Using Chip Master Planning in Automatic ASIC Design Flow to Improve Performance and Buffer Resource Management

Ali Jahanian [a*], Morteza Saheb Zamani[b]

[a] *Department of Electrical and Computer Engineering, Shahid Beheshti University, G. C., Tehran, Iran*
[b] *IT and Computer Engineering Department, Amirkabir University of Technology, Tehran, Iran*

Received 11 October 2009; revised 2 March 2010; accepted 13 March 2010

Abstract

Modern integrated circuits consist of millions of standard cells and routing paths. In nano-scale designs, mis-prediction is a dominant problem that may diminish the quality of physical design algorithms or even result in the disruption of the convergence of the design cycle. In this paper, a new planning methodology is presented in which a master-plan of the chip is constructed at the early levels of the physical design, preparing for the operation of the subsequent physical design stages. As a proof of concept study, the proposed planning design flow is applied to both wire planning and buffer resource planning, and the outcomes are compared against conventional contributions. Experimental results reveal considerable improvements in terms of performance, timing yield and buffer usage.

Keywords: Routability, Planning, Placement.

1. Introduction

As technology continues to scale down, the impact of interconnects on such design characteristics as performance, integrity and reliability is increased. Moreover, the correct estimation of the final interconnect delay at the higher levels of the design flow becomes more and more difficult. In this situation, the predictability of the design flow is degraded and consequently the quality of the optimization algorithms, which are in turn dependent on such predictions, may decrease, probably causing even the design cycle to diverge.

Some methodologies have been proposed to plan the interconnects as well as the buffers at early stages of the physical design. Chen et al. [1] have presented a design flow in which some low level parameters of the interconnects, such as the width and the place of the wires and their routing order, are planned at the floor-planning level. Cong et al. [2] have proposed an interconnect-centric design flow consisting of interconnect planning, synthesis, and layout stages, essentially promoting the interconnect design and optimization at early stages of the design process. However, in this approach, the width of the wires, their layers and the routing order are fixed at the floor-planning level and are based on high level estimation. Lu et al. [3] have developed a planning algorithm that takes into account the retiming considerations in an integrated physical synthesis process. In Lu's algorithm, the status of the global routing is estimated, and the final location of the buffers and the flip-flops are predicted at the floor-planning level. In [4], Cong et al. have proposed a method in which the critical wires are estimated at the floor-planning level, and the required buffers are planned along the critical wires to improve their performance. In [5], Sham et al. have presented a routability-driven floor-planning algorithm, with the congestion and buffer requirements estimated probabilistically at the floor-planning level. Jiang et al. [6] have also presented a floor-planning algorithm with buffer block planning. In particular, a simulated annealing algorithm has been proposed to find the best routing tree with respect to the buffer blocks. In each iteration of the proposed algorithm, a routing tree is constructed, buffers are allocated, and then a Lagrangian relaxation is invoked to improve both delay and area. In [7], Ma et al. have come up with a design flow for early buffer planning at the floor-planning level. In this flow, an initial solution is suggested by a simulated annealing process which is iteratively refined during floor-planning. In this method, the design

* Corresponding Author. E-mail: jahanian@sbu.ac.in

congestion is considered as a constraint when it comes to the buffer budget analysis in the process of buffer planning. These methodologies are generally associated with three drawbacks. The first and foremost is mis-prediction. In particular, all the aforementioned methodologies are comprised of two distinct phases. In the first phase, the characteristics of the wires at early stages of the physical design are estimated, and in the second phase, the interconnect parameters such as the location and the size of the buffers as well as the location, width, and the routing order of the wires will be fixed based on the initial estimations. In effect, planning in this fashion entails fixing the detailed features of the interconnects at higher levels of the physical design. However, the estimation of the final parameters at higher levels (such as floor-planning) is very rough and should not, thus, be used to fix the final values of the design parameters. In general, fixing the detailed features of the interconnects based on high-level estimations at early stages can be very erroneous [8]. The second limitation typical of the above-mentioned methodologies is that they only plan for those interconnects that connect the floor-planning macros together, neglecting, in fact, all the internal wires of the macros. This may reduce the number of the planned nets and degrade the accuracy of the estimations and planning [8]. The last drawback is that these schemes are likely to give rise to some congested areas in the channels between the macros [8], which can be attributed to their insertion of a lot of buffers in the spaces between the macros.

Hill and Kahng in [9] have discussed in detail the issue of predictability and design convergence in nano-scale regime, and have come up with the following practical ways to achieve predictability:

- *Statistical methodology:* It avoids calculations that depend on the specific topology of the design under development. However, statistical methods cannot deliver the requisite accuracy.
- *Constructive or iterative prediction:* It estimates the low level parameters by doing a quick trial placement or routing. The main drawback of this approach is that the design features in each cycle of the design process may be very different from the design in previous iterations and therefore, the design flow may fail to converge.
- *Assume and enforce:* It involves making a specific assertion about a physical property, and then constructing the circuit so as to guarantee that this property will remain valid throughout the design process.

In this paper, a new hierarchical planning methodology is presented based on the "assume and enforce" philosophy. The semantic of the proposed methodology is metaphorically very similar to the design of megacities using master-plan. In the master-plan-based urban design, the global view of the city is planned in a master-plan without fixing the details. The detailed characteristics of the city are, then, gradually configured during the construction process with respect to the master-plan [10]. This approach enhances the accuracy of the predictions at the higher levels of the physical design flow which in turn improves the quality of tools. While the pure "assume and enforce" methodology can improve the prediction accuracy at the cost of imposing rigid constraints, our gradual planning method attempts to strike balance between the prediction accuracy and the overhead of the rigid constraints.

The rest of the paper is organized as follows. In Section 2, we describe the basic concepts associated with the master-plan-based chip design methodology. In Sections 3 and 4, the application of the proposed methodology to both buffer and wire planning is evaluated. Experimental results are presented in Section 4. We conclude the paper in Section 5.

2. Chip Master Planning

2.1. Urban Design and Chip Design

Basically, planning consists of making choices among the options conceivable for the future, and then securing the implementation, by allocating necessary resources. The whole planning exercise is presented in the form of a document known as the master-plan which has proved to be one of the important tools to guide and manage the future growth of the cities in a planned manner [11]. Master-plans have been used in the process of designing many cities in various countries. It is a long-term plan usually prepared with the purpose of guiding the future growth of a city for the next 20 years and is mainly comprised of a report, land use maps, and the program of action. Conceptually, a master-plan is based on the study of the existing situation of each and every component of a city comprising the land use, the socio-economic and other facilities' surveys. Based on the analysis of the existing situation, the forecasting of the future trends is performed, followed finally by making proposals for the growth and management of the city.

One may draw an analogy between the problems in the context of chip design and those from the area of urban design. For example, the routing congestion problem in chips is metaphorically similar to the case with the transportation traffic in the congested streets and avenues of large cities.

Likewise, dropping the pressure in water or gas distribution networks of the cities is comparable to IR-drop in the power distribution rails of the chips. Chip designers try to place the connected cells close to each other so as to decrease both the total wire length and routing congestion; analogously, the city designers try to plan the high traffic sites closer to each other to alleviate the delay of the city travels.

The main drive behind urban master planning is to improve the predictability of the city evolution. Once the master-plan is devised, all the construction processes of the city are carried out accordingly. In other terms, a master-plan can be thought of as a constraint framework which somehow defines the boundaries for the future of the city and makes it more predictable. City master planning is a

gradual process in which the requirements of the city are determined and the boundaries of the city parameters are refined in a progressive manner. We use the existing analogy between the chips and the cities to borrow ideas from the urban master planning methodology - used successfully for designing cities more than half a decade - for application to the chip design, primarily to improve the quality of physical design engines. It is, however, worth noting that chips and cities also differ in a number of aspects. For starters, current chips do not grow as large cities tend to. The second distinction to be made is that chips normally have more routing resources than cities do. In the following sub-section, the proposed process for chip master planning is described in detail.

2.2. The Proposed Chip Master Planning Methodology

In this section, we present our chip planning methodology. In general, we envisage a hierarchical process in which the features of the chip are planned and gradually refined in the early stages of the design flow (e.g. floor-planning and global placement) and the rest of the design stages have to be performed in compliance with the planned parameters. The planned information is referred to as the chip master-plan. Unlike the conventional chip planning approaches, the chip characteristics (e.g., the location and the width of the wires, buffer distribution, etc.) will not be fixed all at once at the early levels of the design flow; in effect, the parameters of the chip are refined under a hierarchical process and with reference to a gradually improving chip master-plan.

With the proposed methodology, the risk of incorrect estimations is greatly reduced at the higher levels of the physical design. In particular, since at the higher levels, estimations are very rough, we only plan for the boundaries of the chip parameters. These parameters are, then, gradually refined at the lower levels, drawing the precise map of the chip where the critical paths and other metrics can be calculated (or at least estimated) more accurately.

We apply our chip master planning to the current floor-placement flow for evaluation. At any given floor-placement level *i*, the planning information of level *i* as well as the floor-placement status of level *i*-1 is utilized to refine the i^{th} level. As shown in Figure 1, at the beginning of level *i*, a conventional floor-placement is carried out. Estimations of level *i* are made using the information obtained from both the floor-placement and the highway planning of level *i*-1, simultaneously. Then, the highways of level *i* (current level) are planned based on the estimations and the floor-placement of level *i*. Finally, the floor-placement of level *i* is refined more to meet the constraints of the planned highways. The planning and the floor-placement go on concurrently in a hierarchical fashion, developing an effective cooperation for both planning the chip parameters and performing the placement.

The proposed methodology should be carried out via a multi-level hierarchical process. Therefore, we require a multi-level floor-placer in which the cell locations are determined gradually within a multi-level process. We have used the CapoPlacer [12] framework to implement and evaluate our chip master planning methodology.

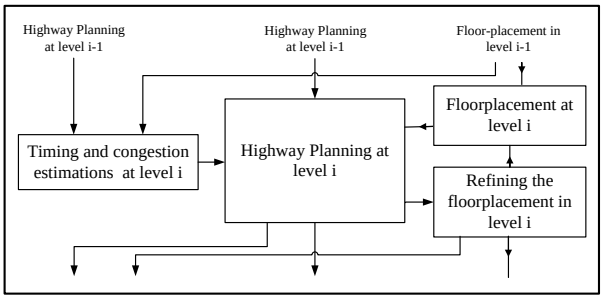

Fig. 1. The design flow of master-plan-based floor-placement.

We have already evaluated buffer planning in [13] and highway planning in [14] but the idea of the chip master plan and its relation to the buffer and the wire plan have not been evaluated before. In other words, master buffer planning and master wire planning can be viewed as two case studies in the context of chip master planning first introduced in this paper. In particular, the main contribution of this paper lies in the generalization of the chip master-plan within the regular ASIC design flow.

2.3. Chip Master-plan Contents

In hierarchical chip master planning, the plans must be interchanged between the levels of the design stages given that the lower levels need to be aware of the assumptions and plans made at the previous levels of the design cycle. Current physical design flows do not have planning data banks for interchange between the levels of the flow. Therefore, new information must be added to the physical design data bank to convey the plan information in addition to the design files. We refer to this data bank as *Chip Master-plan*.

Basically, the chip master-plan at each level of the hierarchical planning process is a data bank of all plans made at that level based on the assumptions and estimations specific to that level. It should, however, be noted that the plans for each level is constructed based on the planning of the previous levels together with the estimations made at that level. The chip master-plan can also be considered as an abstract data over the design process. In other words, the design is analogous to the detailed map of a city with the master-plan portraying the abstract and global view of the city.

In current design flows, only the design information is transferred between the design levels, however, in the master planning design flow, both the design and the master-plan are interchanged to convey both design and plans together from the primary levels of the physical design to the lower levels.

2.4. Evaluation Strategy

We have implemented the proposed design flow in the form of two different planning activities: buffer resource planning and wire planning. These two techniques have been widely used in recent years for the improvement of performance and signal integrity, and are particularly apt for the evaluation of master planning. Our proposed methodology can be utilized for addressing any planning problem in the context of chip design. However, in this paper, we have settled on wire and buffer planning just as two planning case studies to evaluate the proposed idea.

Since master planning is performed gradually, a hierarchical floor-placement environment would serve as a proper backdrop to evaluate the proposed flow. CapoPlacer [12], a well-known bi-sectioning-based floor-placement tool that is widely used in academic research, has been selected in this article as the floor-placement framework. We justify our choice through the following reasons:

- CapoPlacer is a renowned placer for utilizing one of the top five placement algorithms in the ISPD placement contest [12].
- The hierarchical and iterative behavior of CapoPlacer provides a suitable framework within which the distinctive characteristics of the highways can be gradually refined throughout the design flow process so that the full details of the highways can be obtained at the end of the floor-placement.
- The bi-sectioning-based partitioning, used in CapoPlacer, simplifies the white space distribution during the floor-placement.

In the subsequent two sections, we will elaborate on the implementation of chip master planning for buffer and wire planning, respectively.

3. Wire Master-planning

In this section, the proposed master planning methodology is applied to the problem of interconnect planning in order to improve the performance and the timing yield of the design. In the proposed technique, some regions of the design are selected to have auxiliary routing resources. Then, critical and near-critical nets are assigned to these wealthy regions in a hierarchical flow to be routed with less delay using the resources available in these regions. These wealthy regions are referred to as *Highways on Chip*. The location and the amount of the resources in the highways are gradually determined during the hierarchical process. A more detailed description of the specifics of this method can be found in [13].

The technique we are discussing here is very similar to that used for highway design in large cities. Highways are established in the congested paths of the large cities primarily to reduce the transportation delay and the traffic volume. In the master-plan-based urban design process, the global characteristics of the highways are planned into the master-plan (without positions or widths), followed by their gradual positioning and configuration over the course of the urban design and according to the urban master-plan [10]. Step by step bounding of the highways will decrease the risk of incorrect estimations within the urban design process. This methodology is known as Highway Master Planning (HAMPA). In HAMPA, the highway map is basically a subset of the wire master-plan and highway planning is tantamount to the *wire master planning*.

3.2. Basics of Highway Planning

We draw on a neighborhood graph in each global placement stage to obtain a graph representation for highways in that stage. In a neighborhood graph, each vertex represents a bin, and there is an edge between each pair of neighboring vertices [15]. Associated with each edge in the neighborhood graph is a weight, characterizing the amount of resources in between the two corresponding adjacent bins. We refer to this graph as the *highway graph*. The dotted lines in Figure 2 represent those edges with zero or very low planned resources.

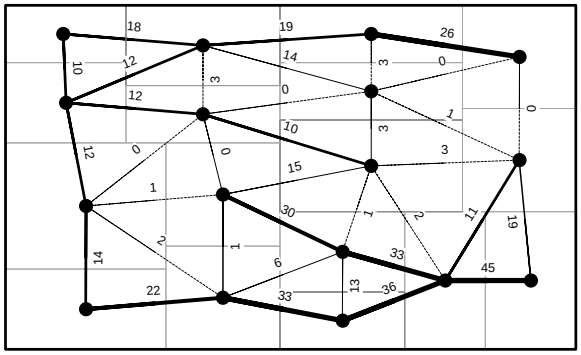

Fig. 2. Highway graph at an intermediate level of the global placement.

Let $G_i(V_i,E_i)$ be the highway graph at the i^{th} level of the placement hierarchy. V_i and E_i are the set of vertices and edges at the i^{th} level, respectively. The weights on E_i can be adjusted by assigning the near-critical nets to the highway graph. Therefore, the problem of highway planning is equivalent to finding the best weights on the edges in E_i such that:

- The delays associated with the near-critical paths are minimized.
- Routing congestion is controlled.
- The availability constraints on highway resources are met.

3.2. The HAMPA Methodology

We have used CapoPlacer [12] as the hierarchical floor-placement framework to implement the HAMPA design flow. Highway planning is performed in concert with the CapoPlacer's global placement process. At the primary

levels of the global placement, the estimation error can be very high. Therefore, highway planning should not start from the first levels of the global placement hierarchy. We have divided the CapoPlacer's global placement flow into two steps: *global placement without planning* and *global placement with planning*. The very first hierarchy level of the global placement at which the planning of highways is started is referred to as the starting planning level (SPL). More specifically, SPL is defined as the first level at which the average size of the bins is smaller than a given threshold (D_{buff}). This value is, in fact, the minimum effective distance between two successive buffers [4]. Following SPL, the stage of global placement without buffer insertion is finished, preparing for the commencement of the global placement with buffer insertion.

In global placement with planning, the highway planning is performed concurrently with the CapoPlacer's floor-placement flow. Each level of HAMPA consists of three main stages:

- *CapoPlacer bi-partitioning:* In this stage, the main operations of CapoPlacer's floor-placement (e.g., bi-partitioning and creation of new bins) are performed. This operation is implemented by CapoPlacer without any modifications as described in [12].
- *Highway planning:* Highway planning is a post-bi-partitioning process. In this stage, the critical and near-critical nets are identified and assigned to the highways. Highway resources are subsequently updated according to the properties of the near-critical nets and to the congestion map of the design.
- *Placement Refinement:* In the last stage, the placement is updated to re-arrange the white spaces in the design (reserved for buffers) according to the planned highways.

More specific details concerning the technicalities of these three stages are described in the following sub-sections.

3.2.1. CapoPlacer bi-partitioning

In this stage, each bin of the design is partitioned (vertically or horizontally) into two bins. CapoPlacer uses either hMetis or MLPart for partitioning [12]. The main metric in partitioning is the cut size but some other criteria such as congestion can also be taken into account.

3.2.2. Highway Planning

In this stage, highways are planned and the weights of the edges in the highway graph are updated. Our envisaged flow for highway planning is shown in Figure 3.

Step 1:	Construct planning structures.
Step 2:	Apply the previous highway weights to the current highway edges.
Step 3:	Static timing analysis by considering previous highways.
Step 4:	Update the highways.

Fig. 3. The proposed design flow for highway planning.

Step 1: In this phase of the algorithm, the planning structures such as bins, the neighbourhood graph and the highway graph are constructed at each level of HAMPA.

Step 2: After CapoPlacer's partitioning, some new bins are created, and therefore, the current highway graph contains more nodes than the previous level graph. In order to reflect the previous plans on the current level of the hierarchy, the edge weights associated with the highway graph at the previous level must be used to adjust the weights in the current highway graph. Further details concerning the mechanics of this algorithm are excluded due to page limitations.

Step 3: In this step, both critical and non-critical net segments are estimated by a static timing analyzer (STA). In this situation, the near-critical net segments are the members of a path whose delay is close to the most critical path delay by a factor α (we have used values 0.85 ~ 0.95 for α). We have improved the conventional static timing analyzer to take the information of the previous highways into account when it comes to the timing calculation.

Step 4: In this step, the weights associated with the highways are updated based on the criticality of the nets and on the congestion levels of the design. The highway planning algorithm is shown in detail in Figure 4. The algorithm operates on each critical net segment[*]. If the net segment has not been planned so far, the shortest path between its corresponding source and sink will be found on the highway graph by Dijkstra's algorithm. This net segment is then assigned to the corresponding edges of the highway graph in accordance with the resultant shortest path. Finally, the algorithm checks if the net segment delay can be improved using the highway resources, in which case, the optimal number of buffers (corresponding to the feasible region for each buffer) and the optimal size of the wires are estimated using a method similar to that discussed in [4], and get assigned to the net.

```
1:  FOR each net segment sik of the net ei
2:      IF (Sik is near-critical AND has not been planned at any of
        the previous levels)
3:          FIND the shortest path on the highway mesh for Sik.
4:          Assign Sik to the edges of the shortest path on the
            highway mesh.
5:          Compute the required resources for the efficient routing
            of Sik.
6:          IF (delay of Sik can be improved using the highway
            resources)
7:              Assign the resources to Sik.
8:              Add in the weights of the edges of the shortest path
                on the highway mesh.
9:          END
10:     END
11: END
```

Fig. 4. The proposed algorithm for highway refinement.

[*] A multi-terminal net may have many net segments derived from the Steiner tree of the net.

3.2.3. Placement Refinement

After planning the highways, the current placement must be refined to meet the availability constraints on the resources required for applying the highway planning. In this phase, the current placement is updated based on the current highways on the chip. The required resources for planning correspond to the routing area and to the proper areas required for buffers to improve the net segment delays. Both of these parameters can be controlled by a proper white space distribution. We have developed an algorithm for white space redistribution which is based on highway requirements and works by shifting the boundary of the bins appropriately.

3.3. Experimental Results for Wire Master Planning

We have implemented our algorithm in C++ on an Intel Dual Core 1.8GHz machine with 2 GB of main memory. Ten circuits have been selected randomly from the IWLS-2005 benchmark set [16]. These circuits come in various size ranges from about 17000 to 92,000 standard cells. The benchmarks have been synthesized in 130nm technology with eight layers of metal. All benchmarks have been floor-planned and placed by the CapoPlacer with 90% of row utilization. Table 1 shows the characteristics of the attempted benchmarks.

In order to evaluate the presented design methodology, its corresponding results have been compared with the outputs of CapoPlacer after detailed placement. We refer to our design flow as HAMPA and to the CapoPlacer's flow as CAPO.

3.3.1. Performance Improvement

Table 2 shows the performance gain of HAMPA's design flow as compared with CAPO's. It should be noted that HAMPA's final results are not immediately comparable to CapoPlacer's, since HAMPA uses buffer insertion but CapoPlacer does not. Therefore, In order to draw a comparison, we have to report on CapoPlacer's results with buffer insertion. We have used *Magma BlastFusion* [17] for timing optimization of CapoPlacer's results with buffer insertion. In Table 2, the columns labelled *CAPO* and *C&B* correspond to the minimum clock period of the design after CapoPlacer and to the clock period after CapoPlacer followed by Magma buffer insertion, respectively. The Column labelled *HAMPA*, on the other hand, represents the minimum clock period of the design resultant from HAMPA's design flow. The column labelled as "*HAMPA vs. CAPO"* represents the performance gain of HAMPA's design flow as compared with CapoPlacer's (without buffer insertion), and finally, the column labelled as "*HAMPA vs. C&B"* shows the performance gain of HAMPA's design flow as compared with CapoPlacer's (followed by Magma buffer insertion).

Table 1

The characteristics of the benchmarks.

Index	Circuit	#cells
1	pci_bridge32	16816
2	dma	19118
3	b20_1	19131
4	b22	28317
5	b22_1	28128
6	wb_conmax	29034
7	b17	37117
8	b17_1	37383
9	ethernet	46771
10	b18	92048

Table 2

Experimental results for HAMPA and CapoPlacer (design performance)

	Design Clock period (ps)			Performance Improvement (%)	
Benchmark	CAPO	C&B	HAMPA	HAMPA vs. CAPO (%)	HAMPA vs. C&B (%)
pci_bridge32	5827.34	5796.63	5791.98	0.60	0.08
dma	4651.59	4423.86	3804	18.22	14.01
b20_1	12003.23	11263.45	8959.02	25.36	20.45
b22	12632.73	11832.35	10282.61	18.60	13.09
b22_1	8047.78	7693.11	6666.62	17.16	13.34
wb_conmax	8732.6	8651.02	7593	13.04	12.23
b17	9623.3	9314.32	8259.73	14.16	11.32
b17_1	9703.12	9503.12	8475.67	12.65	10.81
Ethernet	6019.98	5767.78	4707.74	21.79	18.37
b18	11326.49	10543.12	8123.88	28.27	22.94
Average				17.00%	13.66%

As can be seen in Table 2, the design clock period after HAMPA is, on average, 17% and 13.66% better than those after CAPO and C&B, respectively. It is worthwhile to note that the witnessed improvements are even more noticeable for larger circuits featuring a larger of number of hierarchy levels, in which case HAMPA has a better opportunity to improve the performance of the design. Also, in large circuits, the ratio of the final bin area to the design area is smaller, and therefore, the net delay estimations are more accurate in HAMPA's design flow.

In this section, we have reported on the performance improvements gained at the last level of the global placement. It was shown that the design performance can be improved using a progressive and hierarchical approach. An additional advantage to this progressive process is that at each level of HAMPA's design flow, the delays of the near-critical paths are decreased, improving the total average. This can lead to more robustness against the delay variations, and also improves the timing yield of the design.

3.3.2. Timing Yield Improvement

The timing yield (Y) can be described by a cumulative distribution function (CDF) associated with the delay of the near-critical paths [18]. Decreasing the average delay of the near-critical nets also decreases the timing yield of the design. As pointed out earlier, in HAMPA's design flow, not only the delay of the most critical path but also the average delay of all near-critical nets is decreased. Therefore, in general, HAMPA's design flow can improve the timing yield of the design. We have come up with the following experiment to measure the improvement in timing yield. Assuming that the delay associated with the critical path has 30% variation (this is an acceptable amount of variation as reported in [18]), we define $DME_{critical}$ as the mathematical expectation of the delay of the near-critical nets. $DME_{critical}$ is calculated as:

$$DME_{critical} = \frac{1}{n}(\sum_{i=1}^{n} T_{critical}(i)) + 0.15 * T_{target} \quad (1)$$

where $T_{critical}(i)$ is the delay of the i^{th} near-critical path, and T_{target} denotes the target clock period of the design. Table 3 shows the results obtained for the yield improvement. The columns labeled as *Capo* $DME_{critical}$ and *HAMPA* $DME_{critical}$ show the mathematical expectation of the delay of the near-critical paths associated with the design placed by CapoPlacer and HAMPA, respectively. Also, the last column shows the percentage of the yield improvement.

Table 3
Experimental results for HAMPA and CapoPlacer (timing yield improvement)

Circuit	Capo $DME_{critical}$	HAMPA $DME_{critical}$	Yield Improvement (%)
pci_bridge32	7575.542	6892.80	9.012
dma	6047.067	5500.90	9.03
b20_1	15604.2	14703.88	5.76
b22	16422.55	15059.28	8.30
b22_1	10462.11	9557.33	8.64
wb_conmax	11352.38	10379.12	8.57
b17	12510.29	11447.96	8.49
b17_1	12614.06	10343.74	17.99
Ethernet	7825.974	7023.97	10.24
b18	14724.44	12645.02	14.12
Average			10.02%

As can be seen in the table, under the assumption of 30% delay variation, the timing yield is improved by 10.02% on average. The resultant improvement will be more considerable in nanometer scales where the variation rate is even higher.

4. Buffer Master Planning

In this section, a new buffer planning methodology is presented which leverages the idea of chip master planning. In this methodology, a master-plan for buffers is generated at the primary levels of the hierarchical floor-placement, requiring that the succeeding stages be performed with reference to this master-plan. As in general hierarchical placers, placement is carried out in two steps: global placement and detailed placement. In the global placement stage, the statistical distribution of the required buffers, namely *Buffer Master Plan (BMP)*, is estimated. BMP is then used to guide the detailed placement to distribute white spaces into the proper regions of the chip. During the estimation process for BMP generation, the routing congestion of the regions is taken into account to plan the buffers in non-congested regions in order to avoid the routability problem. The global placement phase is stopped at a pre-computed stage, referred to as the *planning level*, the buffer requirement for each region is estimated, and BMP is generated based on the critical nets distribution and on the routing congestion in various regions of the design. A more detailed description of the specifics of this method can be found in [14].

The CapoPlacer's global placement algorithm has been modified so that the white spaces of the design are distributed according to the BMP. CapoPlacer's detailed placement has also been changed to execute with respect to the BMP such that the distribution of white spaces can be reserved in compliance with the BMP generation. The resultant flow is summarized in Figure 5 and is explained in greater detail throughout the following sub-sections.

1: Compute the planning level.
2: Start the global placer with CapoPlacer down to the planning level.
3: Generate the BMP
3-1: Construct the congestion map.
3-2: Perform the probabilistic static timing analysis (PSTA).
3-3: Extract the BMP according to the current global placement, PSTA and congestion of the design.
4: Change the distribution of the white spaces with respect to the BMP.

Fig. 5. The proposed algorithm for buffer planning.

4.1. Buffer Master-plan

BMP is basically a statistical estimation of the buffer requirements along the nets in various regions of the design. To generate the BMP, the number of buffers is predicted for each net; then, a feasible region along with the mathematical expectation of the number of buffers in that feasible region is estimated. In Figure 6, each polygon represents an estimated buffer region corresponding to the buffer expectation (the buffer expectation is the mathematical expectation of the number of buffers). As can be seen, the intersecting areas stand a greater chance of holding the buffers.

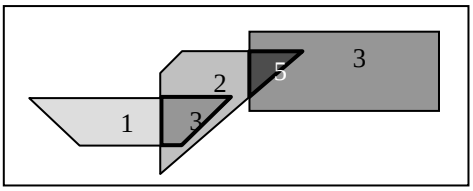

Fig. 6. The estimated buffer regions and expectation of buffers.

Afterwards, the buffer expectations in various regions of the design are mapped onto a grid to form a buffer requirement map. The main consideration in BMP generation is to produce an accurate and robust estimation of the required buffers in the floor–planning stage.

4.2. BMP Estimation Algorithm

At each level of the hierarchical floor-placement, the cells in each cluster are assumed to be placed at the center of that cluster. We use the CapoPlacer tool that uses a recursive bi-partitioning process for floor-placement. In order to produce the BMP, CapoPlacer is stopped at a specific level that is called the *planning level*. The planning level is selected as the first level of the floor-placement at which the average size of the bins is smaller than a certain threshold (referred to as D_{buff} in Section 3.2). After stopping the floor-placer, the BMP is estimated based on the information of the global placement stage. Our algorithm is based on three assumptions:

- The probability of the buffer requirements along the global nets is higher than that of the local nets.
- The probability of the buffer requirements is higher along the critical nets.
- New buffers are inserted into the less-congested areas.

Basically, assumptions 1 and 2 improve the performance and help achieve timing closure, while assumption 3 is necessary to improve the routability, the signal integrity and the manufacturability of the chip.

Given that the cell locations are unknown at the planning level, a Probabilistic Statistical Timing Analysis (PSTA) is required to be conducted to extract the critical paths. In PSTA, the specified design timing constraint (timing wall) has a key role. If the timing constraint is too tight, the statistical number of the critical nets will be increased and, thus, the impact of the critical nets on the BMP becomes greater. Since the buffer requirement estimation algorithms are different for the global and the local nets, once PSTA is conducted, the nets are accordingly divided into global and local nets. More specific details concerning the technicalities of these algorithms are described in the following sub-sections.

4.2.1. Buffer Requirement for Global Nets

Global nets route between clusters whereas local nets are intra-cluster. Let the width and the height of the layout be L_x and L_y, respectively. A uniform grid on the layout with grid cells of size λ_x and λ_y is considered accordingly.

Now a t-terminal net is assumed k of which are placed in clusters $c_1, c_2, \ldots, c_n$. After detailed placement, the i^{th} terminal of the net k will be placed in one of the n_i^k grid cells where n_i^k can be computed as follows:

$$n_i^k = \left\lfloor \frac{A_i}{\lambda_x . \lambda_y} \right\rfloor \tag{2}$$

In this equation, A_i is the area of the cluster in which the i^{th} terminal of the net k is placed. Therefore, the number of all possible routing topologies for a multi-terminal net k with t terminals is computed as: $\prod_{i=1}^{t}(n_i^k)$. Thus, the probability of the i^{th} topology for net k is:

$$P_i^k = \frac{1}{\prod_{j=1}^{t}(n_j^k)} \tag{3}$$

In order to estimate the number of buffers and their locations, the routing tree associated with each net must be predicted. In recent years, various methods have been proposed to estimate the routing details of a net at the floor-planning stage. In this paper, we draw on the RSTST method [19] that uses an improved Single Trunk Steiner Tree (STST) algorithm for high-level interconnect estimation. In [19], it has been shown that if all t-terminals of a net are uniformly distributed in a unit square, the expected wire length for RSTST will be $O(\sqrt{t})$. This estimation is used to find the upper bound of the required number of buffers. Then, the number of the required buffers and the feasible region of each buffer are computed. In order to estimate each buffer's FR for a multi-terminal net, we rely on the bounding box of each sink and source terminal.

As stated before, in order to achieve the desired performance, the probability of buffer insertion on critical global nets should be higher than that on non-critical nets. To calculate the criticality of a net, two parameters are considered for the critical path. The first parameter, namely D_k, is the delay of the path containing the net, and is set to zero for non-critical nets. The second parameter, PN_k is the

number of the paths containing a particular net. Obviously, the nets with a higher PN_k are better candidates for buffer insertion. Equation 4 is used to calculate the criticality of net *k*.

$$C_k = \frac{2 * D_k}{D_k + D_{\max}} * PN_k \quad (4)$$

where D_{max} is the delay of the longest critical path in the design. It is of note that for the nets with $D_k = D_{max}$, the criticality is equal to PN_k and for the nets with $D_k = 0$, C_k is equal to zero. The ultimate buffer expectation for net *k* in the j^{th} region BE_j^k, is computed using Equation 5:

$$BE_j^k = \sum_{i=0}^{NT_j^k} (BN_i^j * P_i^k) \quad (5)$$

where NT_j^k is the number of topologies for net *k* in the j^{th} region and BN_i^j is the estimated number of the required buffers for the i^{th} topology of net *k* in region *j*. Based on the above equations, the total buffer requirement for all nets in the j^{th} region (TBE_j), is calculated using Equation 6:

$$TBE_j = \sum_{k=0}^{GNN_j} (C_k * BE_j^k) \quad (6)$$

where GNN_j is the number of the global nets in region *j* and C_k is the criticality of net *k.*

4.2.2. Buffer Requirement Estimation for Local Nets

Local nets route inside clusters and their final length and the probability of the buffer requirements depend on the size of the containing cluster and on the criticality of the nets.

The probability of inserting a buffer in net *k* (PB_k) is related to the delay of that net (ND_K) divided by the containing path delay (PD_K):

$$PB_k = \frac{ND_k}{PD_k} \Rightarrow PB_k = \frac{NL_k^2}{PL_k^2} \quad (7)$$

where NL_k and PL_k are the length of net *k* and its containing path, respectively. At the planning level, the length of a local net can be approximated by half the perimeter of the cluster containing that net. Thus, the buffer expectation in net *k* (BE_k) can be expressed as:

$$BE_k = PB_k * C_k = \frac{HP_k^2}{PL_k^2} * C_k \quad (8)$$

where C_k is a boolean variable which is set to 1 when net *k* belongs to at least one critical path. Buffer expectations are calculated for all local nets and are then added to those buffers which have been computed for global nets.

4.2.3. Mapping Buffer Regions to Grid

After estimating the buffer regions for all global and local nets, a list of polygons (FRs) along with their buffer expectations has been created. In order to map these polygons to the BMP gird, the intersection of each polygon with the grid cells are found and the buffer expectation of each polygon is added to the buffer expectation of the grid cells. The final buffer expectation for a grid cell *(i,j)* is calculated as follows:

$$GBE_{(i,j)} = \frac{1}{C_{(i,j)}} * \sum_{r=0}^{NIR} (E_r * \frac{A_{(i,j)}^r}{A_g}) \quad (9)$$

where E_r is the estimated number of buffers in region *r* and $A_{(i,j)}^r$ is the intersecting area of region *r* with the grid cell *(i,j)*, A_g is the area of a grid cell, $C_{(i,j)}$ is the congestion level of the design in grid cell *(i,j)* and *NIR* (number of intersecting regions) is the number of regions that intersect with the grid cell *(i,j).*

4.2.4. White Space Re-distribution Based on the BMP

The distribution of white spaces in the current floor-plan (at the planning level) may not conform to the estimated BMP. In this stage, the white spaces of the design are redistributed in conformity to the BMP. We have changed CapoPlacer to create the floor-planning tree during the recursive bi-partitioning flow; then, by sliding the boundaries of the clusters, the white spaces are redistributed based on the BMP at the planning level. Further details regarding this process have been omitted due to page limitations.

4.3. Experimental Results for Buffer Master Planning

We have put our flow into trial on six circuits as listed in Table 4. We have defined two different buffer insertion flows for comparison. In the first flow, the circuits have been placed by CapoPlacer followed by a post placement timing optimization performed by Magma BlastFusion [17] and based on buffer insertion. We refer to this flow as the *conventional buffer insertion flow*. In the second flow, the BMP has been generated at the planning level and the detailed placement has been performed by CapoPlacer with respect to the estimated BMP; as has been the case with the first flow, the post placement timing optimization has been performed by Magma BlastFusion with buffer insertion. We refer to the second flow as the *BMP-based buffer insertion flow (BMPBI).*

Table 4 shows the results for both the conventional and the BMPBI flows. The number of the inserted buffers in the timing optimization process performed by Magma as well as the final clock period of the designs is shown in the columns labelled as *#buffs* and *Delay*, respectively. Furthermore, the power-delay product (PDP) is used to

Table 4
Experimental results for the conventional buffer insertion flow and BMPBI flow

Circuit	WSR	Conventional buffer insertion flow					BMP-based buffer insertion flow (BMPBI)					PDP Reduction	CPU Time Overhead
		#buffs	Delay (ps)	PDP	Avg. PDP	CPU time (sec)	#buffs	Delay (ps)	PDP	Avg. PDP	CPU time (sec)		
des_area	15%	27	1598	23298.84	17716.3	5.1	14	1587	11997.72	15232.92	5.3	1.4%	3%
	10%	19	1578	16190.28			13	1602	11246.04				
	3%	16	1581	13659.84			26	1600	22464				
systemaes	15%	10	2203	11896.2	13060.7	10.7	9	2230	10837.8	10597.32	11.1	2.3%	3%
	10%	14	2193	16579.08			9	2203	10706.58				
	3%	9	2203	10706.58			9	2203	10706.58				
tv80	15%	39	2364	49764.8	46312.1	11	22	2366	28108	42096.4	11.4	9.1%	3%
	10%	41	2355	52135.7			24	2365	37035.9				
	3%	29	2365	37035.9			48	2359	61145.3				
ac97_ctrl	15%	11	823	4888.6	3780.8	14	7	834	3152.5	3477.5	14.5	8.02%	3%
	10%	6	825	2673			9	819	3980				
	3%	7	FAILED	-			4	825	3300				
dma	15%	133	2667	191543.9	192498.9	26.3	127	2706	185577.4	166114.66	27	13.7%	5.7%
	10%	138	2596	193454			99	2712	144983.5				
	3%	263	FAILED	-			119	2611	167782.8				
aes_core	15%	18	1400	25200	31711.3	20.3	19	1390	26410	20496.66	21.7	35%	5.5%
	10%	19	1390	26410			22	1340	29480				
	3%	31	1404	43524			4	1400	5600				
Average												11.58%	3.86%

compare the quality of the result of buffer insertion. For each circuit, three experiments with different percentages of white spaces (*WSR*) were performed.

The average PDP is shown in the column labelled as "*Avg. PDP*". In order to compute the PDP, the specifics of the internal power consumption of the used buffers are extracted from Cadence's CRETE library. In the column "*PDP reduction*", the final reduction of PDP in the BMPBI flow has been compared to the conventional flow. Finally, the column "*CPU Time Overhead*" shows the CPU time overhead induced by BMPBI compared to the conventional flow.

As can be seen in Table 4, larger circuits exhibit higher PDP improvement (1.4% for small circuits vs. 35% for large circuit). This can be attributed to the fact that large circuits have more bins at the planning level, resulting in the generation of a more concise BMP, and thus in the more efficient distribution of buffers. In addition, large circuits feature more levels which can, in turn, result in a higher chance for managing the white spaces in the BMP-based detailed placement flow.

5. Conclusion

We have presented a new hierarchical planning methodology based on the "assume and enforce" philosophy. Experimental results for wire master planning demonstrate a superior performance and a better timing yield. As for buffer master planning, performance can be improved by fewer inserted buffers, leading also to the reduction of the power consumption overhead as well as to a more suitable buffer resource distribution. Our conducted experiments and analyses showed that the resultant improvements are even more considerable for larger and more congested circuits.

As evidenced by the outcome of the experiments, the design convergence has also been improved. Moreover, while the conventional buffer insertion methodology fails under two benchmarks, our method completes successfully. The timing yield is also improved by 10.02%, effectively enabling designers to achieve the desirable hardware in fewer iterations of the design cycle.

Our on-going research is focused on transferring the master-plan-based design to higher abstraction levels such as logic synthesis in order to employ the high level information for planning.

References

[1] H.M. Chen, H. Zhou, F.Y. Young, D.F. Wong, H.H. Yang, and N. Sherwani, Integrated floorplanning and interconnect planning, In Proc. of Int. Conf. on Computer Aided Design, pp.354-357, 1999.

[2] J. Cong and D. pan, Wire width planning for interconnect performance optimization, In IEEE Trans. on Computer Aided Design, Vol.21, No.3, pp.319-329, March 2002.

[3] R. Lu and C.K. Koh, Interconnect planning with local area constrained retiming, In Proc. of Design Automation and Test in Europe, pp.442-447, 2003.

[4] J. Cong, T. Kong, and D. Pan, Buffer block planning for interconnect planning and prediction, In IEEE Trans. on VLSI, Vol.9, No.6, pp.929-937, 2001.

[5] C.W. Sham and E.F.Y. Young, Routability driven floorplanner with buffer block planning, In Proc. of Int. Symposium on Physical Design, pp.470-480, 2002.

[6] I. H. Jiang, Y.W. Chang, J.Y Jou, and K.Y. Chao, Simultaneous floorplan and buffer-block optimization, In IEEE Trans. on Computer Aided Design, Vol.23, No.5, 2004.

[7] Y. Ma, X. Hong, S. Dong, S. Chen, C.K.Cheng, J. Gu, Buffer planning as an integral part of floorplanning with consideration of routing congestion, In IEEE Trans. on Computer Aided Design, Vol.24, No.4, 2005.

[8] C.J. Alpert, H. Jiang, S.S. Sapatnekar, and p.G. Villarrubia, A practical methodology for early buffer and wire resource allocation, In IEEE Trans. on Computer Aided Design, Vol.22, No.5, pp. 573–583, 2003.

[9] D. Hill and A.B. Kahng, RTL to GDSII-From foilware to standard practice, In IEEE Design and Test of Computers, pp.9-12, 2004.

[10] A. Madanipour, Social exclusion in european cities: processes, experiences and responses, regional development and public policy series, Routledge Pub., London, 2003.

[11] R. Hameed and O. Nadeem, Challenges of implementing urban master plans: the lahore experience, In Proc. of World Academy of Science Engineering and Technology, Vol. 17, pp. 335-342, 2006.

[12] J.A. Roy, et al, Capo: robust and scalable open-source min-cut floorplacer, In Proc. of Int. Symposium on Physical Design, pp.224-226, 2005.

[13] A. jahanian and M. Saheb Zamani, Improved timing closure by early buffer planning in floor-placement design flow, In Proc. of IEEE/ACM Great Lakes Symposium on VLSI, pp. 558-563, 2007.

[14] A. Jahanian and M. Saheb Zamani, Performance and timing yield enhancement using Highway-on-Chip Planning, In Proc. of EuroMicro Digital System Design, Italy, 2008.

[15] N. Sherwani, Algorithms for VLSI physical design automation, 3rd edition, Kluwer Academic Publishers, Boston, MA, 1999.

[16] IWLS Benchmarks , Available on http://iwls.org/iwls2005/ benchmarks.html, 2005.

[17] Magma design automation, a complete design solution for structured ASICs, Available on http://www.magma-da.com, 2005.

[18] Y. Matsumoto, M. Hioki, T. Kawanami, and T. Tsutsumi, Performance and yield enhancement of FPGAs with within-die variation using multiple configurations, In Proc. of Int. Symposium on FPGAs, pp.169-177, 2007.

[19] H. Chen, C. Qiao, F. Zhou, and C. K. Cheng, Refined single trunk tree: a rectilinear Steiner tree generator for interconnect prediction, In Proc. of System Level Interconnect Prediction Conf., pp. 85-89, 2002.