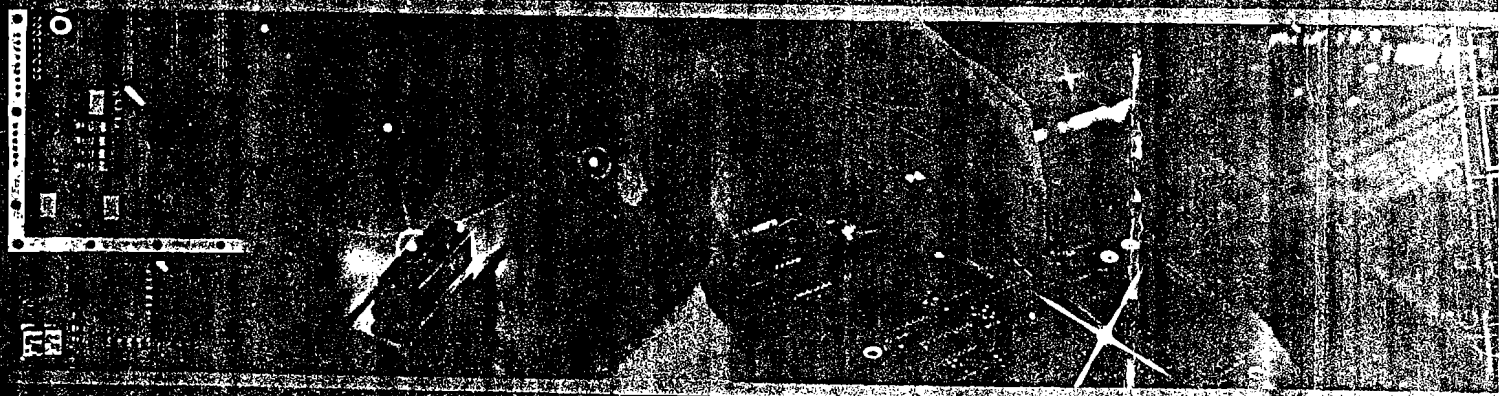
International Conference

On

"Emerging Techniques in Electronics, Computing Embedded System & VLSI Design in ICEVD - 08"



Sponsored By

University of Pune
Under the Quality Improvement Programme

Organised By

Department of Electronic & Telecommunication Engineering

PADMASHRI DR. VITHALRAO VIKHE PATIL COLLEGE OF ENGINEERING

Ahmednagar, Maharashtra.

Electronics & Telecommunication Engineering Dept. PDVVP COLLEGE, A.nagar

Prof. A.C. Suthar

International Conference

on

"Emerging Techniques in Computing, Electronics, Embedded System & VLSI Design" ICEVD - 2008

March 20-21st 2008

Sponsored By

University of Pune
Under the Quality Improvement Programme

Organized by:

Department of Electronics Engineering

Co-Sponsorers

Crompton Greaves
EVERYDAY SOLUTIONS

ni2
ni logic pvt. ltd.

Pamptron
Bio Medical Equipment P. Ltd.
Mumbai

Edited by

Prof. R. S. Deshpande **Prof.A.K.Khureshi**
Prof. S. M. Walke **Dr. A. J. Shirke**

Department of Electronics & Telecommunication Engineering
Padmashree Dr. Vitthalrao Vikhe Patil College of Engineering, Ahmednagar,
Maharashtra

Configurable Hardware Architecture for VLSI CAD Layout Design Rule Checking

Rakesh Gajre
Nirma University, Ahmedabad
rakeshgajre@gmail.com

Anil Suthar
C. U. Shah College of Engg. & Tech., Gujarat
acsuthar@yahoo.co.in

ABSTRACT

Design rule checking (DRC) is an important step in VLSI design in which the widths and spacing of design features in a VLSI circuit layout are checked against the design rules of a particular fabrication process. Some efforts was made to built hardware accelerators for DRC have been proposed, it is often impractical to build a different rule-checking ASIC each time design rules or fabrication processes change. In this paper, a configurable hardware approach for DRC. Because the rule-checking is built in configurable hardware. Here discussed an edge-endpoints-based method for performing Manhattan geometry checking; this approach is well-suited to the constraints of configurable hardware. Design rules do change over time.

Subject Areas:
Nanotechnology
Low power VLSI design

Configurable Hardware Architecture for VLSI CAD Layout Design Rule Checking

R.S.Gajre [1], A.C.Suthar [2], Dr.G.R. Kulkarni [3]
[1] P.G.Student, Department of E & C, Nirma University, Gujarat, rakeshgajre@gmail.com
[2] Asst.Prof, Department of E. &.C, C. U. Shah College of Engg. & Tech., Gujarat, acsuthar@yahoo.co.in
[3] Principal, C. U. Shah College of Engg. & Tech., Gujarat, grkulkarni29264@yahoo.com

Abstract:

Design rule checking (DRC) is an important step in VLSI design in which the widths and spacing of design features in a VLSI circuit layout are checked against the design rules of a particular fabrication process. Some efforts was made to built hardware accelerators for DRC have been proposed, it is often impractical to build a different rule-checking ASIC each time design rules or fabrication processes change.
In this paper, a configurable hardware approach for DRC. Because the rule-checking is built in configurable hardware. Here discussed an edge-endpoints-based method for performing Manhattan geometry checking; this approach is well-suited to the constraints of configurable hardware. Design rules do change over time.

1. Introduction

Integrated circuit die sizes have increased dramatically and simultaneously the smallest possible features on these dies have become much smaller as well. Hence Design Rule Checking has become more and more time-consuming and compute-intensive. The main difficulty with prior approaches, custom-hardware proposals has been their inflexibility. Fabrication processes evolve over time, with new layers or width/spacing rules being introduced. As such, design rule checkers implemented in hardware must be re-designed and rebuilt to address each set of changes. It has been noted that while design rules do change over time and vary between fabrication lines, their fundamental form remains similar. So the goal is to design a general-purpose skeleton for DRC that applies to nearly all fabrication design rules, and then also to tailor the rule-checking hardware for a particular fabrication process.
This paper presents a configurable hardware accelerator for design rule checking. To take the best advantage of the special characteristics of FPGAs, a new DRC methodology has been suggested. This edge-endpoint-based approach reduces the storage and sorting requirements for processing the layout files compared to prior work. This makes the approach particularly amenable to FPGAs and also reduces the cost considerably.

2. Design Rule Checking: Background and Related Work

Design Rule Checking takes as input a low-level description of the mask layers and features required for a particular VLSI design. Such data is produced by the CAD tool on which the layout was created. DRC identifies the places in the VLSI design, where design rules, such as the spacing between two features have been violated. Generally design rules are specified in terms of a parameterized width factor, referred as lambda. Design rules changes frequently and many fabrication processes, particularly in the sub-micron domain, will have subtly different design rules. There are two major types of design rule checking methods.

Bitmap methods were widely used in early approaches. The layout is rasterized into a grid of square cells, with each mask layer represented by a separate bit in each cell. Bit maps are attractive because of the simplicity of some operations such as Boolean operations (AND of two masks, for example) and space/width checking. Bit map approaches are based on Baker's algorithm. Bitmap approach has also few disadvantages. First one is that bitmap approach requires processing a large amount of data; this requires large amounts of memory bandwidth and high parallelism for data processing. The second disadvantage is that in a design system where the grid spacing is much smaller than the minimum feature size, we need a much larger window size to check for width or spacing errors.

Edge-based approaches use edges to represent regions in each mask layer. This will reduce the amount of data needed in general and is less dependent on the mask resolution. The edge files are divided into horizontal edges and vertical edges. Horizontal edges are checked for vertical width errors and horizontal spacing errors; vertical edges are checked for horizontal width errors and vertical spacing errors. The main disadvantage of this approach is that it requires non-deterministic amounts of

indicates the *direction* of the current endpoint. D = '0' if the endpoint is the *starting* endpoint of the associated edge. D = '1' if the endpoint is the *ending* endpoint of the associated edge. The P bit indicates if the endpoint has been processed before. For the endpoints generated from mask layer data, P = '0' and X-extension = 0.

All the endpoints are sorted in increasing order by X-coordinate, Y-coordinate, layer and orientation. Two endpoints in the initial edge file "cancel" if they only differ in the D bit. That is, after endpoints are sorted, they are divided into groups with the same X-coordinate. Within each group they are sorted in increasing order by their first three fields. Note in the endpoint representation, there is no X-coordinate information. Instead, a special endpoint of layer 0 is inserted before each endpoint group of the same X-coordinate and it records the X-coordinate value in its Y-coordinate field.

3.2 Scanline Maintenance

When the scanline is at a particular x coordinate Xc, we categorize and sort pending edges as follows:

1) Xmin = Xc, i.e., the edge starts from this scanline.
2) Xmax = Xc, i.e., the edge ends at this scanline.
3) Xmin < Xc < Xmax, i.e. the edge intersects this scanline.

Endpoints from the first two categories are called *new* endpoints with P = '0' since they have never been processed before. The points of the third category,

However, are "inherited" from the previous scanline. Each time a category-1 edge startpoint is processed, it is passed along to the next scanline. This point inherits all the information from the starting endpoint except that now P = '1' since they have been processed before. We refer to these points as *old* points. Old points are removed from the scanline when they meet the ending endpoints (category 2) of the edge.

3.3 Scanline Processing

Endpoints of the current scanline, stream into the processing unit as shown in Figure 3. There are several processing steps, as described below.

3.3.1 Selecting True Points

In the input data from the CAD tool, there can be overlapped and coincident regions in some layers. These regions should be logically ORed to get the final region profile (see Figure 4). A point, whether it is an old point or a starting/ending endpoint of an edge, is called a *true point*, if it is on the boundary of the final region. A true point is called a *true endpoint*, if it is one of the endpoints of the final region.

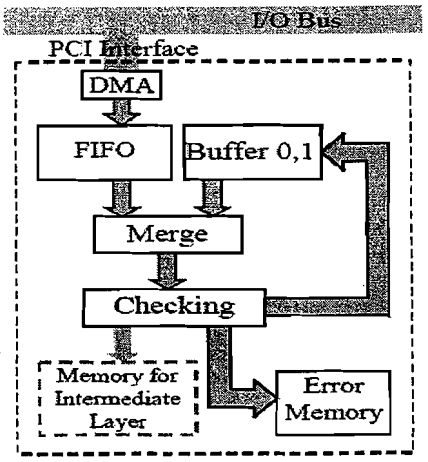


Fig. 3. Basic Processing Unit

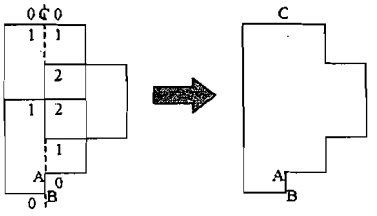


Fig. 4. Selecting True Points

To help select the true points, we keep two counters for each layer: LC and RC. When a point enters the processing unit, the counters corresponding to its layer are updated in the following way:

```
if(input point is a forward point)
switch (input point)
case"old point":LC++;RC++;
break;
case"starting endpoint" : RC++
; break;
case "ending endpoint": LC ++ ;
break;
else
switch (input point)
case "old point": LC — ; RC — ;
break;
case "starting endpoint": RC
— ; break;
case "ending endpoint": LC — ;
break;
```

our existing hardware framework is to extend some edges of the central box or the boxes around it by *d*, so that scanline processing will be able to find the error. We call these extended edges *virtual edges*. Each point on a virtual edge is a *virtual point*.

A virtual point inherits the Y-coordinate, Layer and Orientation information from the edge that generates it. Virtual points are old points, so P = '1'. Here used the D bit to differentiate virtual points and normal points. If D = '0', it is a virtual point; if D = '1', it is a normal point. Unlike normal points that are automatically cancelled when they meet the ending endpoints.

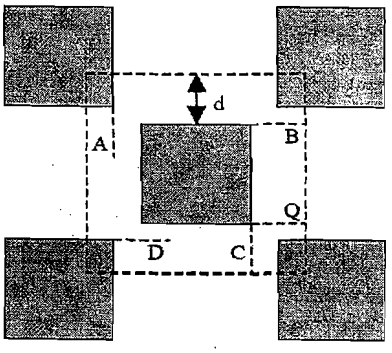


Fig. 6. Virtual Edges

virtual points expire according to their X-extension bits. When the virtual point is first generated, the X-extension bits record the value of d.

Assume the scanline moves to the right by d' after the current scanline has been processed, then the X-extension bits of all the virtual edges are updated by subtracting d' from themselves. If the value of the Xextension bits is less than or equal to zero, the corresponding virtual edge is cancelled. When multiple layers exist in the system, set d to the value of the largest value of minimum width/spacing requirement concerning this layer.

In order to minimize the number of virtual edges and to better accommodate virtual points in our method, we only generate virtual edges when:

1) The right side of a forward true ending endpoint is outside the region. This is used for space checking;

OR

2) The right side of a backward true ending endpoint is inside the region. This is used for width checking.

Thus in Figure 6, only B and D are generated as horizontal virtual edges, only A and C are generated as vertical virtual edges (Rotate the picture by 90 degree counterclockwise). Q will not be generated.

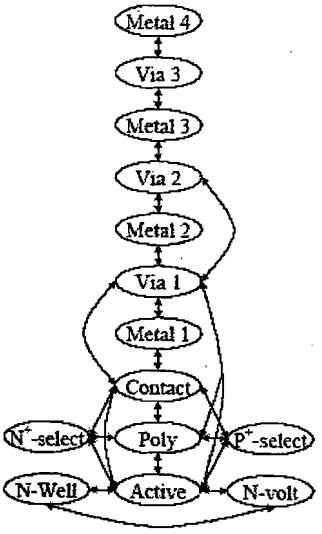
4. Hardware Architecture

Proposed hardware architecture is composed of one or more basic processing units (Figure 3).. The Memory in Figure 3 gets the edge endpoints from mask layers and the buffers store the "inherited" endpoints from the previous scanline. Since the processing unit generates the old points for the next scanline while consuming the old points generated by the previous scanline. Two buffers are used. Buffer 0 is being read while buffer 1 is being written and vice versa.

Merge unit is used to maintain the scanline. Since both the data from the memory and from the buffer are in their canonical order, the merge unit does a simple two-way merge sort to generate the final data stream for processing. The merge sort unit also takes care of "retiring" old points and virtual points. As mentioned in Section 3.2, old points are removed when they meet the ending endpoints. Each old point has the same representation as the corresponding ending endpoint except for the P bit, which is the least significant bit in comparison; thus it will definitely meet the ending endpoint in the merge unit and it will be cancelled then. For virtual points, as mentioned in Section 3.3.3, the Xextension bits of the virtual points are updated in the merge unit, and virtual points are cancelled if X-extension bits are less than or equal to zero.

The checking unit is different from each basic processing unit. Its basic structure is shown in Figure 5. For each layer, we need a set of counters and an additional subtractor. Thus the hardware cost grows almost linearly with the number of layers we send to each checking unit. The only exception is the number of design rules involved. If all layers processed by a checking unit interact with each other, the number of design rules is roughly quadratic to the number of layers. When an intermediate layer needs to be generated, the data from the checking unit will be fed into the memory of one or more subsequent basic processing units. Temporary storage is required to store this intermediate layer. The merge unit of the

Group 1: Poly, Active, N+-select, P+-select. Intermediate layers N-active and P-active are generated.
Group 2: N-active, P-active, N-well, P-well
Group 3: Poly, Active, Contact, Via1
Group 4: Contact, Metal1, Via1, Metal2, Via2



Group 5: Via2, Metal3, Via3, Metal4

Fig. 7. Layer Interaction Graph

6. Conclusions

Because of the intrinsic similarity between different design rules, the paper shows that hardware checking system for different design rules can be accommodated in a general scalable architecture. The variation between design rules of different fabrication processes makes configurable hardware an ideal candidate for the rule-checking unit in general architecture. This paper describes and validates an edge-endpoint-based geometry checking method for Manhattan structure that is amenable to configurable hardware implementation. In comparison with previous edge-based methods, this technique uses a smaller data path width, simplifies data flow control and easily handles the edge reconciliation problem.

As a verification of this general architecture, we analyzed the design rules of SCN4M_SUBM process from MOSIS and implemented the essential hardware for checking a subset of the design rules. The final hardware system runs at 33 MHz and offers a speedup of more than 25X over the conventional software run on the state-of-the-art microprocessors.

References:

[1] Eric C. Carlson and Rob A. Rutenbar, "A Scanline Data Structure Processor for VLSI Geometry Checking", *IEEE Trans. computer-aided design*, vol CAD-6, NO. 5, Sep. 1987

[2] C.M. Baker, "Artwork Analysis Tools for Integrated Circuits", MIT/LCS/TR-239, Master's Thesis, MIT 1980

[3] L. Seiler, "A Hardware Assisted Design Rule Check Architecture", *Proc. 19th Design Automation Conf.*, June 1982, pp. 235-238

[4] Steven M. Rubin, "Computer Aids for VLSI Design", 2nd edition, Addison-Wesley Publishing Company, 1994.

[5] R. Kane and S. Sahni, "A Systolic Design Rule Checker", Proc. *21st Design Automation Conf.*, Jun 1983, pp. 243-250

[6] T.G. Szymanski and C.J. Van Wyk, "Space Efficient Algorithms for VLSI Artwork Analysis", *Proc. 20th Design Automation Conf.*, June 1983, pp.734-739

[7] George E. Bier and Andrew R. Pleszkun, "An Algorithm for Design Rule Checking on a Multiprocessor", *Proc. 22nd Design Automation Conf.*, June 1985, pp. 299-304

[8] U. Lauther, "An O(N log N) algorithm for Boolean mask operations", *Proc. 18th Design Automation Conf.*, July 1981, pp. 555-562

[9] B.W. Lindsay and B.T. Preas, "Design Rule Checking and Analysis of IC Mask Designs", *Proc. 13th Design Automation Conf.*, June 1976, pp. 301-308

[10] MOSIS service, "MOSIS Scalable CMOS (SCMOS) Design Rules", Revision 7.2, http://www.mosis.org/New/ Technical/Designrules/dr-scmos72.html

[11] Zhen Luo, Margaret Martonosi, and Pranav Ashar, "A Configurable Hardware Design Rule Checker", Princeton University Department of Electrical Engineering. Technical Report #CE-99-1.