Compaction of a Routed Channel on the Connection Machine

Shantanu Ganguly
Vijay Pitchumani
Dept. of Electrical and Computer Engineering
Syracuse University, Syracuse N.Y. 13244

Abstract

A parallel algorithm for symbolic compaction of two layered channels has been developed and implemented on the Connection Machine. It allows fast channel compaction with very little wasted space.

Introduction

Channel routing is an important problem in CAD. Many channel routing algorithms have been reported in literature. Several channel routers, especially those that do not use doglegs, provide solutions that can be further compacted to free up a considerable amount of space.

Several compacted channel routers have been developed. Deutsch [2] has reported a channel compaction algorithm. Chen and Kuh [3] have developed Glitter, a variable width, two layered gridless channel router. Cong and Wong [7] have developed routing solutions that are suitable for further compaction.

However, all of these are serial algorithms. We have developed a parallel algorithm for compaction of channel routing, and have implemented it on the Connection Machine. This algorithm allows the use of two layers of metal in any direction, and the initial solution can have doglegs. The compaction procedure does not introduce any extra vias in the channel. The execution time of each iteration of compaction is essentially independent of the channel size.

Connection Machine

The Connection Machine [4] manufactured by Thinking Machines Corporation is an SIMD machine.

The front end is a VAX 8800, with the Connection Machine as a rear end co-processor. There are two versions of the machine - CM1 with a clock rate of 4MHz, and CM2 with a clock rate of 6 MHz. The machines can have upto 64k one bit processors.

The CM processors can be configured as a two dimensional NEWS-grid of varying sizes. There are two modes of communication: nearest neighbor communication on the NEWS-grid, and communication between two arbitrary processors through a General Purpose Routing Network (GPRN). The nearest neighbor communication is significantly faster than the GPRN, which has a communication delay of O(log(size of grid)). Our algorithm was tailored to use the NEWS-grid of the CM for all but one step. This enables us to obtain essentially constant time for each iteration, regardless of the size of the channel being compacted.

An Overview of the Channel Compaction Algorithm

The compaction algorithm is tailored to exploit the constant time characteristic of the NEWS grid operations of the Connection Machine. The channel is represented by a grid of processors, where each processor represents the intersection of a track and a column of the channel in question. For Deutsch's Difficult Example, the channel has 174 columns, and about 20 to 30 tracks (depending on the quality of the initial solution); a channel of this size can be easily mapped on the CM grid.

Each processor stores the number, layer, and direction of the nets running through the particular intersection. Our representation does not allow for variable wire widths at present. When a segment of a wire moves down (up) during compaction, the processors storing information of this wire segment relinquish it to the processors immediately below (above) them. Since the processors operate in SIMD mode, they may in turn receive net segment information from above (below) them.

A few characteristics of the algorithm are as follows:

- We refer to the outer loop of the algorithm as a *pass*. In each pass, there are two *phases*. Downward compaction is executed in the first phase, and upward compaction in the second phase of each pass. The direction of compaction alternates between downward and upward, since there are cases in which compaction in one direction alone does not free the most amount of area. Each phase consists of an inner loop and includes several iterations. Compaction by a single track is done in each iteration.
- In each iteration, a wire segment may move by one track. Longest movable spans are identified and moved. Doglegs are created at the boundary between the moved and static portions.
- The doglegs are created in the same layer of metal as the net being moved down, to avoid creating vias. Also, creation of vias may not be possible if the other metal exists at the new track. Our approach will generally result in a longer span of wire being compacted.
- Compaction is done in one direction until no part of any net can be moved. Then the tracks that have been freed are removed, and compaction is done in the reverse direction. Our current implementation is terminated after one phase since we have found that a second phase is not very likely to free up more tracks. Other termination criteria are also possible.

The complexity of the algorithm is as follows :

1. The number of passes required for a problem is dependent on both the problem size and the quality of the initial solution.
2. The number of iterations of each phase has an upper bound equal to the number of tracks of the channel, since that is the maximum number of times a net can move in a phase.
3. The time for each iteration, to determine the spans of nets that can be moved, is constant.
4. The time for the determination of the termination criterion is O(log(number of tracks $\times$ number of columns)).

Movability Criterion

The criteria that determine which part of a net can be moved are described in this section. A segment of a net is contained in a chain of processors, which are all on the same row of the grid. Since the smallest segment may have a span of two columns, the space freed by compacting a net segment must be at least two columns wide to be useful; otherwise, no other segment would be able to move into the vacated space (Fig. 1-2). Hence, at least four consecutive processors must move down together to free a space large enough for another net to move into (Fig. 3). Since the processors at the ends of the moved segment form doglegs, they do not free any space during compaction.

The above criterion is true for a segment in the middle of a net. However, the end points and bends of a net however, are exceptions to this rule. If a segment at the end of a net is to be moved down to create usable space, the segment needs to be only two columns wide. In this case only one dogleg is formed, so even if one column is vacated, it may be of use when added to other free grid points adjacent to the end of the net (Fig. 4-5).

The compaction may have to create doglegs at one or more ends of the moved segment to maintain continuity with the static portion of the net. In certain cases, vertical segments are created at the via locations of the moved segment. If, at a via location, the vertical wire segment extends in the direction of compaction, the via is moved if the net is moved. On the other hand, if the vertical wire segment at the via extends opposite to the direction of compaction, it may not be possible to move the via. Our first choice is to move the via with the net. If this is not possible, the via is not moved, and a vertical piece of wire connecting the via to the moved segment is created (Fig. 6-7).

Calculation of Movability

A segment is moved if it is at least four processors wide, so that the smallest possible net (which is two processors wide) can move into the vacated space. The recognition of movable segments is done by computing in parallel a variable called *weight* for each net of each processor. Based on the weight of a net and that of its extensions to the east and west, the processor determines whether the net can be moved to the next track processors. The recognition of movable net segments, and their movement both take constant time.

Implementation

The compaction algorithm has been implemented on the CM2 using C*. The NEWS-Grid package macros were used for interprocessor communication.

The front end VAX 8800 reads the input and initializes the processors on the grid with the information of the appropriate track-column intersection. Once the entire channel has been stored on the grid, the compaction algorithm proceeds in parallel for every processor. When the compaction has been completed, a parallel graphic program can be executed to obtain a graphic image of the compacted channel.

In the following text, the variable *movable* of a processor indicates if any net of the processor can be moved. The program has the following structure:

```
read in initial routing of channel
store information on the grid
do in parallel{
    do { Calculate downward movable,
    for each net within processor.
        move-all := ∨ (movable of all
    processors in the grid )
        if(move-all)
          compact-down
    }while{move-all}

    remove empty tracks at top

    do { Calculate upward movable, for
    each net within processor.
        move-all := ∨ (movable of all
    processors in the grid )
        if(move-all)
          compact-up
    }while{move-all}

    remove empty tracks at bottom

}while(tracks are removed from top or
    bottom)
```

In the above algorithm, we use the GPRN for the logical OR reduction involving the computation of move-all. This has a complexity of O(log(number of grid points)). Use of the NEWS-grid for the OR-reduction takes O(no. of tracks + no.of columns), which is a higher order of complexity. The Connection Machine provides freedom to mix the use of the two communication schemes as needed, to maximize the speed of execution. Except for the computation of move-all and the detection of freed tracks, the NEWS-grid is used for all communication, resulting in high speed.

Results

Results from three examples of channel compaction are reported in the table below. The first is a small example with an initial routing of seven tracks ($d_{max} = 6$); the other two are a 28 track non-dogleg solution of the Deutsch's Difficult Example (Fig. 8), and its inverted problem. The results are tabulated below. The timing is on the CM2 version of the Connection Machine.

Compaction Results				
Name of Ex.	Number of Cols.	Number of Tracks	Time (secs)	Iterations
Small	32	7 to 6	0.63	5
Deutsch	172	28 to 21	2.27	25
D's rev.	172	28 to 21	2.12	24

Future Work

Several postprocesing stages are possible for this algorithm. The algorithm creates several unnecessary bends, and straightening these will result in metal minimization. Shifting of vertically adjacent vias will result in smaller channel height. Lastly, it may be possible to exchange layers and remove some vias. We hope to implement parallel versions of these stages.

At present our algorithm is restricted to *symbolic* compaction of wires. We plan to enhance it to handle design rule constraints of feature sizes and spacings.

Summary and Conclusion

Channel compaction can markedly increase the quality of channel routing. The NEWS-grid of the Connection Machine is well suited for the problem. The steps to check for termination, and to check for vacated tracks are O(log(number of grid points)); all other steps are constant time. So the execution time per iteration will grow very slowly with problem size. The number of iterations needed is bounded by the number of tracks in the initial routing. Hence, the overall execution time increases relatively slowly with problem size.

Acknowledgement

This work was conducted using the computational resources of Northeast Parallel Architecture Center (NPAC) at Syracuse University, which is funded by and operates under contract to DARPA, and the Air Force System Command, Rome Air Development Corporation (RADC), Griffis AFB, NY, under contract # F30602-88-C-0031.

References

1. T.Yoshimura and E.S.Kuh, 'Efficient Algorithms for Channel Routing', IEEE Transactions on Computer Aided Design, Vol. CAD-1, no.1, 1982, pp. 25-35.

2. D.Deutsch, 'Compacted Channel Routing', Proceedings of the IEEE International Conference on Computer Aided Design, 1985, pp. 223-225.

3. H.H.Chen and E.S.Kuh, 'Glitter, A Gridless Variable-Width Channel Router' - IEEE Transactions on Computer Aided Design, Vol. CAD-5, no.4, 1986, pp. 459-465.

4. W.D.Hillis, *The Connection Machine*, MIT Press, Cambridge, Mass. 1985.

5. Thinking Machines Corporation *C* Release Notes*, Cambridge, Massachussets, 1987.

6. C.K.Cheng and D.Deutsch, 'Improved Channel Routing by Via Minimization and Shifting', 25th Design Automation Conference, 1988, pp. 677-680.

7. J.Cong and D.F.Wong, 'How to Obtain More Compactable Channel Routing Solutions', 25th Design Automation Conference, 1988, pp. 663-666.

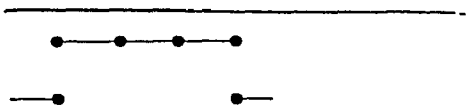


Fig. 1 A no-move case

Fig. 2 Another no-move case

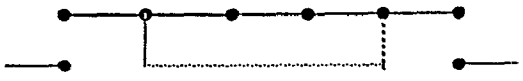


Fig. 3 A move case

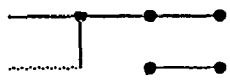


Fig. 4 A move case for a net end

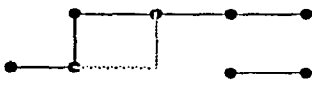


Fig. 5 A move case for a bend

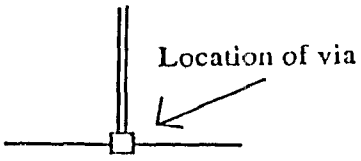


Fig. 6

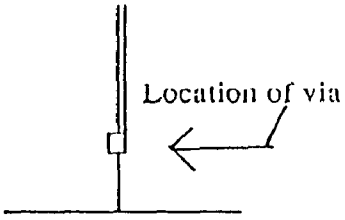
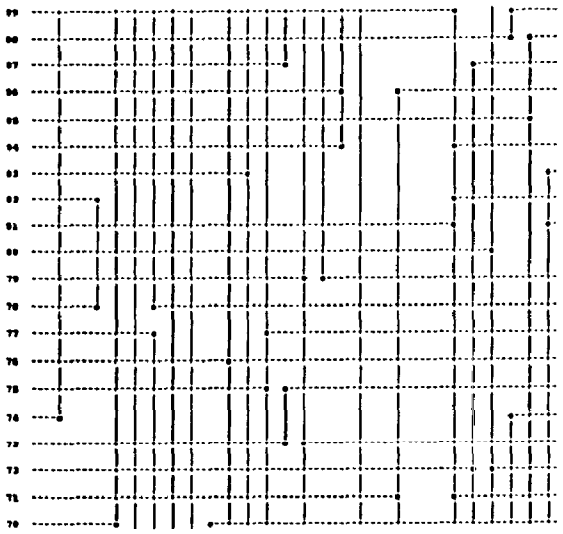
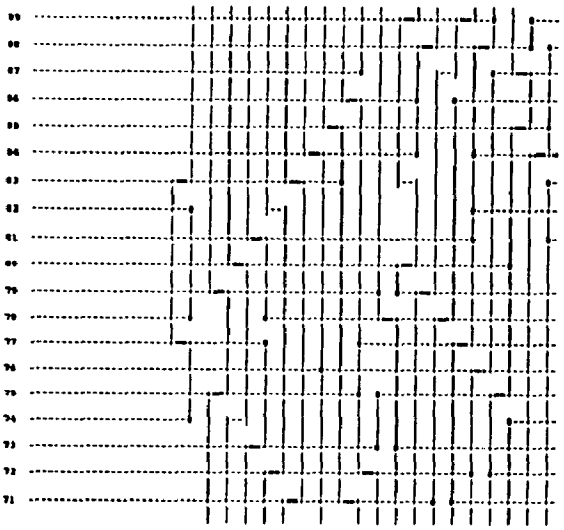


Fig. 7



Before Compaction



After Compaction

Fig. 8 A dense portion of Deutsch's Difficult Example.