Buffer Allocation Algorithm with Consideration of Routing Congestion

Yuchun Ma\textsuperscript{1}, Xianlong Hong\textsuperscript{2}, Sheqin Dong\textsuperscript{1}, Song Chen\textsuperscript{1}, Yici Cai\textsuperscript{1}, Chung-Kuan Cheng\textsuperscript{2}, Jun Gu\textsuperscript{1}

\textsuperscript{1}Department of Computer Science and Technology, Tsinghua University, Beijing, China
\textsuperscript{2}Department of Computer Science & Engineering, University of California, San Diego, La Jolla, CA 92093-0114, USA

Abstract - The dominating contribution of interconnect to system performance has made it critical to plan the buffers and the routes resource in the early stage of the layout. In this paper, we present a congestion estimation model which takes the buffer insertion sites into consideration. Based on the feasible region of the buffer insertion, the two-level tile structure is used to represent the distribution of the feasible buffer insertion sites among the routing tiles. And the buffer allocation method is performed based on the congestion estimation, which can find the buffer locations with good congestion result. Our approach can be embedded into the floorplanning process and the experimental results show the efficiency of our method.

I. Introduction

Due to the recent advances in VLSI technology, the importance of the interconnect delay has increased in the overall performance of a circuit. Buffer insertion has shown to be an effective approach to achieve timing closure. Since buffers are implemented by transistors, they cannot be placed over the existing circuit blocks. Placing a large number of buffers could significantly impact the chip floorplan, especially the routability of the chip. It is very useful that a good planning of the block positions can be obtained during the floorplanning stage so that buffers can be inserted wherever needed in the later routing stages.

Several previous works addressed the interconnect issues in floorplanning design. Cong\textsuperscript{[4]} defined the term “feasible region” (FR) of a net, that is, the largest polygon in which a buffer can be inserted such that the timing constraint can be satisfied. Sarkar\textsuperscript{[3]} added the notion of independence into feasible regions so that the feasible regions of different buffers on a net can be computed independently. These two papers gave the basic idea of Feasible Region for the buffer insertion. Tang and Wong\textsuperscript{[2]} proposed an optimal algorithm based on net flow to assign buffers to buffer blocks assuming that only one buffer is needed per net. Ma et al.\textsuperscript{[1]} gave a 2-step method to figure out the FR effectively which is integrated in the optimization process of floorplanning. Alpert et al.\textsuperscript{[3]} made use of tile graph and dynamic programming to perform buffer block planning. Lou et al.\textsuperscript{[10]} applied probabilistic analysis to estimate congestion and routability, and they showed that their estimations correlate well with post-route congestion. However their congestion model did not take into account buffer insertions. F.Y. Young\textsuperscript{[3]} gave the congestion estimation method based on the distance constraint of the buffer insertion. The accuracy of the estimation heavily depended on the grid sizes. Ma et al.\textsuperscript{[12]} gave a dynamic planning method to evaluate the buffer insertion and the congestion with the buffers. Both Ref.\textsuperscript{[12]}\textsuperscript{[13]} are based on the routing grids. Since the sizes of the buffers should be very small, the estimation based on the grids is often very coarse when the running time is tolerable.

The buffer insertion affects the congestion greatly. The buffer insertion sites considered. The nets are assumed to be routed over the cell in multi-bend shortest monotonic Manhattan distance. The buffer allocation method is applied to reduce the routing congestion with given space capacity, while the monotonic feature is maintained at the same time.

The rest of the paper is composed as follows: Sect.2 gives the definition of IFRs which is used as the buffer planning model; Sect.3 gives the congestion model; the buffer allocation algorithm is implemented with congestion involved in Sect. 4. The experimental results are shown in Sect.5.

II. Feasible Buffer Insertion Sites

In this paper, we suppose that the buffer insertion sites are located in the dead space between the circuit blocks. The feasible buffer insertion sites can be obtained based on the computation of the feasible region. Cong et.al\textsuperscript{[4]} introduced the concept of Feasible Region (FR) for buffer insertion, and presented analytical formula to compute FR. The feasible region for a buffer ‘b’ is the maximum region where ‘b’ can be located such that by inserting buffer ‘b’ into any location in that region, the delay constraint can be satisfied. Sarkar et.al\textsuperscript{[3]} gave the notion of independent feasible regions (IFR) and the IFRs of buffers belonging to the same net do not overlap each other.

The 2-D IFR of a buffer is defined as the union of the 1-D IFRs of that buffer on all monotonic Manhattan routes between source and sink. Since the feasible region will be reduced by the circuit blocks, the feasible region for buffer insertion in the packing is a very complex polygon, normally concave polygon. We partition the dead space into rectangular blocks to capture the constraint that the circuit blocks prevent the insertion of a buffer. Taking advantages of the dead spaces blocks in the packing and the feature of the FR, Ma et al.\textsuperscript{[3]} gave a two-step method to figure out the possible insertion sites for the buffers very efficiently.

III. Congestion Model

The congestion model employed is essentially a 2-D rectangular tile based probabilistic map assuming the nets are routed over the cell in multi-bend shortest monotonic Manhattan distance. The congestion of a routing tile T(x,y) is defined as the expected number of routes passing through T(x,y). The probability for each possible route of a wire is equal in the traditional congestion model. However, the buffer insertion may influence the routing ways greatly. If the probabilities of successful buffer insertion are different at different routing tiles, the probabilities of some routes will be higher if they run through those tiles with higher probabilities of successful buffer insertion.

In our approach, we try to address the buffer insertion with a new congestion model. The probabilities of the routes

\textsuperscript{1}This work is supported by the NSFC 90307005 and NSF of USA CCR-0096383, NSFC and HK GRC joint Grant No.60218004, the National Natural Science Foundation of China 60076016 and 863 Hi-Tech Research & Development Program of China 2002AA1Z1460, the National Doctoral Foundation of China 20020093008.
the probability for $T_3$ of left buffer should be $3(1/13)=3/13$.

Distributed in $T_3$ and $T_7$ and there are 3 FBTs in tile $T_3$. Thus different buffers. As in Fig.4(a), the left buffer have 13 BSs

The route $R$ is feasible if

- Manhattan distance between source and sink measured by
- $SFRT$ set. As in Fig.1(a), $SFRT_T=[T_3,T_7]$, $SFRT_T=[T_6,T_9,T_{11}]$. There are only 4 feasible routes with buffer inserted among the 10 routes between source and sink.

**Lemma 1:** Suppose that there are $n$ routes $R_1, R_2,...,R_n$ pass the $[T_1,T_2...T_k]$ from the source to the sink, the congestion estimations of these edges are equal:

$V_{R_1} = V_{R_2} = ... = V_{R_n} = \frac{1}{n} \cdot (V_{T_1} * V_{T_2}...* V_{T_k})$.

We can figure the probabilities for each route as:

- $V_{R_1} = \frac{1}{2} \cdot (\frac{10}{13}) = \frac{50}{299}$; $V_{R_2} = \frac{1}{2} \cdot (\frac{10}{13}) = \frac{50}{299}$;
- $V_{R_3} = (\frac{10}{13}) = \frac{40}{299}$; $V_{R_4} = (\frac{10}{13}) + (\frac{3}{13}) = \frac{39}{299}$;

**C. Congestion Matrix**

Since we have known the routing probability for each route, the congestion matrix of the packing can be figured out accordingly. Each route may contribute the congestion with its probability to the tiles it passes. Suppose that the tile $T_i$ has $n$ routes $[R_1,R_2,...,R_n]$, passing, the congestion value for this net should be

$$\text{Congestion}^{Ni}(T_i) = \frac{\sum_{i=1}^{n} V_{R_i}}{V_{all}}$$

where $V_{all}$ is the probabilities sum of all FRTs.

The congestion matrix is shown in Fig.1(d). Take the tile $T_7$ as an instance, $T_7$ has three routes passing which is $R_1$, $R_2$ and $R_3$. Congestion$^{Ni}(T_7) = (VR_1 + VR_2 + VR_3) / V_{all} = 140/179$.

By scanning all the nets, the congestion estimation for each tile can be figured out.

$$\text{Congestion}(T_i) = \sum_{i=1}^{n} \text{Congestion}^{Ni}(T_i)$$

Correspondingly, the buffer insertion sites have the same congestion estimation value as the tile in which the buffer insertion site located.

IV. Congestion-Driven Buffer Allocation

To improve the performance of the packing, the buffer insertion should avoid the congested region. All the congestion estimation is based on the single net. After the congestion estimation, the congestion information for whole packing can be obtained. Therefore, if the buffers are allocated along route $R$, the congestion should be evaluated by the tiles passed by route $R$. To do the buffer allocation, we assign the routes with the congestion estimation results.

Suppose that the route $R$ passes $[T_1,T_2,...,T_k]$ from the source to the sink in the packing. We assign the congestion to route $R$ in net $N$ as:

$$\text{Congestion}(R) = \sum_{i=n}^{1} \text{Congestion}(T_i)$$

If several routes $[R_1,R_2,...,R_n]$ have the same passing tiles from source to sink in the distribution graph, we combine these routes into a dummy route $R'$:

$$\text{Congestion}(R') = \sum_{i=1}^{n} \text{Congestion}(R_i)$$

Hence, we devise a buffer allocation method which performs the buffer allocation net by net according to the congestion estimation and the buffer resources.

**Algorithm 1** buffer allocation

For net $N$ is from $N_1$ to $N_k$:

- Sort the feasible routes ascendingly by congestion($R_i$);
- $i = 1$;
While the buffer is not inserted properly and i<n:
If the feasible buffer insertion sites (BS) are available in the feasible routing tiles passed by R:
Insert the buffers at the BSs;
EndIf;
EndWhile;
EndFor

V. Experiments and Conclusion

We have implemented the placement algorithm in C programming language, and all experiments are performed on a SUN SPARC v880 workstation. The parameters used in our experiments are based on a 0.18um technology in [10]. Some MCNC benchmarks are used in the experiments. Because of the lack of information on signal direction in the benchmark files, we choose a pin to be the source and all the others to be sinks, and then decompose a multiple terminal net into a set of two-pin nets. Since the MCNC benchmarks do not come with any timing information, we generate a floorplan by running the CBL floorplanner\(^6\) randomly. Based on this floorplan, we assign target delays to the two-pin nets as follows: for each net, we first compute its best delay by optimal buffer insertion Topt, and assign its target delay as 1.1Topt. Notice that the sizes of the blocks are enlarged for demonstration of the effect of buffer planning.

To speed up the optimization process, the simulated annealing process is divided into two phases: timing optimization phase and buffer insertion phase. In the timing optimization phase, we try to search for an optimal floorplanning that the timing constraints can be satisfied as much as possible.

In the beginning of floorplanning process, the buffer planning is less meaningful because the locations of the blocks are still far from their final position. The cost function used in the phase is shown below:

\[ \text{Cost} = \text{Area} + p \times \text{Wire} + q \times \text{Tviolation} \]

Where Area is the area of the floorplan and Wire is the total wirelength (p is the weight), Tviolation is the number of the net whose optimal timing with buffer inserted is larger than the given timing constraint.

In the second phase, we start the phase of buffer insertion and the cost function used in the buffer insertion optimization phase is shown below:

\[ \text{Cost} = \text{Area} + p \times \text{Wire} + q \times \text{Tviolation} + r \times B_{\text{not inserted}} + m \times \text{Congestion} \]

Here B_{\text{not inserted}} is the number of the buffers not inserted successfully because of the limitation of the dead spaces. Congestion is the average of 5% largest congestion estimations in the packing.

In table 1, we report the experimental results of two floorplanners: a floorplanner F1 described in Ref.[11], and a congestion-driven floorplanner F2 on 2-phase simulated annealing with buffer planning. To compare the results between F1 and F2: 1)#Meet: the number of nets for which the delay constraint is met with successful buffer insertion; 2) #Inserted B / #B: the ratio of the total number of buffers inserted successfully and the total number of the buffer needed to meet timing constraints, 3) #Violation: the number of the nets whose Topt>Ttgt.

Comparing F1 and F2 in table 1, the differences between F1 and F2 on area and wirelength are very small but our algorithm (F2) can achieve better timing performance than the plain buffer insertion planning algorithm(F1). Especially the congestion results in our approach are much better than the results in F1.

As the fundamental method of buffer planning, our method can be extended to optimize the buffer insertion in many different ways, such as in the buffer site methodology that the buffer insertion sites are pre-placed in the blocks. And our algorithm can be extended to handle similar problem such as noise aware floorplanning and routability-driven floorplanning.

References

[12] Yuchun Ma, Xianlong Hong, Sheqin Dong, Song Chen etc “Dynamic Global Buffer Planning Optimization Based on Detail Block Locating and Congestion Analysis” ACM DAC’2003

<table>
<thead>
<tr>
<th>Area (mm²)</th>
<th>Wire (mm)</th>
<th>#Inserted B / #B</th>
<th>#Meet</th>
<th>#Violation</th>
<th>Congestion</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ami33</td>
<td>33 blocks, 123 nets, 363 2-pin nets</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F1</td>
<td>31.15</td>
<td>461.2</td>
<td>192/465</td>
<td>170</td>
<td>13.17</td>
<td>206</td>
</tr>
<tr>
<td>Ours</td>
<td>30.89</td>
<td>483.5</td>
<td>194/386</td>
<td>204</td>
<td>3.76</td>
<td>349</td>
</tr>
<tr>
<td>Ami49</td>
<td>49 blocks, 408 nets, 545 2-pin nets</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F1</td>
<td>146.6</td>
<td>2920</td>
<td>244/568</td>
<td>341</td>
<td>151</td>
<td>329</td>
</tr>
<tr>
<td>Ours</td>
<td>172.2</td>
<td>2978</td>
<td>277/545</td>
<td>330</td>
<td>161</td>
<td>370</td>
</tr>
<tr>
<td>xerox</td>
<td>10 blocks, 203 nets, 455 2-pin nets</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F1</td>
<td>85.41</td>
<td>1327</td>
<td>214/303</td>
<td>370</td>
<td>51</td>
<td>59</td>
</tr>
<tr>
<td>Ours</td>
<td>86.18</td>
<td>1302</td>
<td>100/253</td>
<td>402</td>
<td>39</td>
<td>109</td>
</tr>
</tbody>
</table>

Table I. Comparison of algorithm[1] and our approach

EndFor