VLSI DESIGN
1996, Vol. 4, No. 2, pp. 141–147
Reprints available directly from the publisher
Photocopying permitted by license only

Hardware Design Rule Checker Using a CAM Architecture

SEOKJIN KIM and RAMALINGAM SRIDHAR*

Department of Electrical and Computer Engineering, The State University of New York at Buffalo, Buffalo, NY 14260

This paper presents a hardware implementation of design rule checker using a specialized Content Addressable Memory(CAM) for the Manhattan geometric designs. Two dimensional relationships between rectangular objects in a design are checked with one dimensional design rules. The input data is processed by the pixel pre-processor in such a way that direct comparison between the input data and the stored rules in the CAM is possible. The comparison by the CAM reduces the number of memory references and logic operations of pattern matching and the simple architecture of the system enables a low cost implementation.

Keywords: Design rule check, Hardware accelerator, Content Addressable memory.

INTRODUCTION

Special purpose hardware for electrical CAD has been studied for a number of years. Several experimental systems reported their gain over the general purpose hardware [1]. As the designs get larger, the CAD algorithms require more computing and data processing power and hence the opportunities for a hardware oriented solutions are increased. To prevent the drastic increase in the processing time, hierarchical or online design rule checking methods are often used during various design phases. However, in practice, there still exists the need for batch type design rule checking of the full design for the final verification. An efficient hardware implementation of the design rule check is valuable at this stage.

The mask layout of a design is represented as a set of images with multiple layers and can be visualized with bitmap data. The bitmap representation has several advantages in hardware implementation. It preserves the geometric shape and the location of an object and thus it does not need additional steps for conversion or modification of raw data, which gives inherent simplicity to the algorithm. The DRC algorithms which use raster scan on bitmap data method, utilizes a moving window of a certain size which scans the design onto a fine square grid. The combinatorial increase in the table size of the error templates, as the window size increases, has been a major limitation of the algorithms for practical applications. The existing hardware implementations often use a small size of window, typically 3×3 or 4×4 which

*Corresponding author. 135 Bell Hall, Department of Electrical and Computer Engineering, The State University of New York at Buffalo, Buffalo, NY 14260 Phone: (716) 645-2422 x2139 Fax: (716) 645-3656 E-Mail: rsridhar@eng.buffalo.edu

is not enough to accommodate a practical design rule and require two dimensional error templates [2][3]. A variable sized window approach was proposed, however the implementation is more complex than fixed size approaches [4]. Others use parallel processing technique, using a number of small pixel processing elements connected to each other [5]. Each input pixel maps directly onto the individual processing element, which performs simple logic operations with its neighbors. Even the simple processors need considerable amount of circuitry to perform logic operations and interactions among the neighbors. The limitation on the window size and the high cost of complex hardware are the factors which make the hardware implementation of the design rule check less attractive.

This paper presents a simple and thus a low cost hardware implementation based on the raster scan method for the designs in Manhattan geometry. The system uses a window whose size is big enough to hold the maximum constraint size in a design rule set. The window detects corners on a mask layout and repeats the one dimensional checking along these corners. This eliminates the need for two dimensional rule set table at the cost of multiple one dimensional checkings. The use of one dimensional templates avoids the drastic increase in the table size and thus relatively big window can be used. The system consists of a rule set table and a pixel pre-processor. The rule set table uses a specialized CAM to reduce the number of memory reference and to eliminate external logical comparisons. The pixel pre-processor is a two dimensional mesh of registers with surrounding logics and holds the pixel data underneath the window. The detailed structure and operations are explained in Sections 2 and 3.

2 CAM BASED DESIGN RULE CHECK

In the bitmap representation of the mask design data, each pixel has several bits, one corresponding to each mask layer. A value of 1 or 0 in each bit represents the existence or absence of the layer on the area covered by the pixel. The physical size of a pixel is usually the minimum feature size of a given processing technology and λ is often used as a unit size. Four basic design rules are width, spacing, enclosure, and extension rules. These rules use local context of one or two mask layers at a time. The window based checkers [2] [3] [4] exploit the locality by having a moving window which systematically scans through this bitmap pattern. The size of the window should be at least 1 unit (λ) larger than the size of the maximum constraint in the rule set. Possible error patterns are stored in the rule set table. The design rule check is done by comparing a bitmap pattern under process to the error patterns stored in the table. The use of conventional memory for the rule set table requires thorough search of the rule table for a matched pattern in the worst case and thus the memory access path becomes the limiting factor for the overall system performance. The Content Addressable Memory(CAM) has been successfully applied in the pattern matching or string matching operations where the stored information is retrieved based on the data itself, rather than by its storage location. The input data is compared to each of the memory words in a CAM in parallel and the result is returned usually in one clock cycle by a match output, matched data and address-(es) [6]. Due to the nature of the problem, a CAM naturally fits the design rule checking. In the proposed method, one dimensional rules are encoded and stored in a specially designed CAM. The input pixels from the pixel pre-processing element are broadcast and compared to the contents of the CAM. The CAM generates match output with corresponding address in case of any violation or match.

Even though the number of error patterns is finite for a given window, the combinatorial increase of the error patterns resulting from the increase of window size is a significant limitation of this approach [7]. However, considering that a design rule is a distance between any two objects, it can be expressed in one dimension even though the checking would be repeated in multiple directions or in a circular fashion with reference to a given pixel. The rule set represented in one dimension reduces the table size significantly. A pattern exists for each mask layer. Two of the layers of interest are scanned from left to right,

top to bottom one pixel at a time as shown in Figure 3. At the current pixel location, rule checkings are done in vertical and horizontal directions. Attempts are also made at the current pixel, vertical and horizontal end of the scan bars and the diagonal counterpart of the current pixel to detect corners in the pattern. If any of the four attempts detect corners, rule checkings are done repeatedly at the corners horizontally and vertically up to the size of the maximum constraint of the given layer. Two pairs of flag registers (Figure 5) are used to set the maximum constraint in each direction. Figure 4 shows an example of a corner and its checking flow.

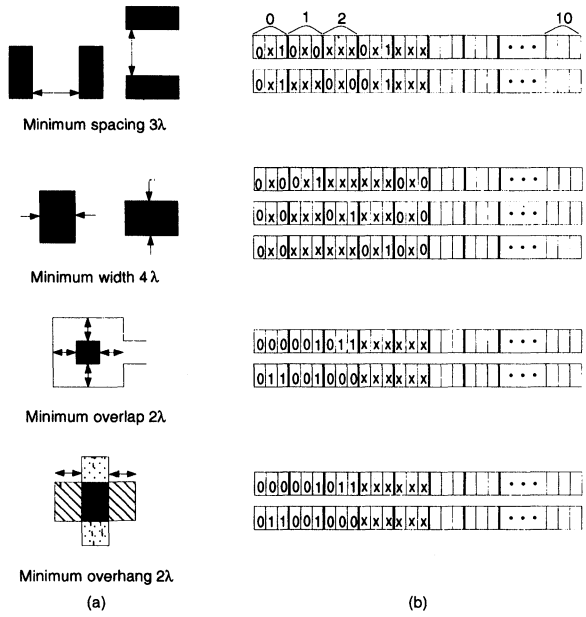


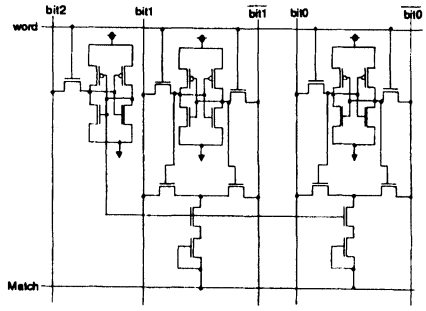
FIGURE 2 Examples of basic design rules

3 IMPLEMENTATION

The design rule checker is divided into two functional blocks, the rule set table and the pixel pre-processor as shown in Figure 5. Each rule in the rule set is encoded and stored in the CAM with a tag which holds a code of associated layer(s). A single word in the CAM represents a rule or constraint. The pixel pre-processor is used to pre-process incoming pixels so that direct comparison between the post-processed pixels and the rules stored in the CAM is possible.

3.1 CAM for Rule Table

Two mask layers of a design are processed for the design rule check and thus each pixel holds two bits of information underneath the pixel in pre-processing. To store and check the matching of the information, three bits of the CAM are needed for pixel information. The design rule encoding scheme is based on the possible values of a pixel; none, layer 1, layer 2, overlap of layer 1 and 2, and don't care Figure 1). Don't care condition is for those rules whose constraints are smaller than the size of the maximum constraint in the rule set. Three bits of the CAM cell are used to represent 5 logic values, similar to the method propsed by [8] [9] . The don't care condition is implemented by setting the most significant bit as 1 which sets the match line in the CAM cell as always high. Several rules and their encoded results based on the CAM specifications are shown in



Bit2	Bit1	Bit0	Logic value
0	0	0	0 (white)
0	0	1	1 (layer 1)
0	1	0	2 (layer 2)
0	1	1	3 (layer 1,2 overlap)
1	–	–	X (don't care)

FIGURE 1 5-valued CAM cell for a pixel

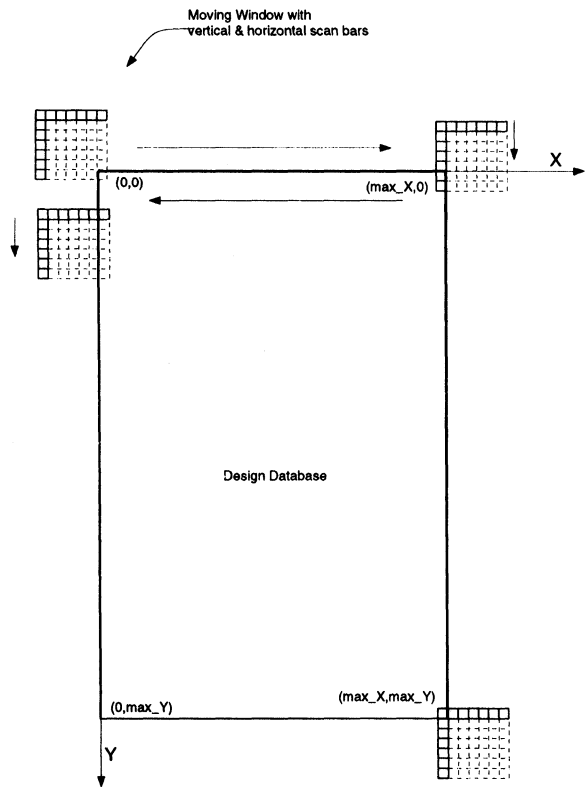


FIGURE 3 Mask scan

Figure 2. The size of a word is fixed to be three times the size of the maximum constraint in a rule set. The size of the CAM cell array is determined by the number of rules in a rule set. In our prototype implementation of the basic MOSIS's SCMOS rules, the size of a word is 33 bits as the size of maximum constraint is 10λ and the CAM size is chosen to be 64 words. A word in the layer selector has 6 bits which enables the CAM to have up to 64 logical blocks.

Figure 1 shows the schematic diagram of a single "pixel bit". The match line is precharged before the compare operation and the diode-connected N-type transistor prohibits the connection to the neighboring bits. If a given input matches with the value stored in a CAM cell, discharging occurs. If the pixel bit has the don't care value, the transistors are turned off between the precharge line and the discharge path, and the pixel bit does not affect the comparing operation at the neighboring bits. Simulation result of match, mismatch and don't care conditions are shown in Figure 9. In the simulation using SPICE3, four bits are written with values of 0, 1, 2, 3 and tested with the values under the normal condition and the don't care condition.

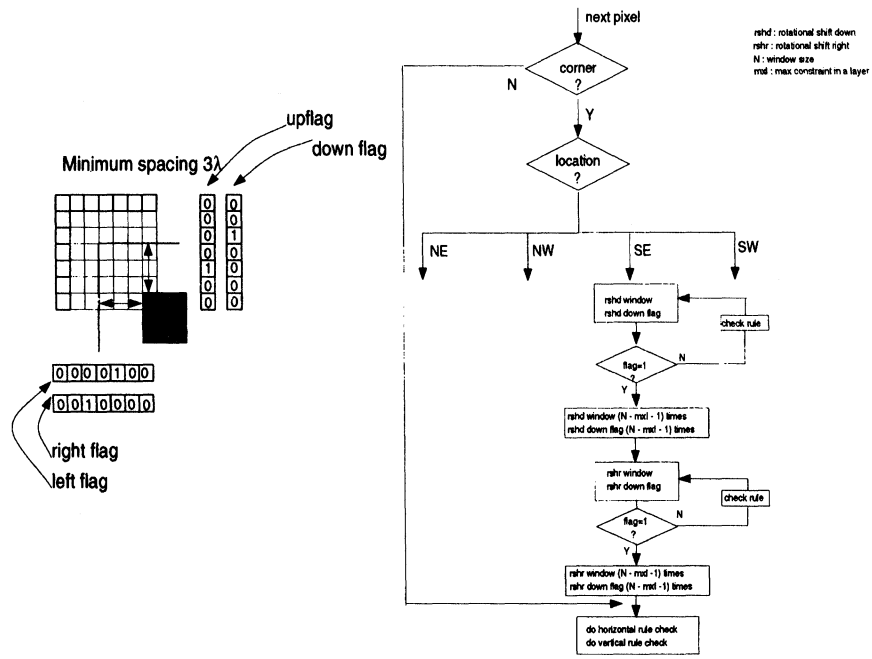


FIGURE 4 Corner checking procedure

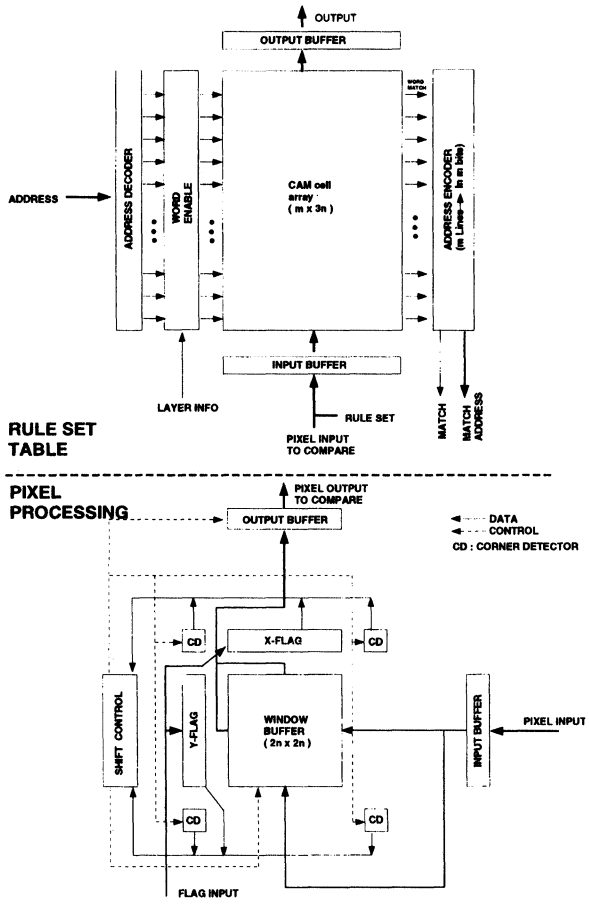


FIGURE 5 System architecture

Since the design rules checking is done for two layers at a time and the rule set table holds the rules for all layers, the CAM needs to enable the processing of the corresponding rules to the layers before the checking operation. The layer relationship to a rule is stored in the word enable array (Figure 5). If a given layer information matches one or more words stored in the layer select array, the corresponding words of design rule are enabled and ready to compare with the input data. The detailed circuit diagram of a layer select word is shown in Figure 6. The memory cells in the layer select array is the same as that of the conventional static CAM which performs the exact match operation. The match sense amplifier senses and holds the layer match status. The inverted output of a match sense amplifier enables or disables the precharge for the subsequent word in the rule set array. The CAM based rule set table is designed using 2μ double metal CMOS design rule for fabrication through MOSIS.

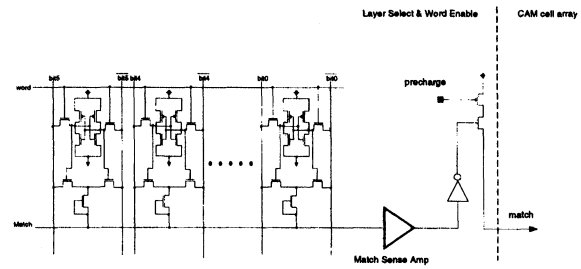


FIGURE 6 Layer select and word enable

3.2 Pixel Pre-processor

The pixel pre-processor consists of a window buffer, two pairs of flag registers, four corner detecting circuits, input and output buffers and a control part. Two modes of checking are available in the rule checking operations. A sequential mode checks the pixels in X-direction and Y-direction of the current reference pixel sequentially. After the two directional checks, the window moves by 1 pixel and repeats the checks. Prior to the horizontal and vertical checks, the diagonal checking operation can be activated by detecting the existence of corner patterns at the four corners of the window. The corner detectors are designed to detect the conditions of three adjacent pixels at each corner of the window which determine whether any of the corner pixels of the window is the corner pixel of a mask layer. The conditions are similar to that of corner based DRC algorithms [10]. If a reference pixel at a corner of the window is in one quadrant and all three adjacent pixels in the other three quadrants are of the same type but different from the reference pixel, the reference pixel is regarded as a corner in a mask layer. The conditions and schematics of the northeast corner detector is shown in Figure 7(a) and 7(b) respectively.

If one or more of the four reference pixels are at the corner, the surrounding pixels within the maximum constraint of the given mask layer are checked for violation serially. The serial shifts of the surrounding pixels occur in two directions: horizontal

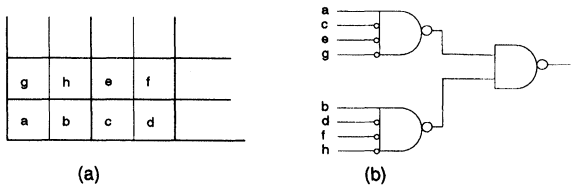


FIGURE 7 Bit locations at Southeast corner and the corresponding corner detector

and vertical. For example, if there is a corner at the northwest corner of the window, the shift-lefts are done followed by shift-ups. The circular shift of the window enables the reference pixel to roll back to its original position after 22 shift operations in one direction and the shift to another direction can start with the original pixel data in the frame. The block diagram of the window buffer and the schematic diagram of a single register are shown in Figure 8(a) and 8(b), respectively. The pixel pre-processor is designed using Xilinx XC3064 FPGA.

4 COMPARISON AND PERFORMANCE ESTIMATION

The advantages and disadvantages of the proposed CAM based design rule check are summerized below as compared to the existing methods.

- The use of CAM reduces the number of memory references and does not require additional hardware for the compare operation.
- The size of the window is large compared to other window based methods [2][3] enough to hold a practical design rule and the size of rule table is not increased drastically since one dimensional checking is used.
- The simple architecture enables a relatively low cost implementation compared to the variable size window method [4] and the simple parallel processing element method [5].
- The sequential corner checking method may degrade the performance when a large number of corners are present in a design.

The performance of the system solely depends on the speed of the pixel pre-processor if there is no I/O latency as the CAM performs match operations in parallel. The number of clocks to process one frame of window varies according to the number of corners under the window. If corner is not present, the pre-processor passes two pixel words in three clock cycles. It takes one clock cycle to detect corner conditions as the four corners of the window are checked simultaneously. The number of shift operations when the pre-processor meets a corner is determined by the maximum constraint size of the layer being processed. The total number of clocks to process a window can be calculated by (1).

$$N_{clock} = 3 + 2c(mc + 1) \tag{1}$$

where c is the number of corners and mc is the maximum constraint size of the layer. Assuming that there exists one corner in each window and the maximum

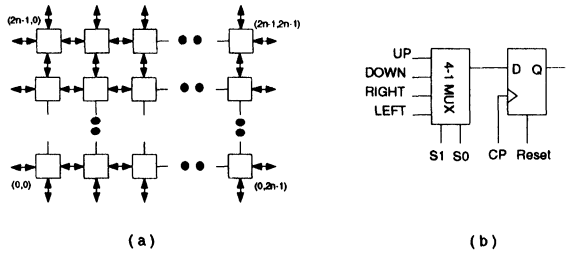


FIGURE 8 (a) Window buffer (b) single register in the window buffer

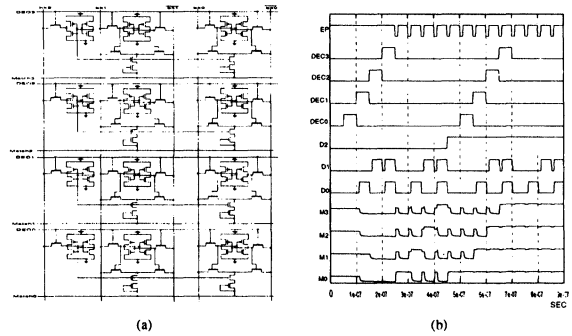


FIGURE 9 (a) 5-value CAM cell test circuit (b) Simulation result

constraint size is 10 λ, 47 clocks are needed to process a window and 1.2 = 10^6 λ^2 of every two layers can be processed in one minute with 10 MHz clock speed.

The number of the CAMs required depends on the number of rules in a rule set. The CAM is designed with the size of 64 words, and thus a single CAM can hold upto 64 individual rules. For example, two CAMs can store the SCMOS design rule provided by MOSIS which has 118 rules. The multiple use of the CAMs do not affect the search cycle time, as the search operation is performed in parallel.

CONCLUSION

This paper has presented a hardware implementation of bitmap oriented design rule checking. The system uses a specialized CAM as a design rule set table and preprocessed input pixel data are compared with the content of the CAM. The use of CAM reduces the need for frequent memory accesses, as well as complex template matching which are often required in the existing systems. A low cost implementation is made possible by its simple architecture. By reprogramming rules in the CAM, the system can be applied to different process technologies with ease. The specialized CAM is designed as a custom chip and the pixel pre-processor is designed on to the XC3064 Xilinx FPGA device.

References

[1] T. Ambler, P. Agrawal, and W. Moore, *Hardware Accelerators for Electrical CAD*. Adam Hilger, 1988.

[2] L. Seiler, "A Hardware Assisted Design Rule Check Architecture," in *Preceedings of the 19th Design Automation Conference*, pp. 232–238, 1982

[3] R. Hartenstein, R. Hauck, A. Hirschbiel, W. Nebel, and M. Weber, "PISA, a CAD package and special hardware for pixel-oriented layout analysis," in *Preceedings of the Internaltional Conference on Comuter Aided Design*, pp. 260–262, 1984.

[4] R. Hartenstein, A. Hirschbiel, and M. Weber, "MOM-Map Oriented Machine," in *Hardware Accelerators for Electrical CAD*, 1988.

[5] T. Blank, M. Stefik, and W. vanCleemput, "A Parallel Bit Map Processor Architecture for DA algorithms," in *Preceedings of the 18th Design Automation Conference*, pp. 837–845, 1981.

[6] L. Chisvin and R. J. D. author, "Content-Addressable and Associative Memory: Alternatives to the Ubiquitous RAM," *IEEE Computer*, vol. 22, pp. 51–64, July 1989.

[7] B. Preas and M. Lorenzetti, *Physical Design Automation of VLSI Systems*. The Benjamin/Cummings Publishing Company Inc., 1988.

[8] J. P. Wade and C. G. Sodini, "A Ternary Content Addressable Search Engine, " *IEEE Journal of Solid State Circuits*, vol. 24, pp. 1003–1013, August 1989.

[9] F. P. Herrmann, C. L. Keast, K. Ishio, J. P. Wade, and C. G. Sodini, "A Dynamic Three-state Memory Cell for High-Density Associative Processors, " *IEEE Journal of Solid State Circuits*, vol. 26, pp. 537–541, April 1991.

[10] M. H. Arnold and J. K. Ousterhout, "Lyra: A New Approach to Geometric Layout Rule Checking," in *Preceedings of the 19th Design Automation Conference*, pp. 530–536, 1982.