CELLULAR IMAGE PROCESSING TECHNIQUES FOR CHECKING VLSI CIRCUIT LAYOUTS[1]

T.N.Mudge, W.B.Teel.
Electrical and Computer Engineering Department
University of Michigan
Ann Arbor, MI 48109
and
R. M. Lougheed[2]
Environmental Research Institute of Michigan
Ann Arbor, MI 48107

Abstract

Simple cellular image processing techniques are described that perform design rule checks on VLSI circuit layouts. The architecture of the Cytocomputer is described; a pipelined image processing computer that can implement cellular, or neighborhood, transformations on image data very efficiently. Design rules for IC layouts are presented. A set of cellular image processing transformations are defined that perform the design rule checks. These can be used to program the Cytocomputer. The results comparing such a program run on the Cytocomputer against a similar design rule checker run on a VAX 11/780 are given.

1. Introduction

It is projected that by 1985 integration levels for VLSI circuits will approach one million devices [1]. Even more complex systems are anticipated from the VHSIC (Very High Speed Integrated Circuits) program [2]. Systems of such complexity require computer-assisted validation at many stages in the design procedure before being committed to silicon. This paper describes how an important class of design rule checks can be performed on bit-map representations of the IC masks using the Cytocomputer [3,4,5], a special purpose computer developed for high-speed image processing by the Environmental Research Institute of Michigan. The design rule checks are those proposed by Mead and Conway in [6], and are appropriate for nMOS technology in which the features conform to a rectangular grid. The paper shows that efficient processing of such checks can be done by a computer whose architecture is geared to cellular image processing. In particular, the paper shows that these checks can be performed using neighborhood transformations that are no more than simple Boolean operations on the bit-map representation of the IC masks.

The Cytocomputer is a special purpose high-speed image processor based upon cellular automata concepts. It consists of a pipeline of identical processing stages capable of performing neighborhood transforms. It is controlled by an interpreter that allows the machine to be used interactively. The paper outlines the machine's architecture and how it implements the above neighborhood transformations.

A course in VLSI circuit design and fabrication currently being run by the ECE Department at the University of Michigan in cooperation with General Motors Research Labs provided a source of masks for use as benchmarks. The artwork for three ICs, three sets of masks, was checked on the Cytocomputer and then by the checker program used in the course. This last program runs on a VAX-11/780, implements the set of checks proposed in [6], and, like the Cytocomputer, works from a bit-map representation of the mask. It is described in [7,8]. Results presented in Section 5 show that the Cytocomputer can perform the design rule checks of [6] on multi-mask layouts two orders of magnitude faster than the checker running on the VAX-11/780.

The paper is organized as follows: the next section describes the design rules of [6]; Section 3 describes the Cytocomputer architecture; Section 4 describes the neighborhood transformations to be used with the Cytocomputer, as well as illustrating their use as design rule checkers; Section 5 presents the results of comparing the Cytocomputer design rule checker against the VAX 11/780 design rule checker, and Section 6 is the conclusion.

2. Design Rules

The design rules are a set of geometrical constraints that the masks of the wafer fabrication process must satisfy. The process of interest in this paper is that described in [6] for nMOS. The constraints are expressed in a

[1]This work was supported in part by an unrestricted grant from the Fairchild Corporation, NSF grant MCS-8009315, and the Environmental Research Institute of Michigan.

[2]R.M.Lougheed is also an instructor with the University of Michigan.

dimensionless form as multiples of a basic length-unit λ. In 1981 λ is in the range 1.5-2 microns for typical commercial processes.

The design rules are shown in Figure 1 (figures and tables are at the end of the paper). The masks involved are those for diffusion, contact cuts, ion implantation, metal, and polysilicon. The design rules partition into three classes: single mask width and spacing constraints, transistor geometry constraints, and contact cut constraints.

The width and spacing constraints are shown in Figure 1(a). The width of a diffusion region should be no less than 2λ, and the spacing between two diffusion regions should be no less than 3λ. The minimum width and spacing for polysilicon and contact cuts should be 2λ, and the minimum width and spacing for metal should be 3λ.

The transistor geometry constraints, are shown in Figure 1(b). D indicates diffusion, I indicates ion implantation, P indicates polysilicon, and T the transistor formed from the polysilicon region crossing the diffusion region. Strictly speaking, since polysilicon is laid down before the diffusion process occurs there is essentially no diffusion under the T region. This region forms a channel in the substrate between the separated regions of diffusion and conduction in this channel is controlled by the polysilicon gate. It can be seen from Figure 1(b) that the minimum distance between a polysilicon region and a diffusion region should be λ unless they cross to form a transistor. In this latter case both the polysilicon region and the diffusion region should extend at least 2λ beyond the T region (see Figure 1(b) left). If the transistor is a depletion mode transistor the ion implantation region required for such a device should extend beyond the T region 1.5λ in all directions. Furthermore, the I region should be separated by at least 1.5λ from any enhancement mode transistors (see Figure 1(b) right).

Finally, there are the contact cut constraints. These are shown in Figure 1(c). C indicates a contact cut. There are two types: normal contacts and butting contacts. The constraints for the normal contact are shown on the left of Figure 1(c) and those for the butting contact are shown on the right. The normal contact joins metal to either polysilicon or diffusion. The butting contact joins polysilicon and diffusion with a metal "jumper". The width and spacing requirement for normal contacts were outlined above (see part(a) of Figure 1). In addition, the two regions to be joined should overlap the contact cut by at least λ. In the case of butting contacts the dimension x should be at least 4λ, and the union of polysilicon and the diffusion regions should overlap the contact cut by at least λ. These two regions should overlap one another by λ under the contact, and the metal "jumper" or cap that joins them should also overlap the contact by at least λ.

In order to check these constraints with the Cytocomputer some simple neighborhood transformations were developed that could be readily programmed on the Cytocomputer to operate on bit-map images of the mask layouts. These are discussed in the Section 4.

3. Cytocomputer Architecture

The most obvious architecture for implementing neighborhood transformations is a two dimensional parallel array of processors. In such an array one processor exists for each pixel in the image. These processors operate in lock-step and communicate with their nearest neighbors to perform neighborhood operations. Although such architectures can be very fast, they have drawbacks in light of the limitations of present fabrication and packaging technology.

The practical shortcomings of parallel array image processors led ERIM to develop an alternative parallel structure, the Cytocomputer. In [4] Lougheed and McCubbrey compare the Cytocomputer and array cellular processor architectures and show that the Cytocomputer architecture can have several advantages over the array architecture, such as low complexity, high bandwidth, and considerable architectural flexibility. A Cytocomputer (see Figure 2) consists of a serial pipeline of neighborhood processing stages, with a common clock, in which each stage in the pipeline performs a single neighborhood transformation of an entire image. Images are entered into the pipeline as a stream of eight-bit pixels in sequential line-scanned format and progress through the pipeline of processing stages at a constant rate. Alternatively, the data stream can be viewed as eight independent one-bit images. Following the initial latency to fill the pipeline, processed images are produced at the same rate they are entered. Shift registers within each stage store two contiguous scan lines while window registers hold the nine neighborhood pixels which constitute the 3x3 window input of a neighborhood logic module. This module performs a preprogrammed transformation of the center pixel based on the values of the center and its eight neighbors. Neighborhood logic transformations are computed within the data transfer clock period, allowing the output of a stage to appear at the same rate as its input. At each discrete time interval a new pixel is clocked into the stage. Simultaneously, the contents of all delay units are shifted one element. In addition, operations which do not involve the states of a pixel's neighbors, such as bit anding, bit setting, etc., are performed in a separate point-by-point logic section to simplify the

neighborhood logic circuit.

To visualize the transformation process, imagine a 3x3 window moving across an image array as shown in Figure 3. The processing stage storage section shows the contents of the latches after pixel A[6,6] has been read. The contents of the neighborhood latches allow the stage to compute the transformed value of the pixel in the center latch, A[5,5]. This transformed pixel becomes an element of the serial input to the next stage. From this example, one can see that the latency of a stage is equal to N + 2 time steps, N being the linelength. One can now visualize a series of 3x3 stage windows following each other across the image, each one processing the previous stage's output as shown in Figure 4.

The concept of a serial processing stage has previously been described in [9]. The important features of a Cytocomputer involve the "chaining" of a large number of physically identical, software programmable modules to achieve high speed (real-time) image processing. This gives very low cost solutions to the majority of image processing problems, in which the original source and final destination of the images are serial in nature, i.e., vidicon cameras, disk storage, etc.

Cytocomputer architecture lends itself well to LSI implementation for several reasons. First and foremost, the limited number of I/O connections required between stages fits well within the pin constraints of single packages. As IC technology advances, more stages may be combined in a single package with no increase in pin requirements. In addition, the large number of identical parts per system provides the volume necessary for economical manufacture. Finally, the flexible, programmable nature of the devices means diverse areas of application are possible, leading to a wide market for such a part.

Following the successful completion of the first generation system, a design effort was carried out to produce a processing stage suitable for implementation in a custom LSI circuit. This design is currently being fabricated by a major IC vendor with production scheduled for late 1981. It incorporates all functions of the previous generation hardware. The instruction set of primitive transformations executable by the processing stages was chosen through careful studies of image processing algorithms implemented on a software simulation.

4. Neighborhood Transformations

In the transformations defined below it is convenient to think of the Cytocomputer as operating on eight separate binary images in parallel, rather than one image of eight-bit pixels. Recall from Section 3 that the Cytocomputer can operate in either of these modes. The images of the five masks of interest--contact cut, diffusion, implant, metal, polysilicon--can then be processed on one pass through the machine.

Denote the binary images (i.e., bit-maps) of the contact cut, diffusion, implant, metal, and polysilicon masks by C, D, I, M, and P respectively. Binary pixels from each of these mask images occupy five of the eight bits in the Cytocomputer's data path. The remaining three bits can be used as temporary variables for the intermediate results that occur when the neighborhood logic applies a transformation to an image, or when the point-to-point logic performs a bit-wise logical operation between images. Denote the temporary bit-maps as T1, T2, and T3.

Consider a 3x3 neighborhood that is represented by a set of binary-valued variables <n, ne, e, se, s, sw, w, nw, center>; the cells are labelled with the points of the compass for ease of reference. The result of a neighborhood transformation, f, is a binary value given by f(neighbor set). Define the following functions:

f_1 = n·ne...center (Boolean "and" of all the neighbors.)

f_2 = n·center

f_3 = e·center

f_4 = s·center

f_5 = w·center

In addition, define $g_1, g_2, \ldots, g_5$, to be the duals of $f_1, f_2, \ldots, f_5$, respectively (replace the "ands" with "ors"). Using these neighborhood transforms define the following functions that transform complete bit-map images into new bit-map images.

erode [A]: apply f_1 to every pixel in image A.

erode_north [A]: apply f_2 to every pixel in image A.

erode_east [A]: apply f_3 to every pixel in image A.

Similarly for erode_south and erode_west. The term erode, as the name suggests, can be thought of as a transformation that shrinks regions of 1s in the bit-map by one pixel. The function "erode" shrinks one pixel in all directions, the function "erode_north" shrinks one pixel in the northerly direction only, the function "erode_east" shrinks one pixel in the

easterly direction, etc. A dual set of functions can be defined that depend on g_1, g_2,..., and g_5. These transform complete bit-map images into new bit-map images as follows.

dilate [A]: apply g_1 to every pixel in image A.

dilate_north [A]: apply g_2 to every pixel in image A.

dilate_east [A]: apply g_3 to every pixel in image A.

Similarly for dilate_south and dilate_west. The term dilate, as the name suggests, can be thought of as a transformation that expands regions of 1s in the bit-map by one pixel. The function "dilate" expands one pixel in all directions, the function "dilate-north" expands one pixel in the northerly direction only, the function "dilate_east" expands one pixel in the easterly direction, etc.

The notions of erosion and dilation are derived from the more general transforms on sets of points of Minkowski addition and subtraction described in [10].

The design rules of Section 2 can be checked by successive application of these transformations together with bit-wise Boolean operations. To illustrate this the width check of 2λ for polysilicon regions and the 2λ overlap check for polysilicon in the neighborhood of a transistor will be covered in more detail.

The following program checks the P image for regions of 1's (presence of polysilicon) that are less than 2λ in width. Any such regions are flagged by placing 1's in the corresponding pixel positions of the T1 image. The masks are assumed to be drawn on a 1/2λ spaced grid. Therefore, each image pixel represents a 1/2λ square region of a mask. Thus the width check corresponds to detecting regions that cannot contain a horizontal or vertical line of four 1s.

Width Check

(1) T1 := erode_east [P]

(2) T1 := erode_south [T1]

(3) T1 := erode [T1]

(4) T1 := dilate [T1]

(5) T1 := dilate_south [T1]

(6) T1 := dilate_east [T1]

(7) T1 := exor [T1,P]

The above seven steps are diagrammed in Figure 5. Notice that all the steps except the last one are neighborhood transformations. The seventh step is a bit-wise exclusive-or between the original image and a image with all regions failing the width test removed (set to 0s). The strategy of the program is as follows: The first two steps produce a image that is shrunk by one pixel along any of its easterly or southerly borders. The next step (3) shrinks the image by one pixel in all directions. Thus any regions of 1's which are less than or equal to three pixels in width will be removed. Steps 4 through 6 replace the eroded pixels everywhere except where removal has occurred. Step 7 flags those places. T1 now contains an image of the errors. Four stages of the Cytocomputer are required for this program.

The following program checks for a polysilicon overlap of at least 2λ in the neighborhood of a transistor.

Overlap Check

(1) T2 := and [P,D]

(2) T3 := exor [P,T2]

(3) T3 := flags from a width test of 2λ on T3 (see previous program)

(4) T1 := or [T1,T3]

Step 1 identifies the transistor gate area by doing a bit-wide "and" between the polysilicon and diffusion mask images. The gate area is then removed from the P image in Step 2, and the resulting image is placed in T3. A width check of 2λ is performed on T3, and any errors are accumulated in T1. T1 serves as the "error image".

Similar techniques to those presented in the above two example programs allow the checking of all of the design rules outlined in Section 3. (For example, spacing checks can be performed as width checks on the bit-wise complement of a mask image). Each step in each program may require a separate stage in the Cytocomputer. The design rules of [6] require approximately 45 stages.

5. Experimental Results

The artwork for three ICs was checked first using the VAX 11/780 based checker, and then using an 88 stage TTL implementation of the Cytocomputer.

The results are shown in Table 1. The three ICs were a quasi-serial inner product generator, a programmable waveform generator, and an eight bit ALU slice. For these three ICs λ=4 microns so they are all approximately 4 mm square. Furthermore, it can be seen from the table that they each contain about 500 active devices. The times shown are CPU seconds. The Cytocomputer is seen to be nearly two orders of magnitude faster. In fact, this

is a conservative figure since the Cytocomputer worked on 1/2λ square pixels whereas the VAX 11/780 worked on 1λ square pixels. The Cytocomputer thus processed four times as much data. Finally, it should be noted that the Cytocomputer times were adjusted for a CMOS single-chip realization with a pixel processing rate of 500 KHz. As noted, the experiment was performed on a TTL version that has a pixel processing rate of 1.6 MHZ.

6. Conclusion

A significant speedup in the time taken to check design rules for VLSI layouts was demonstrated by applying some simple cellular image processing techniques. Much of this speedup was due to the Cytocomputer, a special purpose computer oriented towards cellular image processing. The anticipated increase in complexity of IC designs together with the anticipated increase in demand for ICs would seem to justify such a machine as a component in a VLSI CAD system, particularly since it would represent only an incremental cost to the overall system. Our future research will study the implementation of more complex design rules; for example features with 45 degree angles can be included if masks are represented with three bits/pixel. In addition to more complex design rules our future research will also study how the techniques presented in this paper can fit into a interactive design environment.

References

[1] D. A. Patterson, C. H. Sequin, "Design Considerations for Single-Chip Computers of the Future," IEEE Trans. Computers, Vol. C-29, No. 2, Feb. 1980.

[2] D. F. Barbe, "VHSIC Systems and Technology," Computer Vol. 14, No. 2, Feb. 1981.

[3] R. M. Lougheed, et al., "Cytocomputers: Architectures for Parallel Image Processing," Proc. IEEE Workshop on Picture Data Description and Management, Aug. 1980.

[4] R. M. Lougheed, D. L. McCubbrey, "The Cytocomputer: A Practical Pipelined Image Processor," Proc. 7-th Annual International Symposium on Computer Architecture, May 1980.

[4] S. R. Sternberg, "Cellular Computers and Biomedical Image Processing," Proc. U.S. - France Seminar on Biomedical Image Processing, May 1980.

[6] C. Mead, L. Conway, Introduction to VLSI Systems Addison-Wesley, Reading, 1980.

[7] C. M. Baker, Artwork Analysis Tools for VLSI Circuits, (M.S. Thesis) MIT, 1980.

[8] C. M. Baker, C. Terman, "Tools for Verifying Integrated Circuit Designs," Lambda, 4-th Quarter, 1980.

[9] B. Kruse, "A Parallel Picture Processing Machine," IEEE Trans. Computers, Vol. C-22, 1973.

[10] G. Matheron, Random Sets and Integral Geometry, Wiley, New York, 1975.

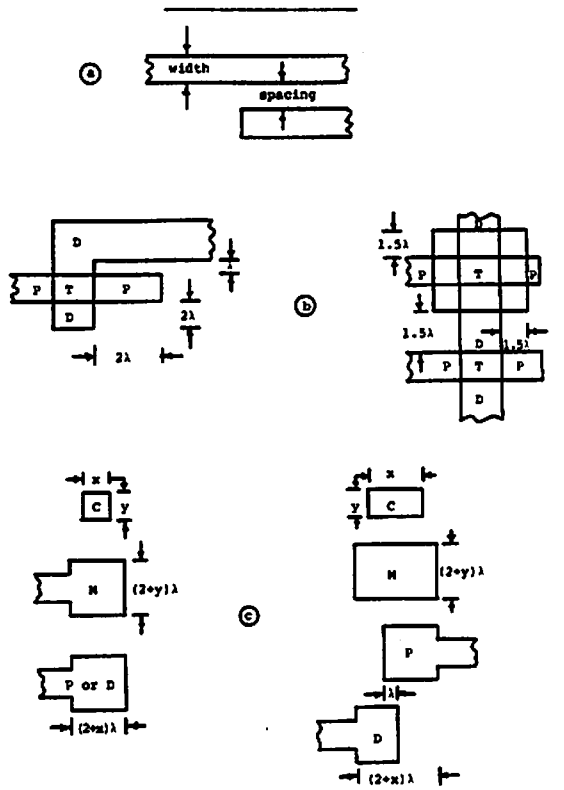

Figure 1. Integrated Circuit Design Rules

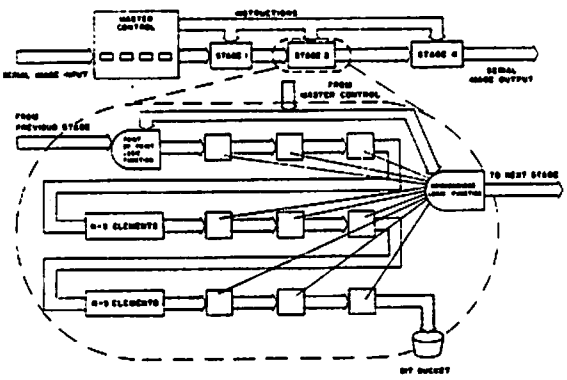

Figure 2. Block Diagram of the Cytocomputer

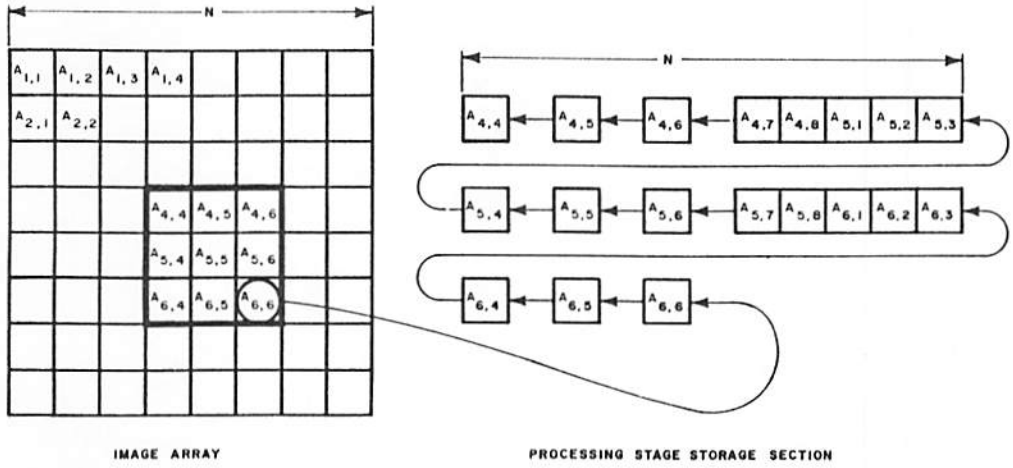

Figure 3. Moving Window Implemented with Shift-register Storage

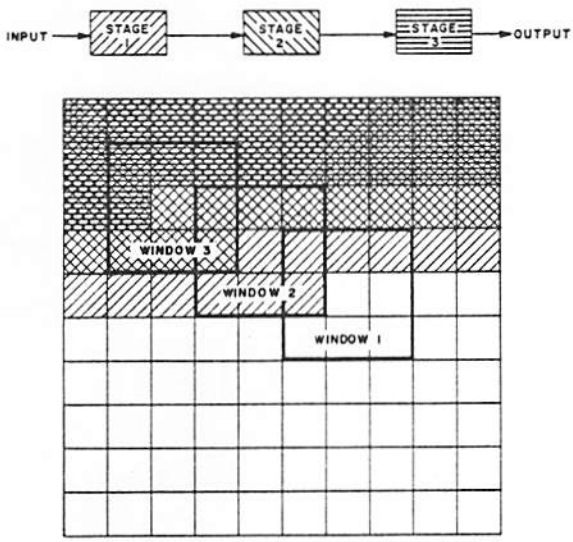

Figure 4. Three-step Image Transformation Showing Window Relationships During Execution

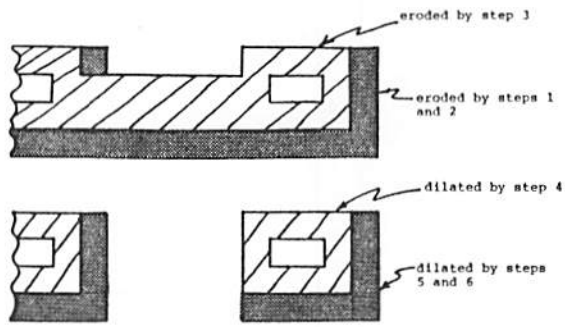

Figure 5. Illustration of Width Check

Experimental Results

Integrated Circuit Function	Size in λ^2	Device Count (approx.)	Time in Seconds	
			VAX 11/780	Cytocomputer
Inner Product	1100x1280	500	627.7	10.5
Waveform Generator	1000x1000	400	513.0	7.5
Eight Bit ALU	1244x1099	600	631.5	10.3

Table 1.