J. Parallel Distrib. Comput. 103 (2017) 3–10

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

A CPS framework based perturbation constrained buffer planning approach in VLSI design

Xiaodao Chen [a,b], Xiaohui Huang [a], Yang Xiang [a,c,*], Dongmei Zhang [a,b,*], Rajiv Ranjan [d], Chen Liao [e]

[a] China University of Geosciences, Wuhan, 430074, China
[b] Hubei Key Laboratory of Intelligent Geo-Information Processing, China University of Geosciences, Wuhan, 430070, China
[c] Deakin University, Australia
[d] Computing Science at Newcastle University, United Kingdom
[e] Marvell Semiconductor, USA

ARTICLE INFO

Article history:
Received 5 May 2016
Received in revised form
15 October 2016
Accepted 23 November 2016
Available online 20 December 2016

Keywords:
Buffer insertion
Buffer planning
Perturbation
Integer linear programming
Parallel computing

ABSTRACT

As VLSI technology advances towards nanoscale devices, interconnect delay is becoming increasingly important, and could be effectively reduced using buffer insertion. The widely-used buffer insertion technique in industry is to insert a set of buffers on the chip, which may overlap some gates, and then greedily move the buffers to the nearest available buffer holes. The moving distance of inserted buffers largely affects the wirelength which may result in the increase of the interconnect delay. This necessitates efficient algorithms to minimize the moving distance of buffers for effective buffer insertion to obtain high-performance VLSI designs.

This paper proposes an efficient, perturbation constrained buffer planning algorithm to maximize the candidate buffer holes with regarding to the feature of CPS based buffering design framework. Instead of directly moving buffers to the existing available buffer holes, the proposed algorithm changes the original placement by moving some gates tinily to provide more flexibility for buffer insertion. The integer linear programming based technique is designed for the physical design flow which allows small moving range of gates. Parallel technique is utilized to solve the ILP problems efficiently when the scale of chip is increasing. Experimental results have shown that the proposed algorithm achieves at most 41.49% increase in the available buffer holes when compared to the algorithm with no gate movement.

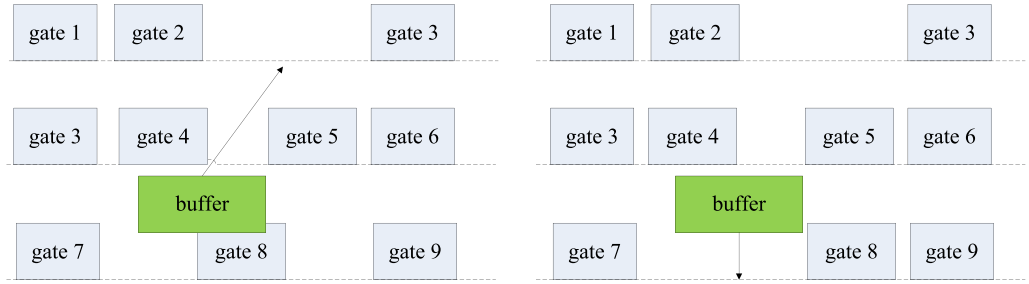
1. Introduction

As VLSI technology advances nanoscale devices, interconnect delay is becoming increasingly critical. As an effective technique to reduce interconnect delay, buffer insertion is indispensable in physical flow design [4]. The widely-used buffer insertion technique in industry is to insert a set of buffers on the chip, which may overlap some gates, and then move the buffers to the nearest available buffer holes. However, the moving distance of inserted buffers itself may increase the wirelength, which results in the increase of interconnect delay. This necessitates efficient algorithms to maximize the candidate buffer holes on chip which can minimize the moving distance of buffers for effective buffer insertion.

Buffer insertion has attracted significant research attention in the recent past. Many existing works focus on the interconnect delay by developing buffer insertion techniques. In [17], a depth first search technique based algorithm has been proposed to minimized Elmore Delay. In [2], a context-aware buffer insertion method is proposed based on the Dynamic Programming technique. In [8], a full polynomial time approximation scheme is proposed for the buffer insertion. In [11], an approach is proposed for simultaneous routing and buffer insertion. There are also different objectives oriented buffer insertion works, such as handling noise [1], floorplanning [5], buffer cost [9,10], wire sizing [3], variations [6], power consumptions [13] and 3D IC design [15]. However, most of the existing works only concentrate on exploring buffer insertion based on the given placement without considering the total moving

* Corresponding authors at: China University of Geosciences, Wuhan, 430074, China.
E-mail addresses: cxdao@yahoo.com (X. Chen), xhHuang94@iCloud.com (X. Huang), yxiang2@gmail.com (Y. Xiang), cugzdm@foxmail.com (D. Zhang).

http://dx.doi.org/10.1016/j.jpdc.2016.11.013
0743-7315/



(a) The given placement. (b) The changed placement.

Fig. 1. Effects of placement changes on the moving distance of buffers.

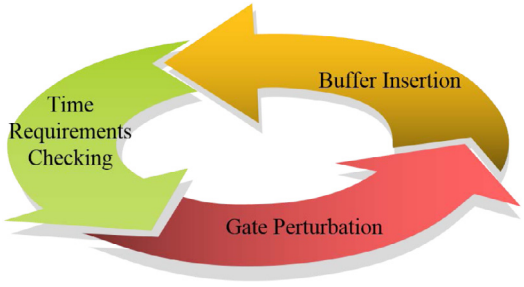


Fig. 2. An energy aware buffer insertion procedure.

distance of inserted buffers. Actually, stringently applying buffer insertion algorithms to the given placement may lead to a large moving distance of inserted buffers, which imposes negative impact on interconnect delay. As a contrast, a slight change in placement may provide more flexibility for buffer insertion to improve the performance of a chip.

The example shown in Fig. 1 illustrates the effects of placement changes on the moving distance of buffers. In the given placement, the buffer can be only placed at the location between gate 2 and gate 3. If gate 8 is moved right, the fractional space between gate 8 and gate 9 is large enough for an inserted buffer. The moving range of the buffer in the changed placement is obviously less than that of the given placement, which leads to better effect of buffer insertion, and more available buffer holes are created for more inserted buffers.

The gate perturbation which is used to create more candidate buffer positions can be integrated with buffer insertion to form an energy aware buffer insertion framework. In this framework, the timing requirements of the circuits are checked. When timing requirements are not all satisfied, buffer insertion technique can be utilized to improving the timing. But if there are not enough candidate buffer positions for buffer insertion, the gate perturbation, which is the maximization algorithm for candidate buffer position, is performed on a part of the chip. Note that, if the gate perturbation and the buffer insertion continually proceed after timing requirements have been met, it will take more computational resources in terms of energy. This can further impact the energy consumption of the high level infrastructures [19,12]. For considering this issue, an energy aware buffer insertion procedure can perform as shown in Fig. 2. From the energy-saving point of view, to meet the timing requirements, the energy aware buffer insertion framework first evaluates whether the time requirements are satisfied. If not, a part of the chip performs the gate perturbation technique followed with buffer insertion technique. After that, timing constraints are re-evaluated to determine whether a new iteration of gate perturbation and buffer insertion are required on another part of the chip. This procedure iteratively proceeds until the timing requirements have been met. The above exhibits essential properties of a Cyber–Physical System (CPS), which is very popular in many research fields, such as researches on cyberinfrastructures [18] and parallel computing [14,20]. Since there are several mature buffer insertion algorithms, this work only focus on the gate perturbation in buffering stage of the design CPS. In this paper, an innovative technique is proposed for perturbation constrained buffer planning problem. Perturbation is referred to as the moving of gates, the proposed algorithm is able to create more buffer holes on chip, and compute a good placement to minimize the total moving distance of inserted buffers. Experimental results demonstrate that the proposed algorithm achieve at most 41.49% increase in the numbers of available buffer holes on chip when compared to the algorithm with no movement of gates, and with the increase of available buffer holes on chip, the inserted buffers can choose nearer buffer holes, which can effectively reduce the total moving distance of inserted buffers, which indicates that the effectiveness of buffer insertion can be significantly improved.

The main contributions of the paper are summarized as follows.

- To the best of authors' knowledge, this is the first work dealing with perturbation constrained buffer planning problem in a CPS design framework.
- A novel technique for perturbation constrained buffer planning problem is proposed. Instead of greedily moving buffers to the available buffer holes, the proposed algorithm make changes to the placement to explore available more buffer holes by utilizing the ILP technique. The proposed algorithm maximizes the candidate buffer positions for improving the effectiveness of buffer insertion.
- The divide and conquer method and parallel technique which is based on the MPI technique are used according to the features of the CPS design. It helps to improve the computation efficiency.

The rest of the paper is organized as follows. Section 2 presents the problem formulation for the perturbation constrained buffer planning. Section 3 proposes the techniques for perturbation constrained buffer planning problem. Section 4 presents the experimental results and analysis. Section 5 concludes the paper.

2. Problem formulation

In the buffering design CPS framework, the chip for the design is divided into number of sub-regions. Buffer planning algorithm and buffer insertion algorithm are performed on the sub-regions till the timing constraints have been met. In this work, we focus on the buffer planning approach, and for a sub-region of the chip this problem can be interpreted as follows. Given a placement of n gates on a sub-region, the base area of the sub-region can be treated as a 2-D plane, which is divided into a set of rows. In this paper, a cell is referred to as either a buffer or a gate. As is shown in Fig. 3, a cell is aligned with an integer track. The intersection of a track and a row is a grid point. The lower left corner of a cell can only be placed at a grid point. Denote by g_i the gate i, where $i \leq n$. Denote by b a buffer. Each gate g_i is associated with a moving range L_{g_i}. This means that

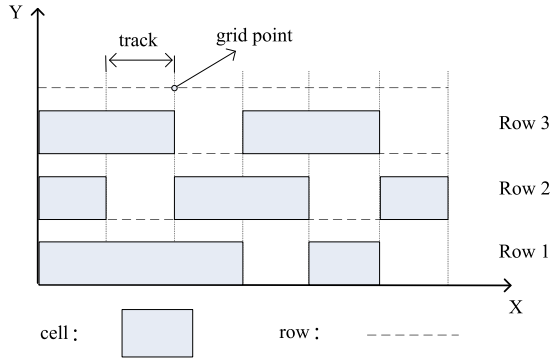


Fig. 3. A placement of gates on a chip.

for vertical moving, a gate can be moved at a grid point as far as L_{g_i}, and for horizontal moving, all gates can be placed in a row as far as L_{g_i}.

After placement design, the space between gates varies. Some of them are large enough to contain one or several buffers, but some of them are not able to contain one buffer. Within the moving range, gates can be moved to explore more flexibility for buffer insertion in the layout. Hence, the space between gates can be tuned and be better utilized for buffer insertion. Given a placement of n gates and the moving range of each gate, after randomly generating a set of buffers on the sub-region, the perturbation constrained buffer planning problem is to minimize the total moving distance of inserted buffers, which need to be located at an available buffer hole, such that the cells do not overlap. Perturbation is referred to as the moving of gates, the moving range of a gate can limit the impact of perturbation to maintain the quality of the original placement. Note that, during this moving procedure, no overlap is allowed. In practice, for different technology nodes, such as 32 or 45 nm technologies, the buffer size can be different. Thus the corresponding candidate buffer position size can be set according to different technology nodes. Formally, the problem is defined as follows.

Perturbation constrained buffer planning problem:

Given some buffers with unit length and width, a placement of n gates on a sub-region of a chip where each gate is associated with a moving range, and a set of randomly generated buffers on the chip, the goal is to maximize the total candidate buffer holes on this sub-region of the chip, by subjecting to moving distance.

3. Algorithms

The proposed algorithm which is designed for the physical design flow allows small moving of gates. The problem is formulated as an integer linear programming (ILP) problem. Since a large number of gates impose negative effects on the efficiency of ILP solving, a strip-based technique adopting divide and conquer method is applied to decompose the ILP problem into a set of ILP subproblems. Boundary issues on the chip are also effectively taken care of by the proposed algorithm.

3.1. Integer linear programming formulation

A grid point can be either occupied by a buffer holes or a gate. More buffer holes on a sub-region of the chip, more flexibility it has to move inserted buffers to reduce the total moving distance. The goal of the buffer planning is to optimize the available buffer holes on the sub-region of the chip by adjusting the location of gates within their moving range. The buffer planning problem can be represented as integer linear programming (ILP) that optimizes for the number of buffer holes.

A buffer is tentatively inserted at each grid point called a candidate buffer hole. Denote by a binary variable β the candidate buffer hole, i.e., $\beta \in \{0, 1\}$. $\beta_k = 1$ means that the lower left corner of a buffer b can be placed at the kth candidate buffer hole. That is, it is a buffer hole. Otherwise, the buffer cannot be placed at the kth candidate buffer hole. That is, it is not a buffer hole. The total number of the candidate buffer hole is the number of grid points on a chip, denoted by m, where $k \leq m$.

For each gate g_i, all the possible grid points within its moving range of L_{g_i} are considered as candidate locations of the gate. Define a binary variable $\gamma_{i,j}$ to be the jth candidate location of g_i, i.e., $\gamma_{i,j} \in \{0, 1\}$. $\gamma_{i,j} = 1$ means that the lower left corner of the g_i is placed at the jth candidate location. Otherwise, it is not placed at the jth candidate location. In the solution, each g_i needs to be placed at exactly one location, for $1 \leq i \leq n$. Then one has

$$\sum_j \gamma_{i,j} = 1, \quad \forall g_i. \tag{1}$$

To ensure that gates or buffers do not overlap with one another, at most one cell can be placed to cover a grid point. As is shown in Fig. 4, grid point k may be covered by some candidate locations of cell 1, cell 2 and cell 3. However, at most one cell can be placed at grid point k. In this example, cell 2 is placed at the grid point. Formally, for any grid point p, denote by $G(p)$ all the candidate locations $\gamma_{i,j}$ of all gates, where $\gamma_{i,j}$ must cover the grid point p. Denote by $B(p)$ the candidate buffer holes β_k which can cover the grid point p. The sum of all the candidate locations for all cells at the grid point and all the candidate buffer holes at the grid point cannot be greater than one. That is,

$$\sum_{i,j} G(p) + \sum_k B(p) \leq 1, \quad \forall \text{ grid point } p. \tag{2}$$

The objective of buffer planning is to maximize the number of the buffer holes, i.e., max $\sum_{k=1}^{m} \beta_k$, where m is the total number of grid points. Assuming that the boundary issues have been taken care of for simplicity, the complete integer linear program for perturbation constrained buffer planning problem is shown as follows:

$$\begin{aligned} \max \quad & \sum_{k=1}^{m} \beta_k \\ \text{s.t.} \quad & \sum_j \gamma_{i,j} = 1, \quad \forall g_i \\ & \sum_{i,j} G(p) + \sum_k B(p) \leq 1, \quad \forall \text{grid point } p \\ & \beta_k, \gamma_{i,j}, \ G, \ B \in \{0, 1\}. \end{aligned} \tag{3}$$

It is helpful to look at a simple example for illustration of the above integer linear programming formulation. As is shown in Fig. 5, there are 1 row and 6 tracks in a chip area. Thus, there are 6 grid points. Suppose that there are 2 gates. The moving range of gate 1 is 1, and the moving range of gate 2 is 2. The candidate locations of gate 1 and gate 2 are shown. Buffers are tentatively inserted at each grid point. Thus, one has candidate buffer holes β_k, $1 \leq k \leq 6$, The objective of the integer linear programming is to maximize the number of buffer holes, i.e., $\sum_{j=k}^{6} \beta_k$. For each gate, the sum of the candidate locations needs to be equal to one, i.e., $\gamma_{1,1}+\gamma_{1,2}+\gamma_{1,3} = 1$, and $\gamma_{2,1}+\gamma_{2,2}+\gamma_{2,3}+\gamma_{2,4}+\gamma_{2,5} = 1$. For each grid point, the sum of the candidate locations that can cover the grid point cannot be greater than one. For example, for grid point 1, one has, $\gamma_{1,2} + \gamma_{2,1} + \beta_2 \leq 1$. Note that $\gamma_{i,j} = \{0, 1\}$, $\beta_k = \{0, 1\}$.

The object and related constrained conditions of Fig. 5 are as follows.

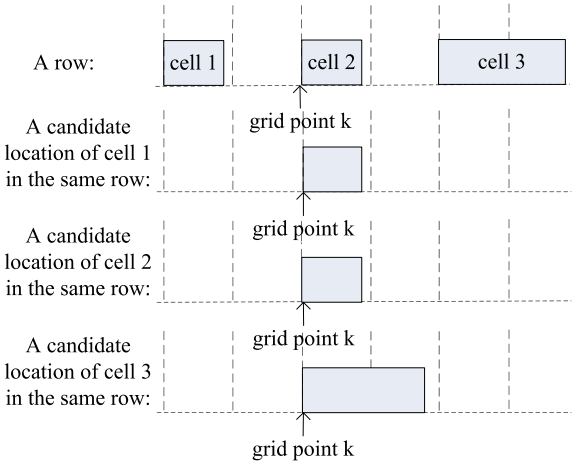


Fig. 4. Illustration of a covered grid point.

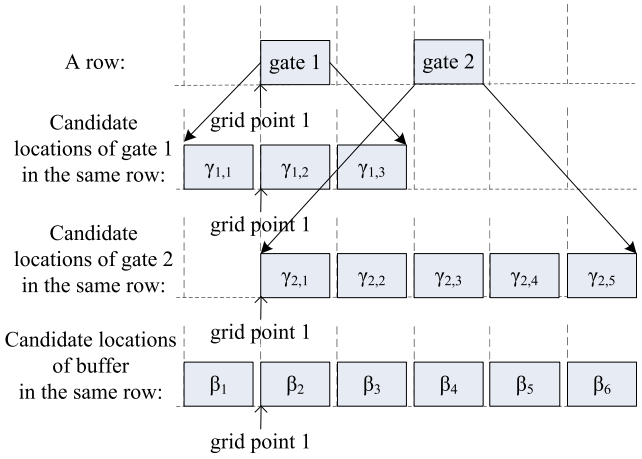


Fig. 5. Illustration of the integer linear programming formulation.

3.2. Strip division and combination

As discussed in Section 1, in the buffering CPS design framework, the perturbation constrained buffer planning is performed on part of the chip to check if the buffer planning can satisfy the timing constraints. Once the timing constraints have been met, no more planning or buffering insertion is required, such that the computing resource can be saved. For this reason, the targeting chip is divided into a set of disjoint strips of uniform sizes. Each strip consists of a number of gates. By now the original ILP problem is decomposed into a set of ILP subproblems as shown in Fig. 6. These ILP subproblems are solved independently and can be proceeded in parallel. After all timing constraints have been met, the strips are summed up as a whole chip.

$$\begin{aligned} \max \quad & \sum_{k=1}^{6} \beta_k \\ \text{s.t.} \quad & \\ & \gamma_{1,1} + \gamma_{1,2} + \gamma_{1,3} = 1 \\ & \gamma_{2,1} + \gamma_{2,2} + \gamma_{2,3} + \gamma_{2,4} + \gamma_{2,5} = 1 \\ & \gamma_{1,1} + \beta_1 \le 1 \\ & \gamma_{1,2} + \gamma_{2,1} + \beta_2 \le 1 \\ & \gamma_{1,3} + \gamma_{2,2} + \beta_3 \le 1 \\ & \gamma_{2,3} + \beta_4 \le 1 \\ & \gamma_{2,4} + \beta_5 \le 1 \\ & \gamma_{2,5} + \beta_6 \le 1. \end{aligned} \tag{4}$$

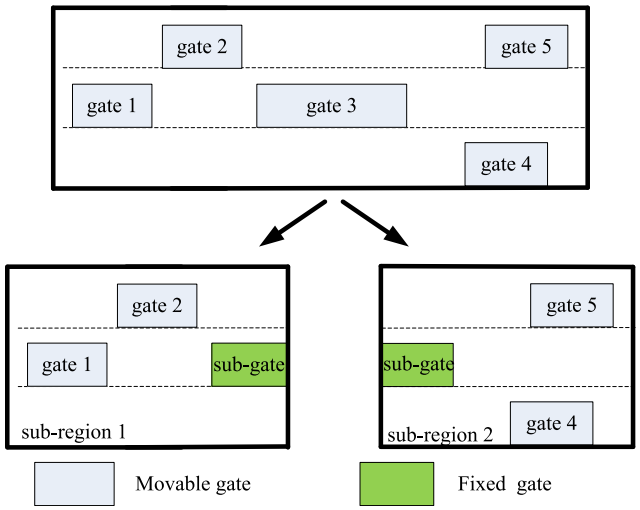


Fig. 6. Illustration of a gate which crosses the boundary line of strip.

Since the proposed approach is to be performed after placement&routing stage, the moving distance of gates has to be restricted to a relatively small area, such that the results of placement&routing are not changed too much. For this reason, the chip can be conceptually divided into sub-regions and the proposed approach can be ran on each sub-region. When the chip is divided into sub-regions, gates located on the boundary need to be took care of. As shown in Fig. 6, the gate on the boundary can be treated as sub-gates, each of them has a fixed location on the corresponding sub-region. This kind of sub-gates are not able to be moved with the proposed algorithm. Thus, these sub-gates can be guaranteed to be combined as one when sub-regions merge together. Note that, in the experiment of this work, the chip is divided into sub-regions which has fixed size. The size of the sub-region is set to be 100×48 which contains about 80 gates. In addition, most of chips are row based ones and their have gates with identical height. For this reason, the proposed algorithm always perform the horizontal cuts along gate rows; and only vertical cuts introduce sub-gates.

3.3. Parallel technique

3.3.1. Divide and conquer method

The proposed method aims to provide candidate buffer holes on chip, which has been converted into an integer linear programming problem. For a whole chip, the number of gates follows the divide and conquer method to achieve a better performance for the CPS based energy aware buffer insertion. Take Fig. 7 as an example, the chip can be divided into four sub-regions. For each of them, an ILP can be formulated to maximize the candidate buffer locations. These integer linear programs can be solved in parallel as long as the gates on the boundaries being processed in a proper way. In this work, these crossing boundary gates are set to be fixed, such that, they have no impact on the combining step of sub-regions solutions.

3.3.2. MPI cluster

According to the attributes of the proposed algorithm, divide and conquer method is utilized to decompose the chip into a number of sub-regions. The integer linear program of the large chip is decomposed into a number of integer linear programs of sub-regions, each of them can be processed in a parallel way. In this work, a customized MPI cluster has been built to speed up the integer linear programs solving procedures. The main idea of the proposed MPI cluster has been shown in Fig. 8(a), which consists of a master node and several client nodes.

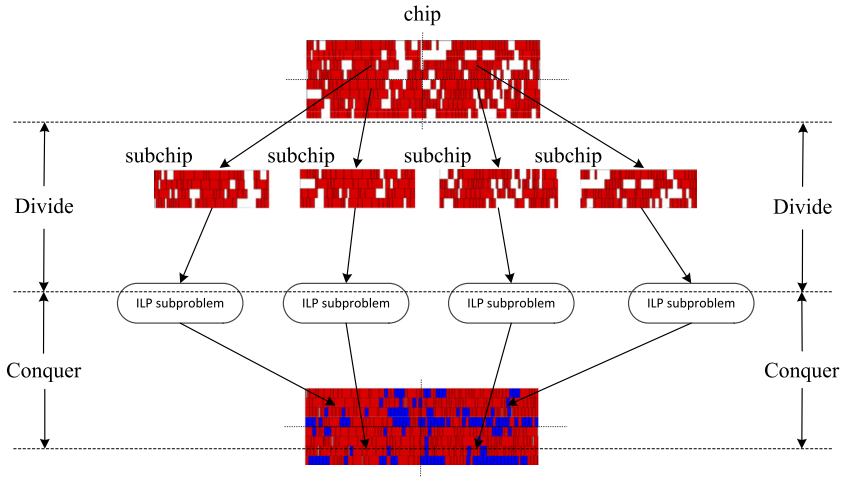
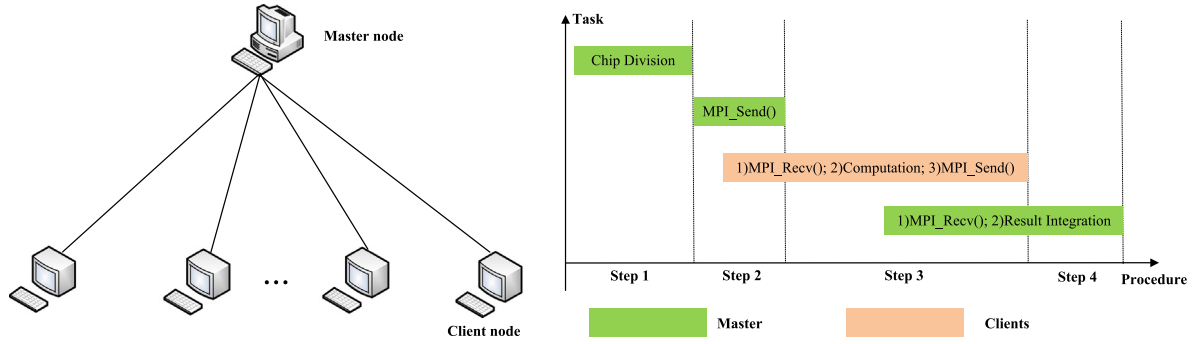


Fig. 7. An energy aware buffer insertion procedure.



(a) An MPI cluster example. (b) The process procedures in MPI cluster.

Fig. 8. MPI cluster process procedure.

The computing procedures in the MPI cluster can be divided into four steps which has been shown in Fig. 8(b). Firstly, a chip of large size is divided into a number of sub-regions on the master node. In this way, the integer linear programs of the chip is decomposed into several smaller size integer linear programs of sub-regions. These decomposed integer linear programs are about to be solved by client nodes independently. The master node utilizes the `MPI_Send` function to send these integer linear programs of sub-regions to client nodes. Once a client node receives an integer linear program from the master node using `MPI_Recv` function, the computation of the corresponding integer linear programs can take place to solve independently. After integer linear programs being solved, the client node sends the result back to the master node by `MPI_Send` function. At last, the master node calls the `MPI_Recv`, receives all the computation results sent by the client nodes in cluster, and integrates all the computation results.

4. Experimental results

The experiments have been performed on an MPI cluster with four computation nodes, each node of the cluster is with 3.4 GHz Intel(R) *i*7 cores and 8 GB memory. The proposed perturbation constrained buffer planning approach is implemented in C++. The *MPICH* 1.4, which is the widely used MPI standards [7], is utilized for communication functions in the Cluster. We took part of the Capo's solution on bigblue1 benchmark circuits [16] as the initial chip placement for our experiments. Note that, since this work focused on the buffer planning approach, we do not perform the buffer insertion and timing constraints check in the experiment.

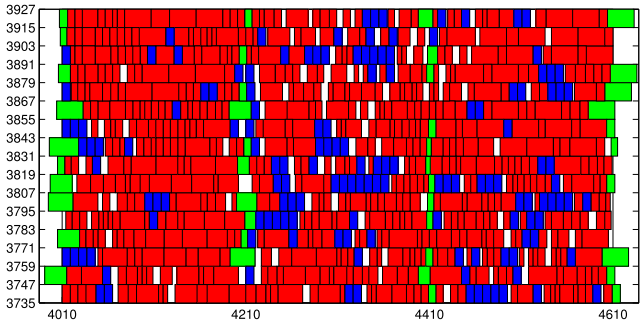


Fig. 9. The distribution of gates and buffers on the chip based on the widely-used buffer insertion technique.

We assume that the chip needs as much buffer candidate location as possible.

To illustrate the correctness and effectiveness of the proposed method, we first show the results for a small part of the chip. The chip has been divided into pieces of subchips for parallel computing, where red blocks refer to gates that can be moved within their moving ranges and green blocks refer to gates that are fixed at the boundary of the chip. It is clear that the moving ranges of these gates are 0.

Fig. 9 gives available candidate buffer insertion positions without running our proposed algorithm, where blue blocks refer to the available candidate buffer insertion positions. In contrast, Fig. 10 shows available candidate buffer insertion positions after running our proposed algorithm. It is clear that, before our proposed method is run, most gaps on chip are not wide enough to

Table 1
Experimental results on the number of buffer holes by applying gates moving based technique and the computation time by utilizing different computation nodes in the MPI cluster.

Test case	*GateNum*	N_{before}	N_{after}	*Improvement*	T_1 (s)	T_2 (s)	T_3 (s)	T_4 (s)
1	12 402	1 533	2 169	41.49%	1403	683	468	451
2	23 684	3 579	4 728	32.10%	2479	1410	877	885
3	34 560	5 841	7 446	27.48%	3478	1746	1265	1159
4	45 201	8 256	10 148	22.92%	4570	2435	1647	1407
5	55 614	10 051	12 594	25.30%	6033	3089	2056	1673

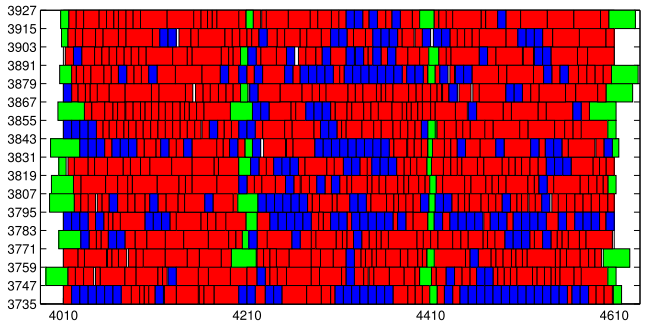


Fig. 10. The distribution of gates and buffers on the chip based on the gates-moving buffer insertion technique.

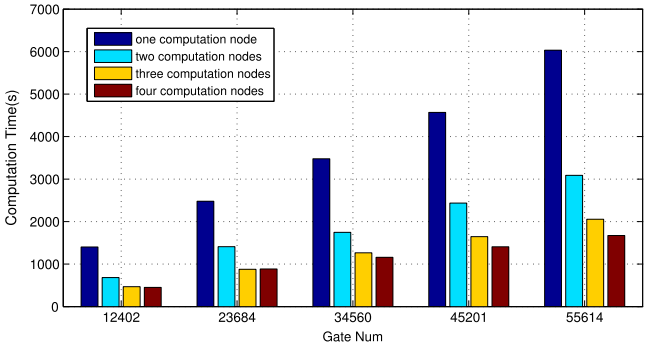


Fig. 11. Computation time with different gate num and computation nodes.

insert buffers, which somehow leads to a waste of chip resources. In contrast, after our proposed method is run, only a few narrow gaps remain and the total number of candidate buffer positions are significantly increased.

Fig. 11 shows the computation cost with different computing nodes which includes the master node and client nodes. The following observations are made:

- The parallel technique can effectively reduce the computational time. The reason is that, the proposed method has a large amount independent workload which can be paralleled, and the communication cost is considerable less than the computing cost.
- With increasing number of computing nodes in the MPI cluster, the computational time is gradually decreased. The computational time of four nodes is significantly decreased when compared to the case of one node. However, the communication time and parallel overhead between nodes in the MPI cluster increase when we increase the number of computing node.

Table 1 compares the numbers of candidate buffer holes before and after running our proposed algorithm. It also shows the computing time of the proposed algorithm when utilizing different number of computing nodes in the MPI cluster. In Table 1, *GateNum* refers to the total gate number of the chip, N_{before} refers to the number of candidate buffer holes of the chip before running our proposed algorithm, N_{after} refers to the number of candidate buffer holes of the chip after utilizing our proposed algorithm, *Improvement* refers to the improvement of N_{before} over N_{after}, and T_1, T_2, T_3, T_4 refers to the computing time in the MPI cluster by utilizing up to 4 computation node(s).

Following observations are made:

- The candidate buffer holes of the chip both before and after running the proposed method are increased, when we increase the gate number and the chip size. Because there are more spaces that can be utilized on the chip.
- The proposed method can always provide more candidate buffer holes when compared to the initial placement. The improvement can be up to 41.49% ($GrowthRate = \frac{N_{after} - N_{before}}{N_{before}}$).
- The improvement ratio of buffer holes is decreasing from testcase 1 to testcase 4. The reason is that gates on the chip are not evenly distributed, thus some parts of the chip initially have more number of gates and more blank space. But as the number of gates are increasing, the number of buffer holes, which refers to N_{after}, are always increasing as shown in Table 1.

Our proposed approach is targeting to provide more candidate buffer position for buffer insertion based on the placement testbench which has gate size and location information. But the testbench does not have interconnection information to perform the buffer insertion algorithm. To verify the relation between the number of candidate buffer locations and circuits timing, we utilize the benchmark which contains 500 industrial nets and run the buffer insertion algorithm [8] with different number of candidate buffer locations. Results have been shown in Fig. 12. With increasing number of candidate buffer locations, the average delay over 500 industrial nets is decreasing. When the number of candidate buffer locations is increased by 40%, the timing delay can be improved by 8.9%; and if the number of candidate buffer locations is increased by 20%, the timing delay can be improved by 5.9%. Note that, in Table 1, the improvement of the proposed algorithm is in between 22.92% and 41.49%.

5. Conclusion

As VLSI technology advances towards nanoscale devices, interconnect delay is becoming increasingly important, and could be effectively reduced using buffer insertion. The widely-used buffer insertion technique in industry is to insert a set of buffers on the chip, identify overlapping buffers and gates, and move the buffers to the nearest available buffer holes. The moving distance of inserted buffers largely affects the wire length which may result in the increase of the interconnect delay. Meanwhile the CPS based buffering design framework desires a power aware buffering approach can efficiently handle the timing issue of the chip. This paper is the first work that proposes efficient algorithms on perturbation constrained buffer planning in placement based on integer linear programming under a CPS based buffering design framework. Instead of directly moving buffers to existing available buffer holes in the given placement, the proposed algorithm changes the placement with constrained perturbation for better buffer insertion. In addition, the proposed technique is respectively designed for given small moving ranges of gates on chip. Experimental results demonstrate that the proposed

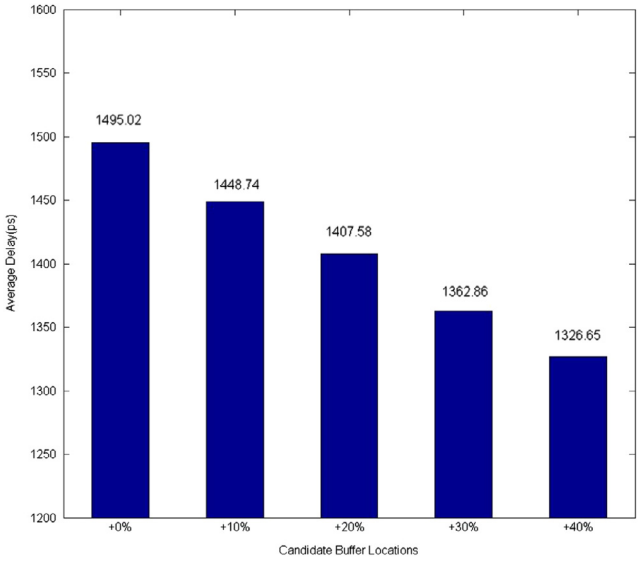


Fig. 12. Circuits delay with different number of candidate buffer locations.

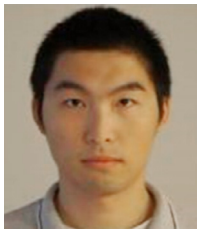
algorithms achieve up to 41.49% increase in the number of buffer holes when compared to the algorithm with no movement of gates, which indicates that the effectiveness of buffer insertion can be largely improved to obtain high-performance VLSI designs.

Acknowledgment

This study was supported in part by the National Natural Science Foundation of China (No. 61501411).

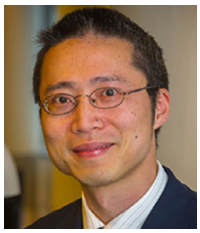
References

[1] C. Alpert, A. Devgan, S. Quay, Buffer insertion for noise and delay optimization, in: Proceedings of ACM/IEEE Design Automation Conference, 1998, pp. 362–367.
[2] C. Alpert, M. Hrkic, S. Quay, A fast algorithm for identifying good buffer insertion candidate locations, in: Proceedings of International Symposium on Physical Design, 2006, pp. 47–52.
[3] C. Chu, D. Wong, A new approach to simultaneous buffer insertion and wire sizing, in: Proceedings of the International Conference on Computer Aided Design, 1997, pp. 614–621.
[4] J. Cong, An interconnect-centric design flow for nanometer technologies, Proc. IEEE 89 (4) (2001) 505–528.
[5] J. Cong, T. Kong, D. Pan, Buffer block planning for interconnect-driven floorplanning, in: Proceedings of the International Conference on Computer Aided Design, 1999, pp. 358–363.
[6] L. Deng, M. Wong, Buffer insertion under process variations for delay minimization, in: Proceedings of the International Conference on Computer Aided Design, 2005, pp. 317–321.
[7] http://www.mpich.org/, http://www.mpich.org/.
[8] S. Hu, C. Alpert, J. Hu, S. Karandikar, Z. Li, W. Shi, C.N. Sze, Fast algorithms for slew constrained minimum cost buffering, IEEE Trans. Comput.-Aided Des. (TCAD) 26 (11) (2007) 2009–2022.
[9] S. Hu, Z. Li, C.J. Alpert, A fully polynomial time approximation scheme for timing driven minimum cost buffer insertion, in: Proceedings of ACM/IEEE Design Automation Conference, DAC.
[10] Z. Jiang, S. Hu, J. Hu, Z. Li, W. Shi, A new RLC buffer insertion algorithm, in: Proceedings of the International Conference on Computer Aided Design, 2006, pp. 553–557.
[11] E. Kan, C. Uttraphan, Shaikh-Husin, An optimized algorithm for simultaneous routing and buffer insertion in multi-terminal nets, in: Proceedings of International Conference on Electrical and Electronic Engineering, 2015, pp. 1–9.
[12] S.U. Khan, L. Wang, L.T. Yang, F. Xia, Green computing and communications, J. Supercomput. 63 (3) (2013) 637–638.
[13] R. Li, D. Zhou, J. Liu, X. Zeng, Power-optimal simultaneous buffer insertion/sizing and wire sizing, in: Proceedings of the International Conference on Computer Aided Design, 2003, pp. 581–586.
[14] Y.M.A.Z. Lizhe Wang, Jining Yan, Pipscloud: High performance cloud computing for remote sensing big data management and processing, Future Gener. Comput. Syst. (2016) http://dx.doi.org/10.1016/j.future.2016.06.009.
[15] M.A.A.S. Mohapatra, M. Chrzanowska-Jeske, Buffered interconnects in 3d ic layout designs, in: Proceedings of Proceedings of the 18th System Level Interconnect Prediction Workshop, (4), 2016, pp. 4:1–4:8.
[16] J. Roy, D. Papa, S. Adya, H. Chan, A. Ng, J. Lu, I. Markov, Capo: Robust and scalable opensource mincut floorplacer, in: Proceedings of International Symposium on Physical Design, 2005, pp. 224–227.
[17] L. van Ginneken, Buffer placement in distributed RC-tree networks for minimal elmore delay, in: Proceedings of International Symposium on Circuits and Systems, 1990, pp. 865–868.
[18] L. Wang, D. Chen, Y. Hua, Y. Ma, J. Wang, Towards enabling cyberinfrastructure as a service in clouds, in: Proceedings of International Symposium on Physical Design, Vol. 39, (1), 2013, pp. 3–14.
[19] L. Wang, S.U. Khan, D. Chen, J. Kolodziej, R. Ranjan, C. Xu, A.Y. Zomaya, Energy-aware parallel task scheduling in a cluster, Future Gener. Comput. Syst. 29 (7) (2013) 1661–1670.
[20] L. Wang, J. Tao, R. Ranjan, H. Marten, A. Streit, J. Chen, D. Chen, G-hadoop: Mapreduce across distributed data centers for data-intensive computing., Future Gener. Comput. Syst. 29 (3) (2013) 739–750.

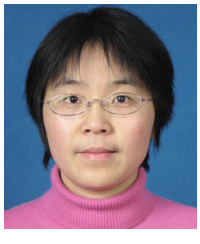


Xiaodao Chen received the B.Eng. degree in telecommunication from the Wuhan University of Technology, Wuhan, China, in 2006, the M.Sc. degree in electrical engineering from Michigan Technological University, Houghton, USA, in 2009, and the Ph.D. in computer engineering from Michigan Technological University, Houghton, USA, in 2012. He is currently an Associate Professor with School of Computer Science, China University of Geosciences, Wuhan, China. His research interests include Design Automation for petroleum system, High Performance Computing and Optimization.

Xiaohui Huang received B.S. degree from China University of Geosciences. He is currently a graduate student with School of Computer Science, China University of Geosciences, Wuhan, China. His research interests include Optimization and data analysis.



Yang Xiang received his Ph.D. in Computer Science from Deakin University, Australia. He is the Director of Centre for Cyber Security Research, Deakin University. His research interests include network and system security, data analytics, distributed systems, and networking. In particular, he is currently leading his team developing active defense systems against large-scale distributed network attacks. He is the Chief Investigator of several projects in network and system security, funded by the Australian Research Council (ARC). He has published more than 200 research papers in many international journals and conferences, such as IEEE Transactions on Computers, IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Information Security and Forensics, and IEEE Journal on Selected Areas in Communications. Two of his papers were selected as the featured articles in the April 2009 and the July 2013 issues of IEEE Transactions on Parallel and Distributed Systems. Two of his papers were selected as the featured articles in the Jul/Aug 2014 and the Nov/Dec 2014 issues of IEEE Transactions on Dependable and Secure Computing. He has published two books, Software Similarity and Classification (Springer) and Dynamic and Advanced Data Mining for Progressing Technological Development (IGI-Global). He has served as the Program/General Chair for many international conferences such as SocialSec 15, IEEE DASC 15/14, IEEE UbiSafe 15/14, IEEE TrustCom 13, ICA3PP 12/11, IEEE/IFIP EUC 11, IEEE TrustCom 13/11, IEEE HPCC 10/09, IEEE ICPADS 08, NSS 11/10/09/08/07. He has been the PC member for more than 60 international conferences in distributed systems, networking, and security. He serves as the Associate Editor of IEEE Transactions on Computers, IEEE Transactions on Parallel and Distributed Systems, Security and Communication Networks (Wiley), and the Editor of Journal of Network and Computer Applications. He is the Coordinator, Asia for IEEE Computer Society Technical Committee on Distributed Processing (TCDP). He is a Senior Member of the IEEE.



Dongmei Zhang received Ph.D. from China University of Geosciences. She is currently a Professor with School of Computer Science, China University of Geosciences, Wuhan, China. Her research interests include Optimization and 3D visualization.

Rajiv Ranjan received the Ph.D.degree in 2009 in engineering from the University of Melbourne. He is a research scientist and a julius fellow in CSIRO Computational Informatics Division (formerly known as CSIRO ICT Centre). His expertise is in data center cloud computing, application provisioning, and performance optimization. He has published 62 scientific, peer-reviewed papers (seven books, 25 journals, 25 conferences, and five book chapters). His hindex is 20, with a lifetime citation count of 1660+ (Google Scholar). His papers have also received 140+ ISI citations. Seventy percent of his journal papers and 60% of conference papers have been A*/A ranked ERA publication. He has been invited to serve as the guest editor for leading distributed systems journals including IEEE Transactions on Cloud Computing, Future Generation Computing Systems, and Software Practice and Experience. One of his papers was in 2011s top computer science journal, IEEE Communication Surveys and Tutorials.

Chen Liao received Ph.D. in Computer Engineering from Michigan Technological University. Houghton, USA in 2013. She is currently an engineer in Marvell Semiconductor. Her research interests are in the area of Computer-Aided Design of VLSI Circuits and Combinatorial Optimizations.