same power connections, and standardized connection points, they can be readily incorporated into existing automated layout systems.

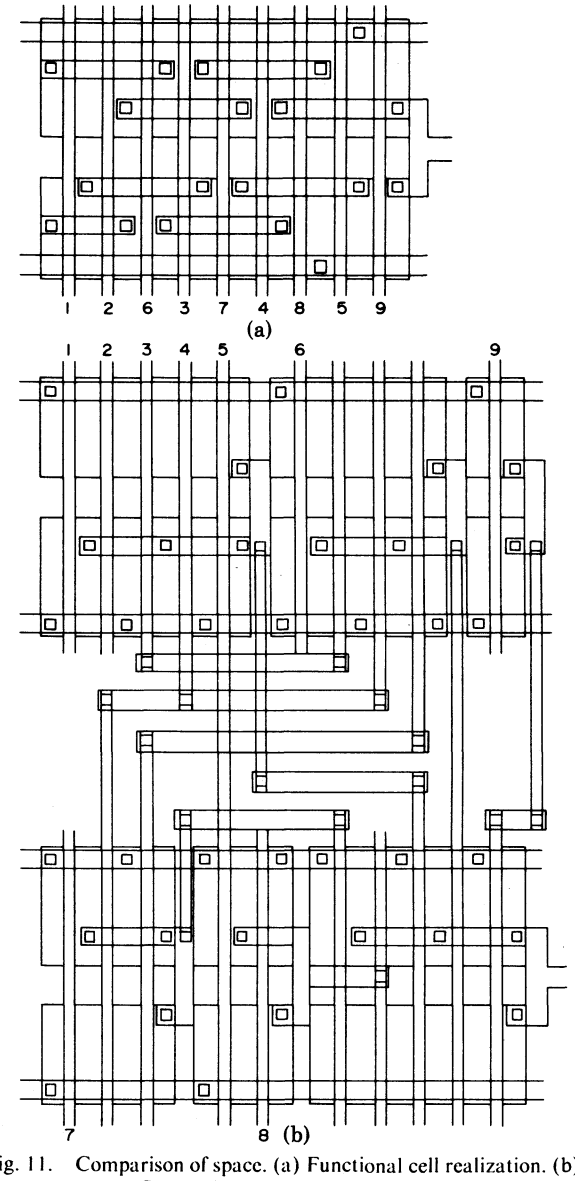


Fig. 11. Comparison of space. (a) Functional cell realization. (b) Conventional NAND realization.

Fig. 12 shows an example of the overall layout scheme using the cells proposed in this paper.

IX. Conclusions

A systematic survey of CMOS functional cells and the enumeration of random logic functions made it clear that there are thousands of useful cells. A systematic method to implement a function on an array of CMOS transistors has been shown and a graph-theoretical algorithm which minimizes the size of the array has been presented. An example shows that the functional cell approach can reduce the space of a conventional NAND gate realization considerably. In general, a significant space reduction can be expected.

The CMOS functional array is also useful as a basic cell for a conventional design automation system. Implementing functional arrays into a MOS LSI design automation system will be considered after further studies of logic synthesis and performance validation.

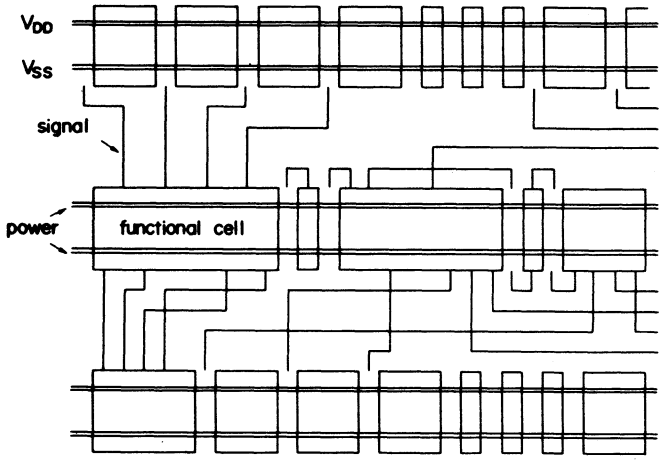


Fig. 12. Example of a row-based layout scheme.

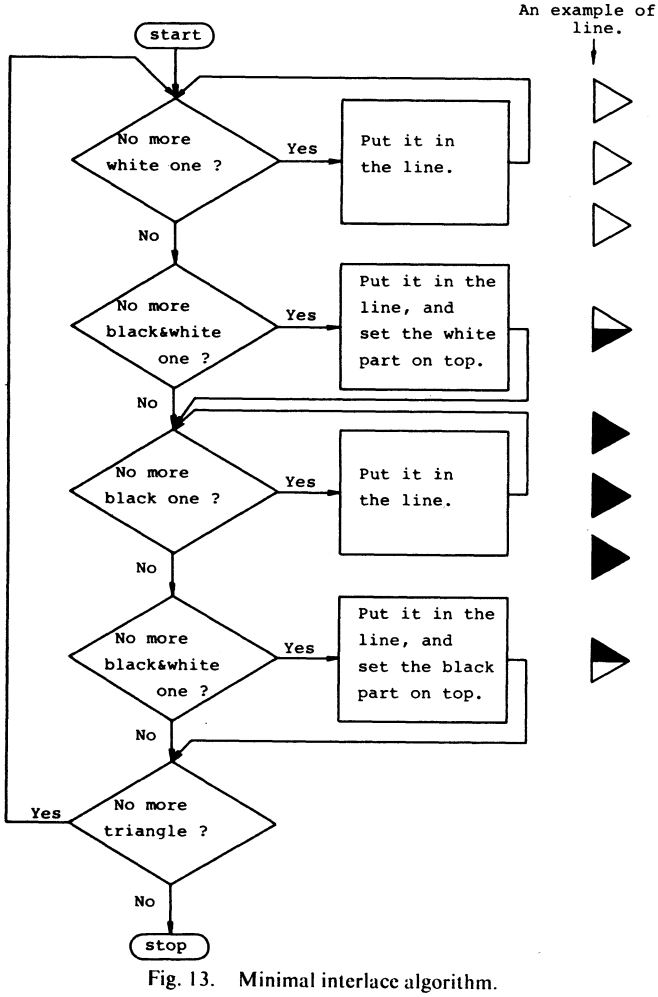


Fig. 13. Minimal interlace algorithm.

APPENDIX

ALGORITHM FOR CALCULATING MINIMAL INTERLACE

The algorithm is outlined in Fig. 13. Fig. 14(b) shows the model for the logic diagram of Fig. 14(a). The black and white triangles correspond to real and pseudoinputs, respectively.

Triangles 1, 2, and $p1$ in subtree $T1$ are rearranged by the algorithm. The result is represented by a single triangle with

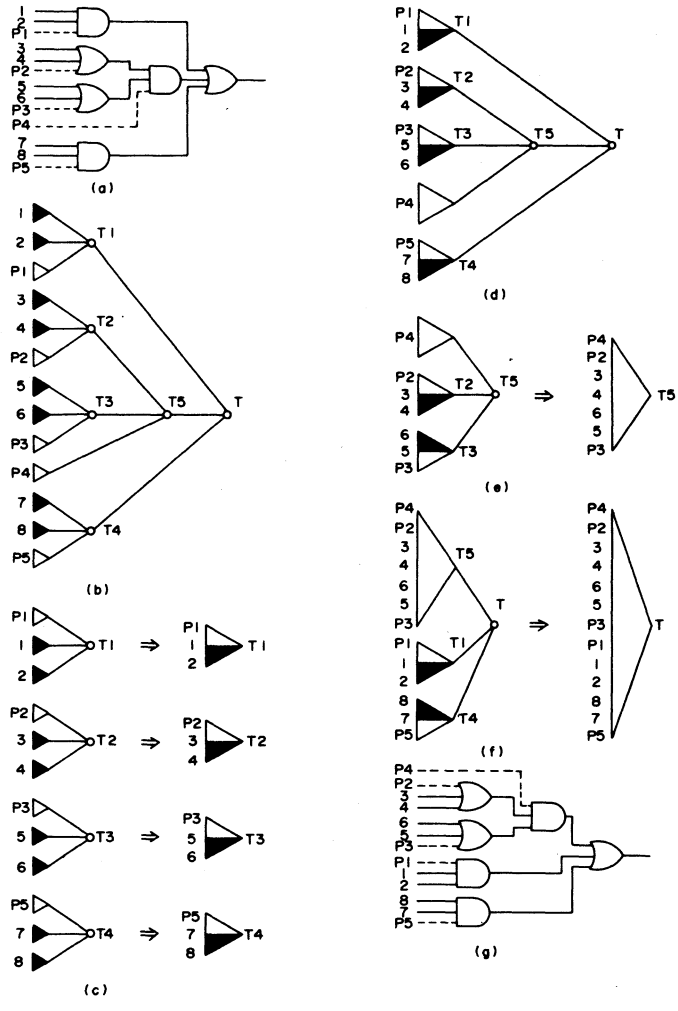


Fig. 14. Example of applying the minimal interlace algorithm.

a white top and a black bottom because the color of the top triangle $p1$ is white and the color of the bottom triangle 2 is black. $T2$, $T3$, and $T4$ are similarly represented by new triangles in Fig. 14(c). The subtrees $T1$, $T2$, $T3$, and $T4$ in Fig. 14(b) are replaced by these new triangles in Fig. 14(c) and a new model is obtained as illustrated in Fig. 14(d). The rearrangement of $T5$ is shown in Fig. 14(e). Note that $T5$ is represented by a white triangle because the color of the top triangle $p4$ is white and so is the color of the bottom of triangle $T3$ in Fig. 14(e). The final rearrangement of the three is shown in Fig. 14(f). The sequence of the inputs can then be obtained by backtracking. The logic diagram in Fig. 14(g) shows one of the sequences with minimal interlace.

ACKNOWLEDGMENT

The authors would like to thank T. Tsuchimoto, J. Tanahashi, and H. Kikuchi of Fujitsu for their encouragement in the early stages of this study. The study of alternative circuit topologies using logical equivalencies was suggested by Dr. K. Miura and Dr. H. C. Lai of Microtechnology. The authors would also like to thank F. Hirosi of Fujitsu for suggesting the algorithm for constructing the minimal interlace.

REFERENCES

[1] W. N. Carr and J. P. Mize, *MOS/LSI Design and Application*, *Texas Instruments Electronics Series*. New York: McGraw-Hill, 1972.

[2] T. Ibaraki and S. Muroga "Synthesis of networks with a minimum number of negative gates," *IEEE Trans. Comput.*, vol. C-20, pp. 49–58, Jan. 1971.

[3] K. Nakamura, N. Tokura, and T. Kasami, "Minimal negative gate networks," *IEEE Trans. Comput.*, vol. C-21, pp. 72–79, Jan. 1972.

[4] H. C. Lai, "A study of current logic design problems, Part 1: Design of diagnosable MOS networks," Ph.D. dissertation, Dep. Comput. Sci., Univ. of Illinois, Urbana, 1976.

[5] B. E. Forbes, "Silicon-on-sapphire technology produces high-speed single-chip processor," *Hewlett-Packard J.*, pp. 2–8, Apr. 1977.

[6] A. Weinberger, "Large scale integration of MOS complex logic: A layout method," *IEEE J. Solid-State Circuits*, vol. SC-2, pp. 182–190, Dec. 1967.

[7] D. G. Schweikert, "Computer-generated IGFET layout using vertically packed Weinberger arrangement," in *Dig. IEEE Int. Solid-State Circuits Conf.*, Philadelphia, PA, 1971, pp. 118–119.

[8] R. P. Larsen, "Computer-aided preliminary layout of customized MOS arrays," *IEEE Trans. Comput.*, vol. C-20, pp. 512–523, June 1971.

[9] D. Gibson and S. Nance, "SLIC—Symbolic layout of integrated circuits," in *Proc. 13th Des. Automat. Conf.*, San Francisco, CA, June 1976, pp. 434–440.

[10] ——, "Symbolic system for circuit layout and checking," in *Proc. IEEE Int. Symp. Circuits and Syst.*, 1977, pp. 436–440.

[11] A. Feller, "Automatic layout of low-cost quick-turnaround random-logic LSI devices," in *Proc. 13th Des. Automat. Conf.*, San Francisco, CA, June 1976, pp. 79–85.

[12] G. Persky, D. N. Deutsch, and D. G. Schweikert, "LTX—A system for the directed automatic design of LSI circuits," in *Proc. 13th Des. Automat. Conf.*, San Francisco, CA, June 1976, pp. 399–407.

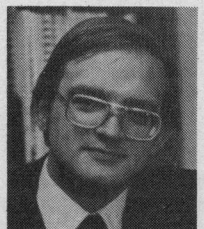
[13] T. Uehara and W. M. vanCleemput, "Optimal layout of CMOS functional arrays," Digital Syst. Lab., Stanford Univ., Stanford, CA, Tech. Rep. 142, 1978.

[14] F. Harary, *Graph Theory*. Reading, MA: Addison-Wesley, 1969.

Takao Uehara (M'70) was born in Tokyo, Japan, on January 27, 1942. He received the B.S., M.S., and Dr.Eng. degrees from Waseda University, Tokyo, Japan, in 1965, 1967, and 1970, respectively.

Since 1970 he has been with Fujitsu Laboratories, Ltd., where he has been engaged in research on CAD systems and computer architecture. From March 1977 through March 1978 he was a Visiting Scholar at the Digital Systems Laboratory, Stanford University. Presently, he is Manager of the Architecture Section, Information Processing Laboratory of Fujitsu Laboratories.

Dr. Uehara is a member of the Institute of Electronics and Communication Engineers of Japan and the Information Processing Society of Japan.



William M. vanCleemput (S'68–M'76) received the M.S. degree in electrical and mechanical engineering from the University of Leuven, Leuven, Belgium, the M.S. degree in mathematics and the Ph.D. degree in computer science from the University of Waterloo, Waterloo, Canada, in 1972 and 1975, respectively.

Currently, he is an Assistant Professor of Electrical Engineering and Computer Science at Stanford University, Stanford, CA. He is also a member of the Computation Research Group at the Stanford Linear Accelerator Center. His research interests include several areas of digital design automation: automated layout of integrated circuits and printed circuit boards, design languages, simulation, and design verification.

Dr. vanCleemput is the Vice-Chairman of the Association for Computing Machinery's Special Interest Group on Design Automation (SIGDA) and of the IEEE Technical Committees on Design Automation and Computer Architecture. He is also a member of IFIP Working Groups 5.2 (CAD) and 10.2 (Design Automation).

On Classes of Positive, Negative, and Imaginary Radix Number Systems

ISRAEL KOREN, MEMBER, IEEE, AND YORAM MALINIAK

***Abstract*—A unified approach to a broad class of finite number representation systems is proposed. This class contains all positive and negative radix systems and other well-known number systems. In addition, it can be extended to include imaginary radix number systems. The proposed approach enables us to develop a single set of algorithms for arithmetic operations.**

***Index Terms*—Arithmetic operations, finite number representation systems, imaginary radix, negative radix, positive radix, radix-complement.**

Manuscript received July 23, 1979; revised May 21, 1980.

I. Koren was with the Department of Electrical Engineering-Systems, University of Southern California, Los Angeles, CA. He is now with the Departments of Electrical Engineering and Computer Science, Technion, Haifa, Israel.

Y. Maliniak is with the Department of Electrical Engineering, University of California, Santa Barbara, CA 93106.

I. INTRODUCTION

THE positive-radix and radix-complement number systems are the most commonly used finite number representation systems and numerous algorithms for arithmetic operations in these systems have been developed and implemented in digital computers. Other fixed-radix systems have received a great amount of attention in recent years [1]–[6], [11], [12]. Various algorithms for arithmetic operations in these systems have been developed and new applications facilitated by their use have been presented, e.g., digital filters [7].

A unified approach to these finite number systems is clearly in order [1]–[3]. Such an approach is proposed here and it is shown that the above mentioned systems are members of a broad class of finite number representation systems. We first

0018-9340/81/0500-0312$00.75