Elektron. Informationsverarb. u. Kybernet. 19 (1983) 10/11, 547 – 557
(Journal of Information Processing and Cybernetics – EIK)

Extending Remarks on the *Eustace-Mukhopadhyay* Approach to Design Rule Checking of Integrated Circuit Layouts

By *R. Karl-Adolf Zech*

Abstract: Recently, *Eustace* and *Mukhopadhyay* [1] presented a surprisingly simple approach to the design rule checking problem of rasterized layout work which is based on finite state automata. The present paper introduces a systolic hardware structure for realizing this approach, thus further enhancing its performance by orders of magnitude.

Introduction

It is well-known that design rule checking (DRC) is one of the bottlenecks in the design and fabrication process of integrated circuits leading to many hours of running time on main frame computers.

Recently, *R. A. Eustace* and *A. Mukhopadhyay* [1] presented an ingenious and amazingly simple approach to this problem for rastered layout data [2] which is based on finite state automata. As the authors state, the simplicity and regularity of their approach is rather well suited for hardware implementation.

Based on this technique, the present paper proposes a systolic [3], [4], [5] hardware structure permitting a high degree of parallelism and making it suitable for VLSI implementation [5], [6], [7]. It can be expected that this extension will reduce the time complexity of the DRC problem by orders of magnitude.

Moreover, this architecture does not require main frame computing power. Instead, only some additional smart memory or data supply unit and an "intelligent" message output terminal is needed.

This approach applies well to simplified λ-design rules for bitmapped layout representations such as the MC-rules [6] for nMOS. On the other hand, the MC-VLSI design methodology [5], [6], [7] allows for the design and fabrication of inexpensive but efficient DRC VLSI hardware of this kind.

1. The *Eustace-Mukhopadhyay* approach

In this section, the technique of *Eustace* and *Mukhopadhyay* (EM-algorithm) is outlined. Let the layout of a design be represented in a rastered manner. That is, the layout is rastered by a grid each element of which is of fixed extent, $\lambda \times \lambda$. Then every picture element, called pixel, is an n-bit Boolean vector. A pixel P of a layout point p has bit i on if (and only if) p belongs to layer i, and off otherwise. Assuming the six layers ion implant, diffusion, polysilicon, contact cut, metal, and glass, in that order, for the simple nMOS process of [6], then, for example, pixel $P = 111000$ describes a point belonging to the polysilicon, diffusion as well as ion implant layers.

Most of the design rules require minimum width and spacing checks within one layer or between several layers [2]. For example, [1] presents a typical spacing check between two diffusion areas. A minimum spacing of 3λ could be checked by an automaton which simply counts the off-bits in the diffusion layer horizontally as well as vertically (Fig. 1).

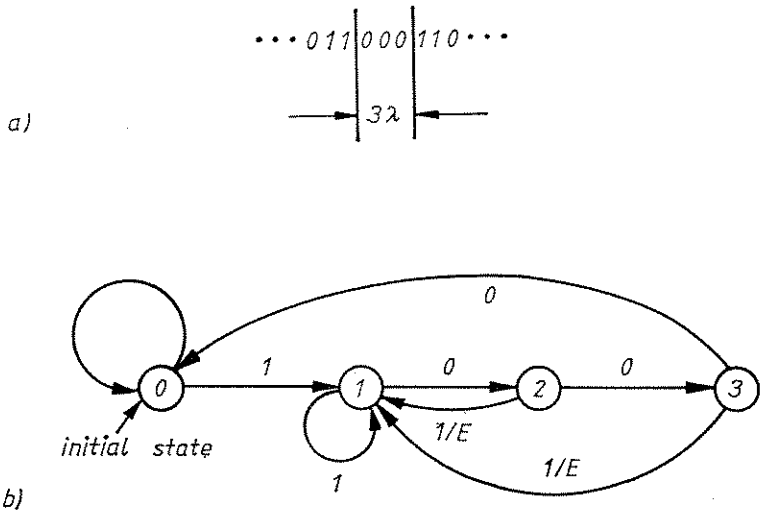

Fig. 1 (from [1]).
a) Minimum spacing of 3λ to be ckecked;
b) automaton for 3λ-spacing check (E: error message)

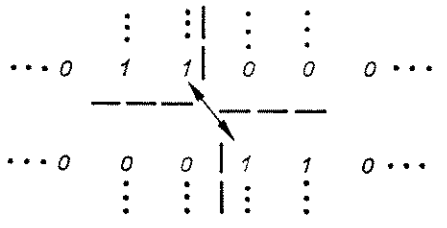

Fig. 2. Example (from [1]) showing that diagonal checks are necessary

This automaton generates output E ("error") if less than 3λ spacing between two diffusion regions are detected. Problems arise when diagonal checks are necessary. For example, the situation sketched in Fig. 2 will not be detected as erroneous of only horizontal and vertical checks are performed.

To solve this problem, the authors expand their finite automata approach by storing information in the states of the automaton which is used for diagonal checks. Let $\boldsymbol{P}$ be the set of possible pixel values, $\boldsymbol{P} = \{0, 1\}^m$, and $A = [X, Y, Z, f, g, z_0]$ be a finite state automaton where:

$Z = \{0, 1, \ldots, n\}$ is the set of states,
$Y = \{\text{blank}, E\}$ is the set of outputs (error message),
$X = \boldsymbol{P} \times Z$ is the set of inputs,
$f: Z \times X \to Z$ the next state function,
$g: Z \times X \to Y$ the output function of A, and
$z_0 = 0 \in Z$ is the initial state of A [8].

The definition of f and g depends on the rule actually to be checked by A.

A operates on the pixel field which is extended by surrounding lines and columns each entry of whose represents the zero (empty) pixel. Now let us think of the pixel values as of the corresponding decimal numbers. A starts in position [1, 0] and moves linewise from left to right, as illustrated in Fig. 3. In each step, A performs the following operations:

(i) read next pixel P on the line
(ii) read value z' below this pixel
(iii) compute $z'' = f(z, [P, z'])$, where z is the present state of A
(iv) write value z in present pixel cell
(v) move one step forward and change present state to z''; if the end of the line is encountered, go to the left end of the next line to start again or stop, if last line.

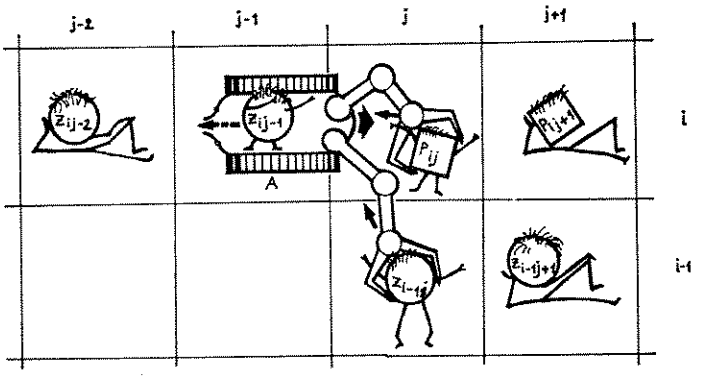

Fig. 3. Illustrating the operation of A interpreted as a mechanical vehicle: A moves on line i from left to right, computing its next state value z_{ij} using its present state z_{ij-1}, the next pixel value P_{ij} and the value z_{i-1j} below P_{ij}, leaving its present state on position $[i, j-1]$. Note that line $i-1$ is either a zero line (for $i = 1$ at the beginning), or has been evaluated within the preceding run of A over line $i-1$.

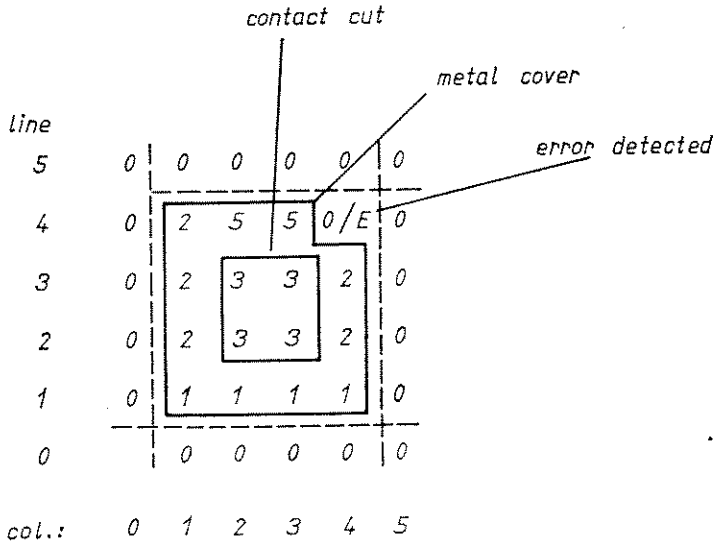

Fig. 4 (from [1]). Metal extension check example.
In line 4, column 4, an error has been detected because left value 5 ("upper metal extension OK") and below value 2 ("right metal extension OK") is requiring metal at that position.

In this way, the picture is filled with values containing all relevant information about the corresponding pixel which merely depends on its left-below environment.

Fig. 4 shows the result of this procedure for the metal extension check of contact cuts, taken from [1]. Note that one could also include simultaneous checks for minimum cut window width.

It should be further noted that no shrinking operations [9] are necessary.

2. A systolic DRC-hardware structure

The EM-approach has been implemented in software [1]. However, as the authors pointed out, the implementation of the functions f and g in hardware is a straightforward task, e.g. using a PLA, thus relieving the host computer of a computing burden which takes nearly one half of the total CPU time.

This section proposes a hardware structure based on the EM-approach described in Section 1, which is parallel in nature. At least in principle, this structure allows to do the DRC job without a host computer.

Consider Fig. 3 again. A reads the values stored in line $i-1$ and stores the new values in line i, which in turn are read by A during the next cycle when A marches along line $i+1$. So nobody can hinder us letting another copy of A marching on line $i+1$, just one step behind our original A operating on line i. The new A can take over every newly computed value of line i directly without any expense for storing it. Continuing this idea, one can send a chain of k automata to run over lines $i, \ldots, i+k-1$, as shown in Fig. 5.

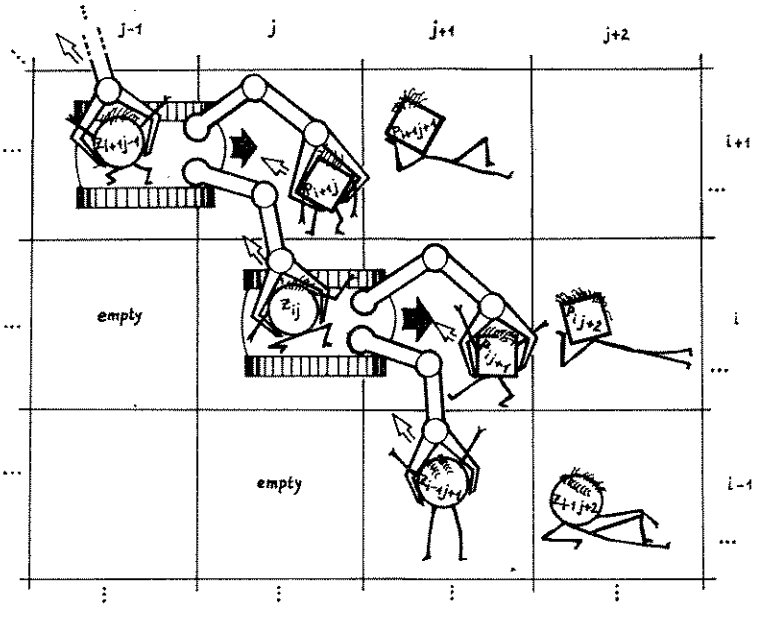

Fig. 5. Parallel operation of more than one automaton which are arranged in a staggered fashion

Fig. 6 shows the corresponding hardware structure. Let the rastered layout data be stored in a memory or let the pixel values be input directly from a rasterization hardware [9], [10].

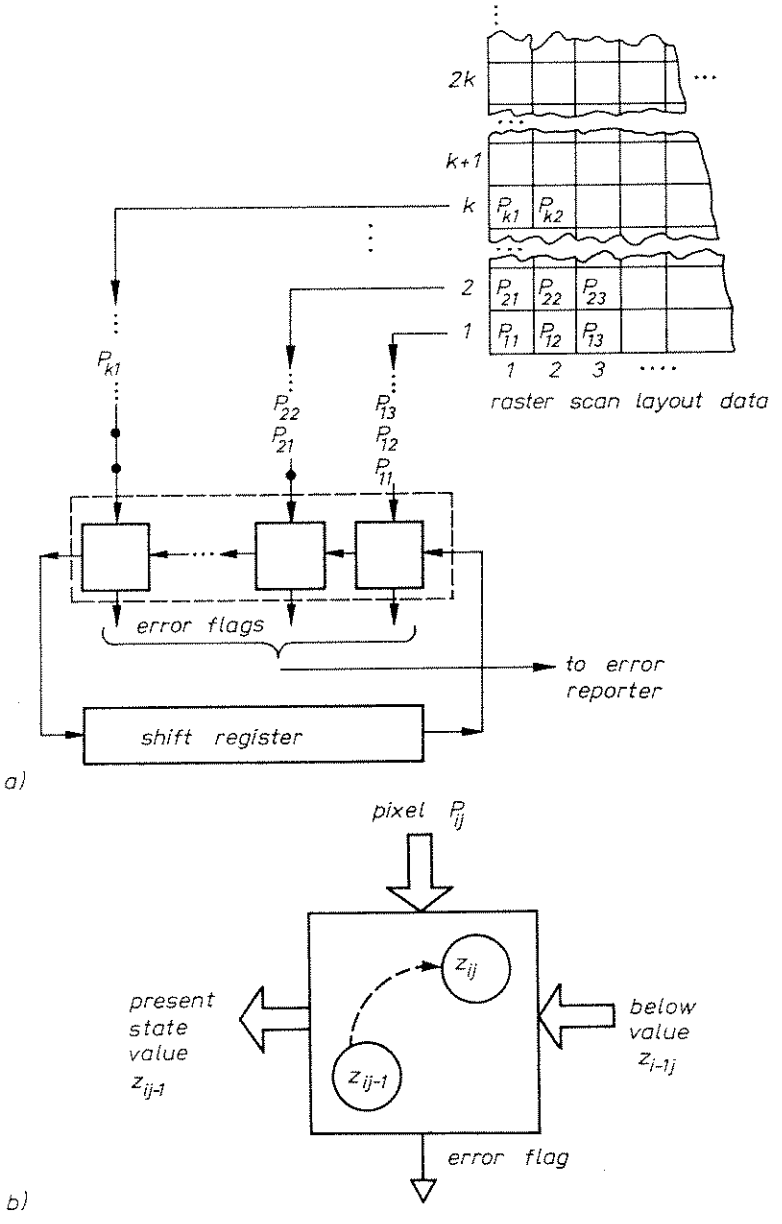

Fig. 6.
a) k identical DRC automata chained as a linear array processing k lines of the rastered layout in parallel (systolically). In its simplest case, the "host computer" may consist of a smart memory for the layout data and a comfortable error message display unit
b) DRC automaton of the array in a).
The preceding automaton supplies the below value z_{i-1j}. Its present state is z_{ij-1} representing the left value. Together with the actual pixel P_{ij} the value z_{ij} can be computed as the next state of the automaton. Meanwhile the left neighbour can utilize the present state value z_{ij-1}

The linear array of k automata $A_1, \ldots, A_k$ is combined with a shift memory the length of which equals the length of the raster lines to be processed plus one. Before starting, this memory is cleared such that during the first cycle A_1 is receiving zeros

only which in fact reflects the situation of A marching on line 1 and reading zeros from line 0. Moreover, all automata are initially cleared such that their inner states are equal to the zero values of column 0. In step 1 of cycle 1, pixel P_{11} enters A_1 which in turn computes its next state value. In step 2, this value is passed to A_2 which is receiving the first pixel of line 2, P_{21}, A_2 computes its next state value, and so on. Note that the input of every line is delayed by one step with respect to its predecessor line such that the correct below value is available at the correct time.

The last automaton, A_k, passes its state value to the shifter. In cycle 2, lines $k + 1, \ldots, 2k$ are being processed. Now, the shifter passes the values stored in cycle 1 back to the array, to automaton A_1, which utilizes them as the below values for processing line $k + 1$.

Thus, during the processing of the rastered layout partitioned in line groups $1, \ldots, k, k + 1, \ldots, 2k, \ldots, (n - 1) \cdot k + 1, \ldots, n \cdot k$, in contrast to the EM-method, only every k-th line of values has to be stored.

The linear array displays error bits which can be used by the host computer or by an "intelligent" error reporter to produce a human readable error report.

To check for all design rules, for every rule one specific type of DRC automaton must be generated and chained together in the way just described. To avoid multiple inputs of the layout lines, the two-dimensional array of Fig. 7 can be used. The pixels once read in are passed to the neighbour automaton checking for another rule (see Fig. 8). So we have a two-dimensional systolic array [3], [4] with two orthogonal data streams: the pixels and the state values.

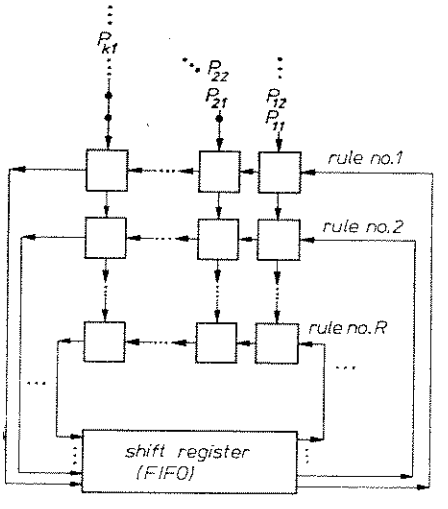

Fig. 7. Two-dimensional array with additionally transfering the pixel data orthogonally to the flow of the state values

For handling the error messages, consider Fig. 9 showing an automaton checking for rule r. In addition to the next pixel value, an error information is read in. This is the greatest number of all previously checked rules where an error for the actual pixel has been encountered. If hitherto no error has occured, this number is zero. If the checking for rule r leads to an error, the binary coded form of r is passed, and the input error value otherwise. Note that the error values and the corresponding pixels are flowing in parallel such that at the end of the array the following information is

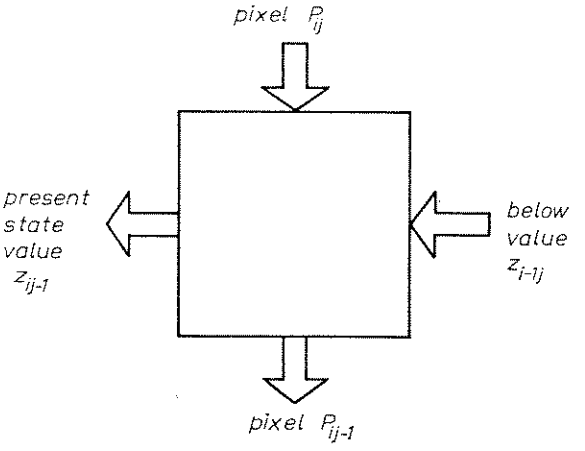

Fig. 8. I/0 of a DRC automaton of the two-dimensional array; the received pixels are passed one time step later

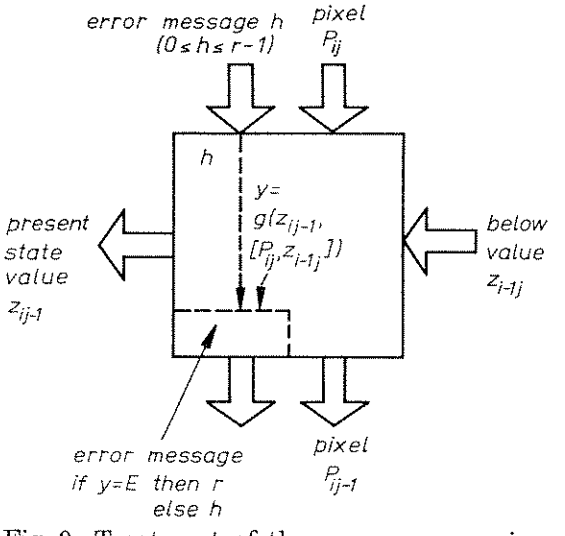

Fig. 9. Treatment of the error messages in a DRC automaton for rule no. r

available:

for pixel P in relative line u and column v { no error encountered or error encountered at least for rule r

In [1] the tables of the DRC automata are constructed in such a way that the number of error messages is minimized. For example, if some line does not pass the minimum width rule, only one single message is produced.

To further reduce the error information to be processed by the error reporter, an additional linear array at the below border of the array according to Fig. 10 can be applied. Thus, at the end of the array the following error information can be derived:

in column v {lines $(t-1)\,k+1, \ldots, tk$: no errors encountered or at least in line $(t-1)\,k+u$ and at least for rule r an error has been detected (i.e.: errors for lines $(t-1)\,k+1, \ldots, (t-1)\,k+u-1$ and for rules $1, \ldots, t-1$ are also possible but not reported).

where $t = 1, 2, \ldots$ is the cycle number.

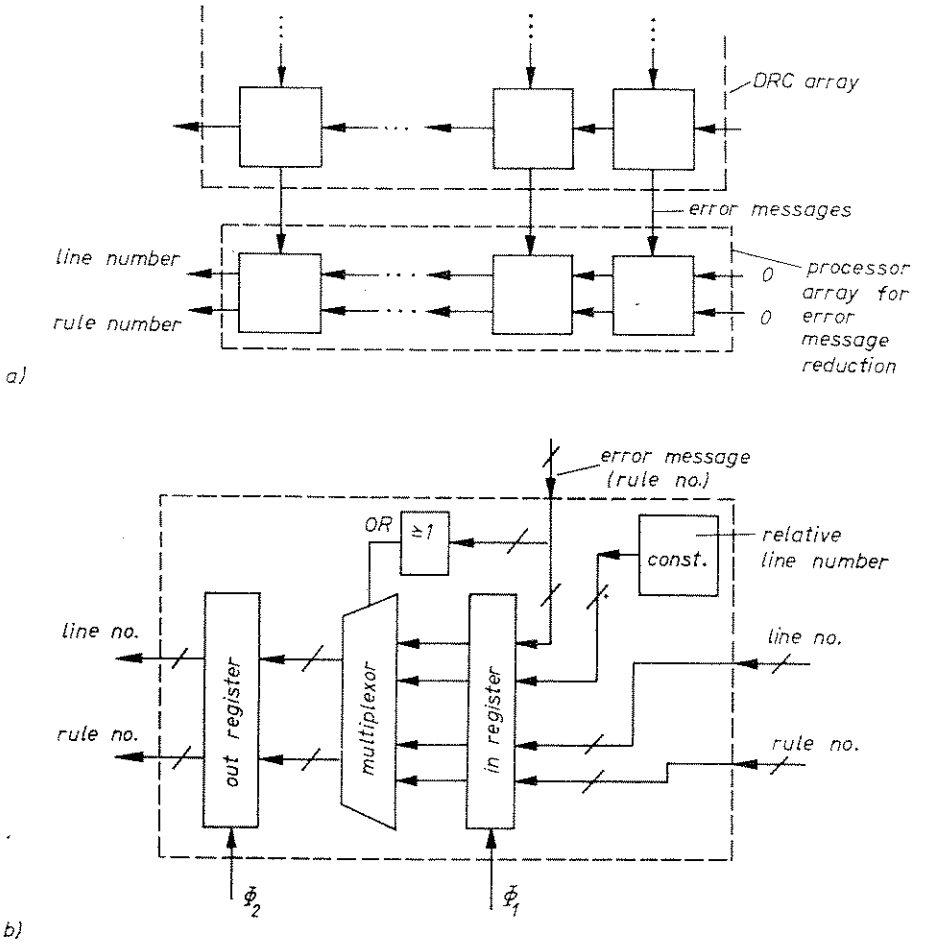

Fig. 10. Reduction of the error information by using an additional processor array

a) overall structure; the left end error processor outputs the greatest relative line number where an error has been detected as well as the greatest number of all violated rules; the column number is externally determined by counting;

b) basic structure of an error processor cell;
Φ_1 and Φ_2 are the in and out clock signals, respectively

If the accuracy of the line information is not sufficient, this array may be broken to reduce the loss of information, but in general this simple message should be precise enough to tell the designer where to correct his mistake.

The structure described so far suffers from the necessity of the large shift register memory limiting the number of columns of the layout. The next section will show how to circumvent this drawback.

3. Avoiding the shift register

To avoid the large shift register one must assure that every DRC automaton either receives valid below values (which otherwise are supplied or guaranteed by the register) or to suppress wrong error messages due to lack of correct below values while finally assigning every layout pixel its correct value or error flag, respectively.

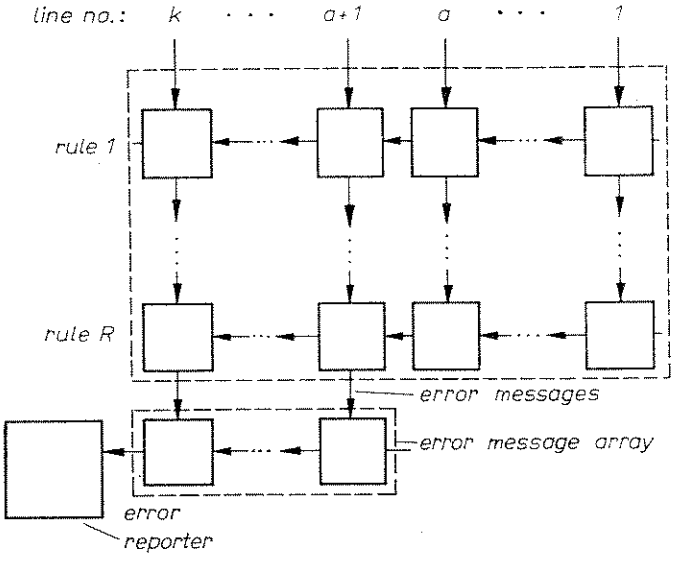

Fig. 11. Hardware structure for making unnecessary the shift memory by using an overlapping partition of pixel lines: the first a lines of every group are identical to the last a lines of the preceding group

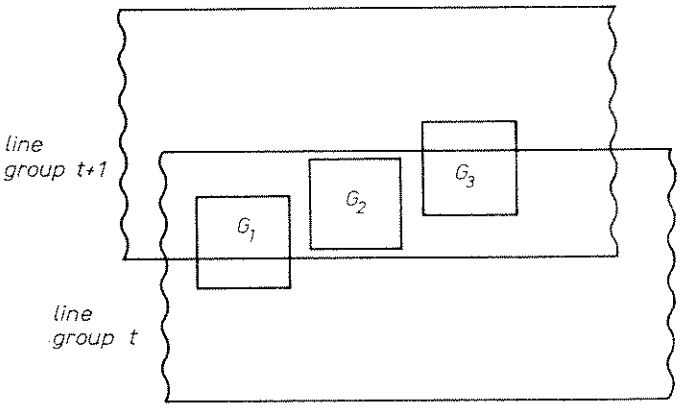

Fig. 12. Overlapping line groups and three positions of maximal extent design rule areas

G_1: Area lies completely within line group t and is therefore fully checked during cycle t; during the next cycle $t + 1$, G_1 is not fully contained in group $t + 1$ such that the DRC array will produce wrong error states which however are not displayed.

G_2: This area is completely contained in both line groups. Errors are displayed either in cycle t, or in cycle $t + 1$ if discovered in line $a + 1$ of group $t + 1$.

G_3: This area is contained completely in group $t + 1$ but only fragmentarry in group t. Errors occuring outside group t are tracked and displayed in cycle $t + 1$ because all conditions are newly computed; errors already displayed in cycle t are not marked once again.

Let a be the extent of the largest design rule. Then let the first a lines of a line group be identical to the last a lines of the previous group, where the first line group has zero lines as its first a lines.

Then the hardware structure of Fig. 11 does the job. Note that it covers all possible cases of overlapping figures, as shown in Fig. 12.

It should be emphasized that the length of the pixel lines and therefore the size of the layout is restricted now by the pixel supply equipment only.

4. Performance evaluation

The clock rate is limited by the maximal time for computing the functions f and g in the different automata. Realizing f and g using PLA's and assuming automata with, say, at most 32 states, a technology to be checked for with 6 mask levels [6] and no more than 16 complex design rules, there are 16 proper inputs and 7 outputs such that 300 ns per processing step can be a realistic expectation. A typical chip of 7.6×7.6 mm^2 will result in rastered layout data of 3050×3050 pixels when $\lambda = 2.5$ μm [11].

Let $a = 6$ and $k = 20$, then every cycle checks 14 lines in parallel taking 300 ns $\times$ (3050 + 20) $\approx$ 0.92 ms, or 0.22 seconds for the complete design if the pixels can be supplied sufficiently fast. If $\lambda = 1$ μm, then it takes 1.5 seconds, in contrast to 87 seconds when using the special hardware proposed by *Seiler* [9]. Even if a 50% time overhead for additional delays are taken into account, this shows that the performance level — in principle — is improved by one or two orders of magnitude. Besides that, the EM-approach used here is more flexible than the Seiler-approach.

5. Conclusion

Based on the finite automaton approach to design rule checking for VLSI in [1] the paper presents a systolic high performance structure which is suitable for checking complete VLSI designs. Such hardware support circuitry will become more attractive as simplified design rules [6] are being used.

Acknowledgements

A dept of gratitude is owed to my family and my wife, *Liebgard*, for their patience and moral support while I was cutting our common rare spare time. Appreciation also goes to Dipl.-Ing. *Werner Klarkowski*, Computer Center, Humboldt-Universität Berlin, for assisting the software modeling of the method described herein. Especially, I whish to mention Dipl.-Math. *Dieter Leistner*, formerly chief programmer, director of the computer center, Institut für Nachrichtentechnik Berlin, for having been a constant source of motivation during the last years. His considerate guidance provided a unique working atmosphere and greatly influenced the productivity of my work.

References

[1] *Eustace, R. A., A. Mukhopadhyay,* A Deterministic Finite Automaton Approach to Design Rule Checking for VLSI. In: Proc. 19th Design Automation Conference, Las Vegas (Nevada); IEEE Computer Society, Los Angeles 1982; pp. 712—717.

[2] *Baker, C. M.*, Artwork Analysis Tools for VLSI Circuits. MS Thesis, MIT Cambridge (Mass.), 1980.
[3] *Kung, H. T., Ch. E. Leiserson*, Systolic Arrays (for VLSI). In: Sparse Matrix Proceedings 1978; Eds.: *I. S. Duff, G. W. Stewart*; Society for Industrial and Applied Mathematics, 1979; pp. 256—282.
[4] *Foster, M. J., H. T. Kung*, The Design of Special-Purpose VLSI Chips. Computer Magazine **13** (1980) Jan., 26—40.
[5] *Hartenstein, R. W.*, VLSI-Bausteine in geringen Stückzahlen für Spezial-Anwendungen. Elektron. Rechenanl. **22** (1980) 4, 159—173.
[6] *Mead, C., L. Conway*, Introduction to VLSI Systems. Addison Wesley, Reading (Mass.) 1980.
[7] *Heinz, G.*, Grundzüge des höchstintegrierten Schaltkreisentwurfs (VLSI). INT-Mitteilungen (Berlin), Ausgabe A, 2/82.
[8] *Starke, P. H.*, Abstrakte Automaten. Deutscher Verlag der Wissenschaften, Berlin 1969.
English Translation: Abstract Automata. North Holland, Amsterdam 1972.
[9] *Seiler, L.*, A Hardware Assisted Design Rule Check Architecture. In: Proc. 19th Design Automation Conference, Las Vegas (Nevada). IEEE Computer Society, Los Angeles 1982; pp. 232—238.
[10] *Locanthi, B.*, Object Oriented Raster Displays. In: Proc. Caltech Conference on Very Large Scale Integration 1979; California Institute of Technology, Pasadena (Ca) 1979; pp. 215—225.
[11] *Hon, R. W., C. H. Sequin*, A Guide to LSI Implementation. 2nd Ed., XEROX PARC, Palo Alto (Ca) 1980.

Kurzfassung

Kürzlich veröffentlichten *Eustace* und *Mukhopadhyay* eine überraschend einfache Möglichkeit zur Behandlung des Problems der geometrischen Entwurfsregelprüfung gerasterter Layoutdaten, die auf endlichen Automaten basiert. Die vorliegende Arbeit schlägt eine systolische Hardwareimplementierung für diesen Ansatz vor. Auf diese Weise wird das Verfahren um Größenordnungen beschleunigt.

Резюме

Юстас и *Мукопадей* недавно опубликовали неожиданно простой метод испытаний геометрических параметров лейаута, заданных в растре. Метод базируется на использовании конечных автоматов. Данная работа предлагает систолическую реализацию прибора на этой основе; таким образом, испытание ускоряется на несколько порядков.

(Received: First version November 11, 1982,
present version March 2, 1983)

Author's address:

Dr. R. K.-A. Zech
1058 Berlin
Schliemannstr. 28
German Democratic Republic