Algorithms for Corner Stitched Data-Structures

Mark A. Shand

Abstract. Corner Stitching was first presented by Ousterhout as a data-structure for VLSI CAD. This paper describes the data-structure in detail. It presents, in greater depth than previously, algorithms for the basic operations described in Ousterhout's original paper. The algorithms for enumerating and updating arbitrary rectangular areas are new. Their constant space complexity bounds are an improvement over previous algorithms for those operations that were recursive. From a practical standpoint, the elimination of the recursion has also made them much faster.


1. Introduction. Corner Stitching is a data-structure for representing rectangular regions in a plane. It was originally developed by Ousterhout [1] as a data-structure for CAD of VLSI layouts. Many traditional data-structures for this application treat regions as objects and hold these objects in some global structure. Corner Stitching represents regions by simply associating values with areas, and linking these areas only to their neighborhood. This allows many algorithms to operate on a region in time typically linear with respect to the area of the region.

In CAD of VLSI layouts the most important interactions occur between mask features lying in close proximity. One of the most computationally demanding aspects of layout CAD is the verification of mask geometries; minimum widths and separations dominate design rule checking, and abutment and overlap of geometries dominate the extraction of circuit descriptions. Similarly in constructive operations, such as compaction and channel finding, examination of neighborhoods plays a key role. Corner Stitching, by linking regions to their neighborhood, is able to provide cost-effective operations suited to the needs of VLSI CAD.

Ousterhout's original definition of Corner Stitching gives special treatment to regions in the plane representing background values. The definition in this paper treats regions with differing values uniformly. This allows a cleaner interface to the Corner Stitched geometric database. The uniform treatment also brings Corner Stitching into sharper contrast with traditional data-structures used for the representation of two-dimensional rectangular geometric data.

The remainder of the paper considers the basic algorithms underlying Corner Stitching. We present algorithms to find the database record representing the area containing a given point (Point Finding), to enumerate all records intersecting a rectangular region (Area Enumeration), and to change the values in a rectangular

1 This research was supported by the Australian Government Postgraduate Research Awards Scheme and Xerox Corporation.

2 Basser Department of Computer Science, The University of Sydney, Sydney, New South Wales, Australia 2006.

Received August 20, 1985; revised February 10, 1986. Communicated by Mark R. Brown.
region (Change Tile, Change Area, and Functionally Change Area). The Change Tile algorithm is presented in greater depth than when presented previously by Ousterhout (it was called Tile Deletion in his paper). The Area Enumeration, Change Area, and Functionally Change Area algorithms are new. They require only constant space and are improvements over earlier recursive algorithms that performed similar operations.

2. Contrast with Existing Techniques. In considering data-structures to deal with a VLSI layout, one must consider the characteristics of the VLSI layout itself. A VLSI layout is a two-dimensional pattern of shapes indicating regions where certain materials will be deposited on a planar substrate in order to create some particular electrical circuit. Normally, layout is described by a hierarchical collection of cells, where each cell contains a collection of geometrical shapes and instances of subcells. Of particular interest are the data-structures used to represent geometry within a cell. For the purposes of this discussion the Manhattan restriction is applied whereby the boundaries of shapes in the layout are constrained to be parallel to the coordinate axes.

One way to represent a region whose boundary consists of horizontal and vertical lines is as the union of a set of rectangles. In fact early mask-making machines, which transcribe circuit patterns to physical materials, achieved their goal by exposing a series of rectangles in a photographic emulsion. Not surprisingly then, the standard VLSI interchange format CIF ([2], § 4.5, pp. 115-127) describes the contents of a cell in a VLSI layout as a collection of objects. The allowable objects are boxes and composite shapes that may be described by boxes, such as wires and polygons. The pattern formed is defined to be the union of all the objects (rectangles) in all the instances of cells.

Traditional data-structures used for VLSI CAD reflect the object-based view of layout. In structures such as linked lists, bins, (k-d)-trees [3] and quad-trees [4] objects are represented explicitly, and the structures just provide an access path to these objects. The algorithmic complexities vary, but the basic principle of providing a global access structure to an object-based representation of layout is shared. Often the objects are restricted to rectangles only. Layout analysis tools working from an object-based description of layout can expend considerable effort sorting objects to find the union of these collections of uncorrelated rectangles [5].

Most layout editing programs also reflect a view of layout as a collection of objects. Icarus [6] and Kic [7] are examples of such editors. The interface to these editors allows the user to create, select, move, and delete boxes and other objects, these objects may overlap arbitrarily yet each remains distinct. Caesar [8] is a layout editor that is not of this type. The Caesar interface is based on filling and erasing rectangular areas of the layout, the sequence of rectangle transactions committed by the user are not accessible. Internally Caesar used a simple linked list of rectangles to keep track of the geometric shapes created in the layout. Thus Caesar departs from the rectangle approach in its interface, but not internally. Corner Stitching arose from the search for a more powerful and natural internal structure for an editor with a Caesar-like interface [9].