國立交通大學

電子工程學系 電子研究所碩士班

碩士論文

在佈局階段同時對緩衝器與正反器做放置規畫以及電壓下降的最小化

Simultaneous Buffer / Flip-Flop Station Planning and Voltage Drop Minimization in Floorplan Design

研 究 生：潘信華

指導教授：陳宏明 博士

中華民國九十七年一月

在佈局階段同時對緩衝器與正反器做放置規畫以及電壓下降的最小化

Simultaneous Buffer / Flip-Flop Station Planning and Voltage Drop Minimization in Floorplan Design

研究生：潘信華　　Student: Hsin-Hua Pan

指導教授：陳宏明 博士　　Advisor: Prof. Hung-Ming Chen

國立交通大學

電子工程學系　電子研究所碩士班

碩士論文

A Thesis
Submitted to Department of Electronics Engineering & Institute of Electronics
College of Electrical and Computer Engineering
National Chiao Tung University
in Partial Fulfillment of Requirements
for the Degree of
Master
in
Electronics Engineering
October 2008
Hsinchu, Taiwan, Republic of China

中華民國九十七年一月

在佈局階段同時對緩衝器與正反器做放置規畫以及電壓下降最小化

研究生：潘信華　　　　　　　　　　　　指導教授：陳宏明

國立交通大學

電子工程學系 電子研究所碩士班

摘要

隨著製程的進程，我們知道互連線已經是決定整個電路的效率以及複雜度最重要的因素。 緩衝器放置是一種對於改善互連線非常有效率的技術之一。 在佈局階段做緩衝器的放置時，通常會將緩衝器集結在一個區域，這樣有可能會因爲額外電流而造成電壓下降過大。另一方面，隨著晶片愈來愈大，操作頻率愈來愈高，使得很多的全域訊號需要好多個週期才跨過整個晶片到達目的地。這使得我們需要把互連線管線化。 我們提出了一種可以在佈局階段做互連線的管線化並且在放置緩衝器以及正反器時，我們也會避免發生電壓下降過大的問題。由實驗結果來看，我們的方法可以得到低延遲的系統，且不會發生任何電壓下降過大的問題。

Simultaneous Buffer / Flip-Flop Station Planning and Voltage Drop Minimization in Floorplan Design

Student : Hsin-Hua Pan Advisor : Hung-Ming Chen

Department of Electronics Engineering
& Institute of Electronics
National Chiao Tung University

Abstract

As the technology scales, it is well known that interconnect has become the dominant factor in determining the overall circuit performance and complexity. Buffer insertion is one of a very effective and useful techniques to improve the interconnect performance. The buffer insertion during floorplan stage usually clusters buffers in a region to minimize the area overhead, which may cause additional current and have the IR-drop violation. On the other hand, in complex digital system with relatively large die areas operating at very high frequencies, many global signals traveling across the chip need several clock cycles to reach their destinations, thus requiring the adoption of pipelined interconnects. We propose a methodology to pipeline interconnect during the floorplan stage and consider the IR-drop during the planning of buffers and flip-flops. The experimental results show our method can get a low system latency and without any IR-drop violation.

誌謝

首先要特別感謝的人，是我的指導教授陳宏明老師，因為老師的耐心指導與包容，讓我能順利的完成這篇論文。

此外，要感謝的是 VDA LAB 實驗室所有的成員，謝謝他們兩年來的砥勵、幫助及帶給我的歡樂，讓我兩年的生活充滿歡笑及淚水。

家人對我的支持、鼓勵更是我研究路上最大的依靠，對他們的感謝，更是筆墨難以形容。

最後由衷感謝所有我幫助關懷過我的人。

潘信華

民國九十七年一月 於新竹

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As technology continues to scale down, the sizes and delay of transistors and modules are getting smaller, and a significant portion of circuit delay is coming from interconnects. For deep submicron (DSM) VLSI designs, it is widely accepted that interconnect has become the dominant factor in determining the overall circuit performance and complexity. There exists many synthesis techniques, such as topology construction, driver sizing, buffer insertion, wire sizing and spacing. Among them, buffer insertion is widely recognized as a key technology for improving the performance of VLSI interconnects performance. As the intrinsic delay of a buffer becomes smaller, transistor count and chip dimension getting larger, more and more buffers are expected to be needed for high performance designs. It was projected that over 800K buffers will be inserted on a single chip in the 50nm technology [7][8].

The problem of buffer insertion has been studied extensively in the literature. An elegant dynamic programming algorithm was proposed in [19] to determine optimal buffer assignments of the candidate locations of a given interconnect topology. Several other works based on the same technique, [12][17][16] to cite a few, have also been proposed incorporating other optimization steps such as noise or area minimization, wire sizing, etc. However, those buffer insertion algorithms were designed for post-layout interconnect optimization. Because buffers consume silicon resource, it is infeasible to insert a large number of buffers individually after placement or routing when most silicon and routing resources are occupied. To overcome this problem, researchers tried to consider buffer insertion during postfloorplanning[9] [17][18], floorplanning[11][5], and placement[10]. For a given floorplan, channels and dead spaces are used as buffer blocks, which accommodate buffers[9][17][18]. If a given floorplan is not good enough, the expansion caused by the insertion of buffer would result in much area overhead. [11] integrate the buffer block planning into floorplanning stage and invoke the Lagrangian relaxation to minimize the area overhead. [5] inserts buffer during placement based on the planning of buffers at the floorplanning stage and congestion consideration.

All of these works, though, only consider the case where a buffered signal is required to arrive at its destination within one single clock cycle. However, in complex digital system with relatively large die areas operating at very high frequencies, many global signals traveling across the chip need several clock cycles to reach their destinations, thus requiring the adoption of pipelined interconnects, i.e., wiring structures in which buffers are interleaved with memory elements such as latches and flip-flops[6].

Besides, the greedy clustering of buffers into a buffer block ([9][17][11]) may increase the current in the region which are around the buffer bock and cause the IR-drop violation. Figure 1.1 shows an example of this situation. Due to the IR-drop, supply voltage in logic may not be an ideal reference. This effect may weaken the driving capability of logic gates, reduce circuit performance, slow down slew rate (and thus increase power consumption), and lower noise margin[22]. As pointed out by the authors in[15], it is important to perform IR-drop during the floorplanning. Therefore, in our methodology, when we are inserting buffers, we will consider not only the area overhead minimization but also the IR drop violation.

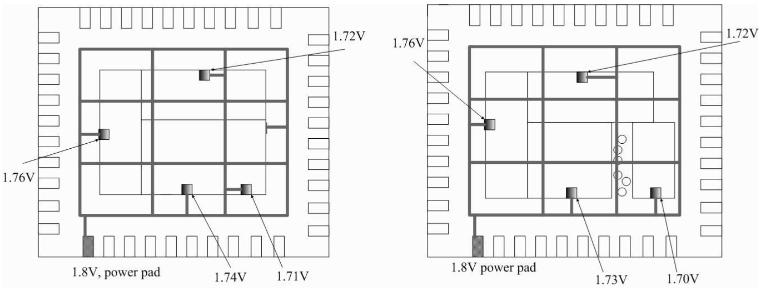


Figure 1.1: (a) An instance of floorplan and its P/G network structure. The worst-voltage at the P/G pins is about 5% of the supply voltage[15]. (b) After the buffer insertion, the worse-case voltage drop is increased from 5% to 5.6%.

1.1 Our Contributions

In this thesis, we adopt the methodology of [11] to plan buffer block and minimize the IR-drop. And we integrate the interconnect pipelining into floorplan stage to get a low system latency floorplan.

1.2 Organization of the Thesis

In Chapter 2, we briefly describe the independent feasible region (IFR) to insert buffer, interconnect pipelining for minimum latency, and how to estimate the IR-drop effectively and efficiently. In Chapter 3, we discuss our method to insert buffer while considering IR-drop, and pipelining interconnect for a intermediate floorplan. The simulated annealing algorithm in Chapter 4. Our experimental results are presented in Chapter 5. Finally we give the conclusion of this thesis in Chapter 6.

Chapter 2

Preliminaries

In this section, we introduce the independent feasible region (IFR) of a buffer, interconnect pipelining for minimum latency, IR-drop estimation and system throughput applied in our work. Then we give the problem formulation.

2.1 Delay Model

A wire of length l is modeled as a π model. A buffer is modeled with input capacitance C_b, intrinsic delay T_b, and output resistance R_b. A D-type flip-flop is modeled with input capacitance C_{ff}, and intrinsic delay T_{ff}. See Figure 2.1 for an illustration.

2.2 Independent Feasible Region (IFR)

In this section, we present the computation of independent feasible regions proposed by [17]. The independent feasible of a buffer is the region where the buffer can be placed to meet the timing requirement of the net, while the other buffers are placed within their respective independent feasible regions.

Given a wire segment of length l with driver resistance R^D, load capacitance wire resistance per unit length r_w, and wire capacitance per unit length c_w, its Elmore

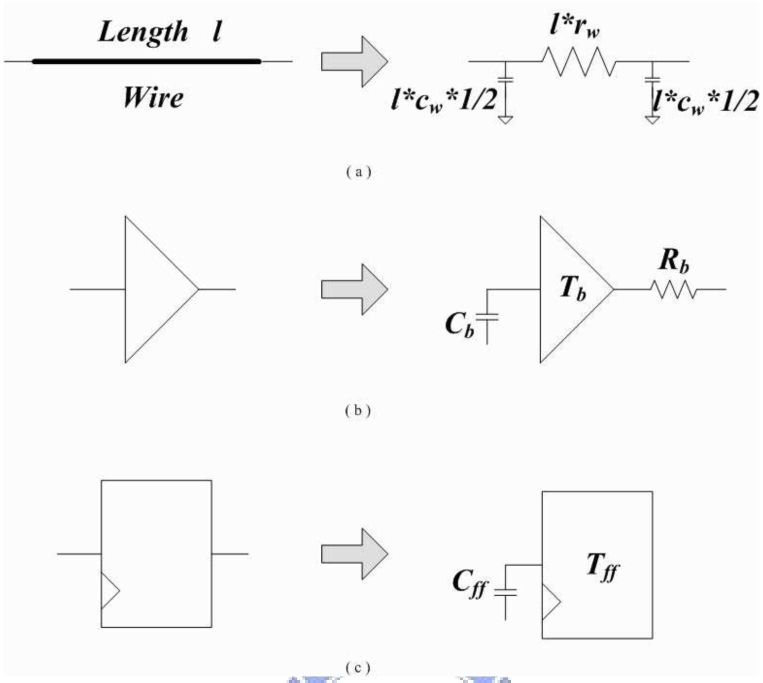


Figure 2.1: Delay model. (a) r_w and c_w are the unit-length resistance and unit-length capacitance of a wire segment. (b) C_b, R_b, T_b are the input capacitance, output resistance, and intrinsic delay of a buffer. (c) C_{ff} and T_{ff} are the input capacitance and intrinsic delay of a flip-flop.

delay is calculated by

$$D(R^D, C^L, l) = \frac{r_w c_w l^2}{2} + (R^D c_w + r_w C^L)l + R^D C^L, \tag{2.1}$$

Using the above expression, the Elmore delay of a single-source single-sink net $\mathbf{N}$ (two-pin net) of length l with n buffers can be expressed as

$$D^{\mathbf{N}}(x_1, x_2, ...x_n, l) = D(R^D, C_b, x_1) + D(R_b, C^L, l - x_n) + \sum_{i=1}^{n-1} D(R_b, C_b, x_{i+1} - x_i) + nT_b$$

where x_i is the location of the ith repeater.

The optimal locations of the n buffers for delay minimization of the net as shown in [3] are

$$x_i^* = k_1 + (i-1)k_2, i \in 1, 2, ..., n, \tag{2.2}$$

where

$$k_1 = \frac{1}{n+1}(l + \frac{n(R_b - R^D)}{r_w} + \frac{C^L - C_b}{c_w}) \tag{2.3}$$

$$k_1 = \frac{1}{n+1}(l - \frac{(R_b - R^D)}{r_w} + \frac{C^L - C_b}{c_w}), \tag{2.4}$$

We denote the optimal delay for the net $\mathbf{N}$ of length l with n buffers by

$$D^N_{opt}(n,l) = D^{\mathbf{N}}(x^*_1, x^*_2, ..., x^*_n, l). \tag{2.5}$$

The IFR of a buffers is the region where it can be placed while meeting the timing specifications of the net, assuming that the other repeaters are placed within their respective IFRs. Formally, we define the IFR for the ith repeater of a net $\mathbf{N}$ as

$$IFR_i = (x^*_i - \frac{W^i_{IFR}}{2}, x^*_i + \frac{W^i_{IFR}}{2}) \cap (0, l),$$

such that $(x_1, x_2, ..., x_i, ..., x_n) \in IFR_1 \ x \ IFR_2 \ x \ ... \ x \ IFR_n$ and $D^{\mathbf{N}}(x_1, x_2, ..., x_n, l) \leq D^{\mathbf{N}}_{req}$. Here, W_{IFR} and D^N_{req}, respectively, denote the width of IFR_i and the timing requirement associated with the net $\mathbf{N}$. Moreover, if $D^{\mathbf{N}}_{req} \geq D^{\mathbf{N}}_{opt}$, the width $W^{\mathbf{N}}_{IFR}$ of the independent feasible region for each buffer of net $\mathbf{N}$ is

$$W^{\mathbf{N}}_{IFR} \ = \ 2\sqrt{\frac{D^{\mathbf{N}}_{req} - D^{\mathbf{N}}_{opt}(n,l)}{r_w c_w (2n-1)}}, \tag{2.6}$$

In the preceding discussions, we were limited to buffers insertion along a one-dimensional (1-D) line. Implicit in the discussion was the assumption that the route from source to sink is specified by some global router. For buffer insertion during floorplanning, however, no routing information is available. We assumed that each net would be routed with a shortest path within the bounding box containing the two terminals. Therefore, we need to compute two-dimensional (2-D) regions in which the buffers can be placed. The 2-D IFR of a buffers is defined as the union of the 1-D IFRs of the buffers on all monotonic Manhattan routes between source and sink. Therefore, 2-D IFRs, is a hexagon or a degenerated hexagon bounded by the bounding box and two parallel lines of slop +1 or -1. The respective distance from the source terminal to these parallel lines are $x^*_i - W^{\mathbf{N}}_{IFR}/2$ and $x^*_i + W^{\mathbf{N}}_{IFR}/2$.

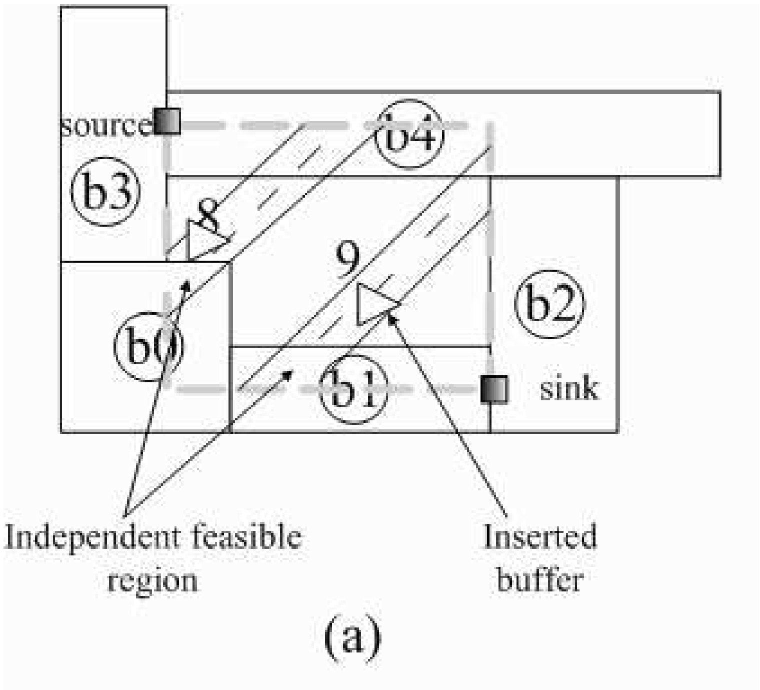


Figure 2.2: Net requires two buffer. Each buffer can be inserted into its independent feasible region. In the case shown in this figure, one buffer is inserted to the dead space *8*, the other is inserted to the dead space *9*.

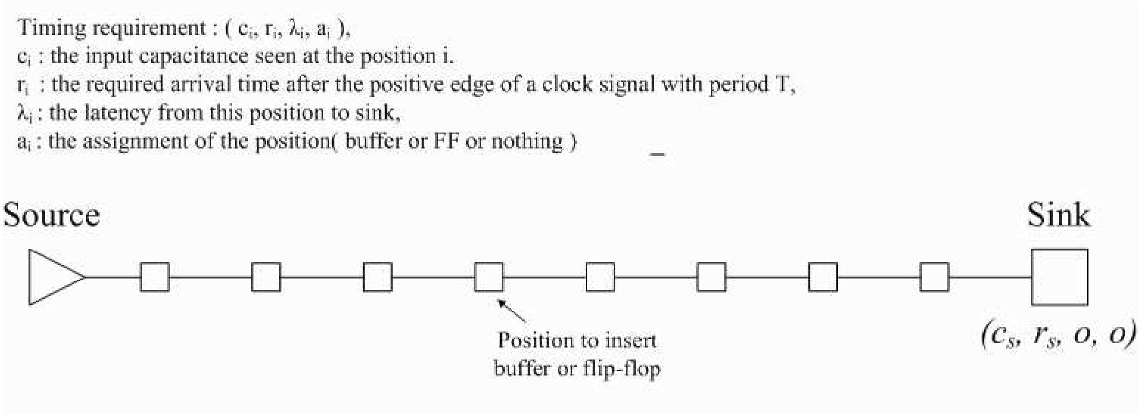
Figure 2.2 shows the independent feasible regions of two buffers on a two-terminal net **N**.

2.3 Interconnect Pipelining for Minimum Latency

As the Figure 2.3 shows, the problem is to find a assignment which satisfy the timing requirement and have the minimum latency. Here we will briefly describe the method called MiLa presented by [6].

The MiLa uses a bottom up approach which traverses from the sink to the source to get all the possible solutions while pruning the inferior solution to ease the solution space and complexity. A inferior solutions is one which will never be the optimum solution. Below we list the inferiority property:

Inferiority Property : for two solutions S_1 and S_2, S_1 is inferior to S_2 if at least



Timing requirement : (c_i, r_i, λ_i, a_i),
c_i : the input capacitance seen at the position i.
r_i : the required arrival time after the positive edge of a clock signal with period T,
λ_i : the latency from this position to sink,
a_i : the assignment of the position(buffer or FF or nothing)

Figure 2.3: Given a two-pin net, position to insert buffers or flip-flop(FF), and the timing requirements at the sink. $(c_i, r_i, 0, 0)$=(the capacitance seen at the s, the required arrival time after the positive edge of a clock signal φ with period T_φ, the interconnect cycle latency when going from s to sink, the assignment of buffer or FF at node s).

one of the following is true:

1. $\lambda_1 = \lambda_2, c_1 \geq c_2, r_1 < r_2$;

2. $\lambda_1 = \lambda_2, c_1 > c_2, r_1 = r_2$;

3. $\lambda_1 > \lambda_2, c_1 \geq c_2, r_1 \leq r_2$;

4. $\forall \lambda, \forall c, r < 0$.

In Properties 1 and 2, it is easy to see that S_1 is an inferior solution since any gate driving a net with S_1 assignment will have an input required time always worse than that of the same gate driving with assignment S_2, while having the same input capacitance and the same latency. When S_1 has a latency higher than that of S_2, as in Property 3, S_1 is inferior for the same reason as in 1 and 2 when it has identical input capacitance and required time, because S_1 and S_2 have the same timing but S_1 wastes an unnecessary extra clock cycle. Finally, when the required time is negative regardless of latency and input capacitance, as in Property 4, S_1 is inferior because it does not meet the basic timing constraint of a clocked system where the required time is bounded from zero to a maximum value equal to the clock period. Figure 2.4 is an example of this property.

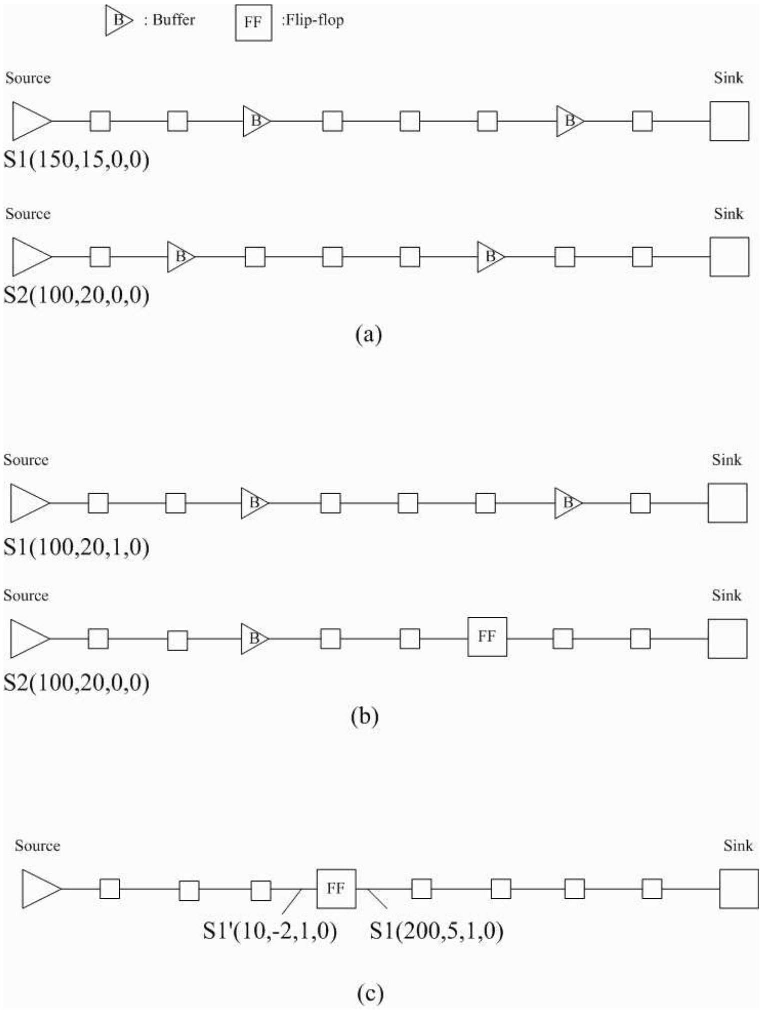
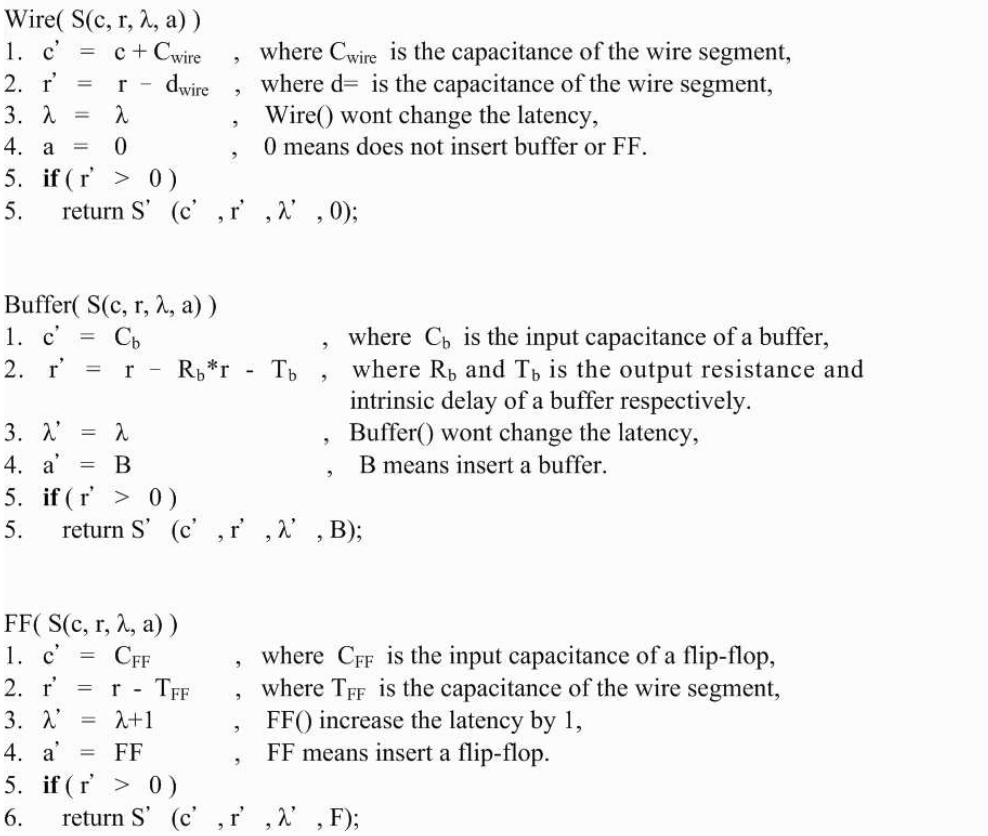


Figure 2.4: Inferiority property. (a) According to the Property 1, S_1 is inferior. (b) According to the Property 3, S_1 is inferior. (c) S_1 is the solution before inserting FF, S_1' is the solution after inserting FF. The required arrival time become -2, according to Property 4, S_1' is inferior.

The bottom up traverse approach starts from the sink using three operations recursively to get all the solutions. The three operations are *Buffer()*, *FF()*, *Wire()*. The *Buffer()* and *FF()* operation add the solution of inserting a buffer or FF to the current solution. The operation *Wire()* propagates the current solution up to the upper position by adding the wire delay and wire capacitance to the current solution. The three operation is accomplished using the delay model in Section 2.1 The pseudocode is reported in Fig 2.5.



```
Wire( S(c, r, λ, a) )
$c' = c + C_{wire}$    , where $C_{wire}$ is the capacitance of the wire segment,
$r' = r - d_{wire}$    , where d= is the capacitance of the wire segment,
$\lambda = \lambda$    , Wire() wont change the latency,
$a = 0$    , 0 means does not insert buffer or FF.
if ( r' > 0 )
    return S' (c' , r' , λ' , 0);


Buffer( S(c, r, λ, a) )
$c' = C_b$    , where $C_b$ is the input capacitance of a buffer,
$r' = r - R_b*r - T_b$    , where $R_b$ and $T_b$ is the output resistance and
                            intrinsic delay of a buffer respectively.
$\lambda' = \lambda$    , Buffer() wont change the latency,
$a' = B$    , B means insert a buffer.
if ( r' > 0 )
    return S' (c' , r' , λ' , B);


FF( S(c, r, λ, a) )
$c' = C_{FF}$    , where $C_{FF}$ is the input capacitance of a flip-flop,
$r' = r - T_{FF}$    , where $T_{FF}$ is the capacitance of the wire segment,
$\lambda' = \lambda+1$    , FF() increase the latency by 1,
$a' = FF$    , FF means insert a flip-flop.
if ( r' > 0 )
    return S' (c' , r' , λ' , F);
```

Figure 2.5: Three operations: Wire (), Buffer (), FF ().

Figure 2.6 is an example. Assumed that the segment between two adjacent positions have equal length. The delay and capacitance of each segment is $d_{wire} = 2, c_{wire} = 20$. The $T_b = 0.5$, $R_b = 0.01$, $C_b = 10$, $T_{ff} = 0.5$, and $C_{FF} = 20$. We start from the sink and the timing requirement at the sink is (50, 5, 0, 0). We first use the Wire() to propagate the solution up to position 1. The result of the Wire() is shown on Figure 2.6(b), the solution on position 1 is (70, 3, 0, 0). Then we use the Buffer() and FF() on this solution. As we can see from the Figure 2.6(c), the result is (10, 1.8, 0, B) and (20, 5, 1, FF). Now we have three solutions on position 1 which are (70, 3, 0, 0), (10, 1.8, 0, B), (10, 5, 1, FF).We then use the Wire() to propagate the solutions up to position 2. The result of Wire() is shown in Figure 2.6(d),the solution (30, -0.2, 0, 0) is inferior according to the property 4. Then the result of the operation Buffer() and FF() is shown in Figure 2.6(e). The result of Buffer() are (10, -0.4, 0, B) and (10, 2.1, 1, B) which the (10, -0.4, 0, B) is inferior. The result of FF() are (20, 5, 1, FF) and (20 ,5, 2, FF) which (10 ,5, 2, FF) is inferior to (10, 5, 1, FF) according to the property 3 and (40, 3, 1, 0) is inferior to (10, 5, 1, FF) according to property 1. The final solution on position 2 is (90, 1, 0, 0), (10, 2.1, 1, B), and (20, 5, 1, FF). Keeping the operations recursively, we will finally get a solution with minimum latency and minimum required arrival time.

2.3.1 System Throughput

Data rate is the major performance metric in system design and it is calculated by multiplying the system throughput and the operating frequency. Obviously, the throughput serves as an important performance factor. The throughput depends on not only the latency of functional blocks but also the latency incurred by long wires[20]. Below is an example of the relationship between floorplan and throughput.

It is assumed that the data transfer cannot be successfully completed from **b** to **d** in FP1 and from **b** to **c** in FP2 within one clock cycle. Hence, a pipeline element must be inserted in both cases. After inserting pipeline elements, surprisingly, the performances of two floorplans are significantly different. As shown in Figure 2.8, FP1 remains its throughput to be 1 while FP2's throughput drops from 1 to 3/4 (25% performance loss).

This example shows the strong effect of floorplanning to throughput. If the floorplan without carefully dealing with this issue, the degradation of throughput becomes large.

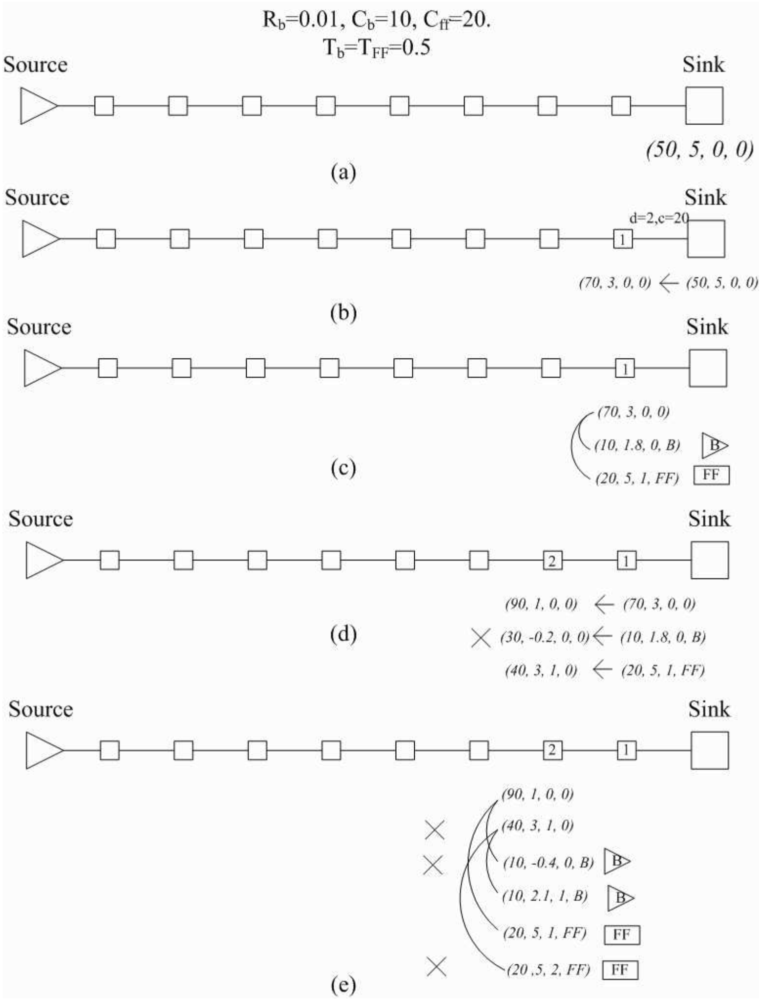


Figure 2.6: Assumed that the d_{wire} and c_{wire} of each wire segment are equal. (a)A two-pin net and the timing requirement at the sink. (b) The result of Wire() operation. (c) The result of Buffer() and FF(). (d) We use the Wire() to propagate the solution to position 2. The solution (30, -0.2, 0, 0) is inferior according to property 4. (e) The result of Buffer() and FF(). The solution (10, -0.4, 0, B) is inferior. The solution (10 ,5, 2, FF) and (40 ,3, 1, 0) are inferior according to property 3 and property 1 respectively.

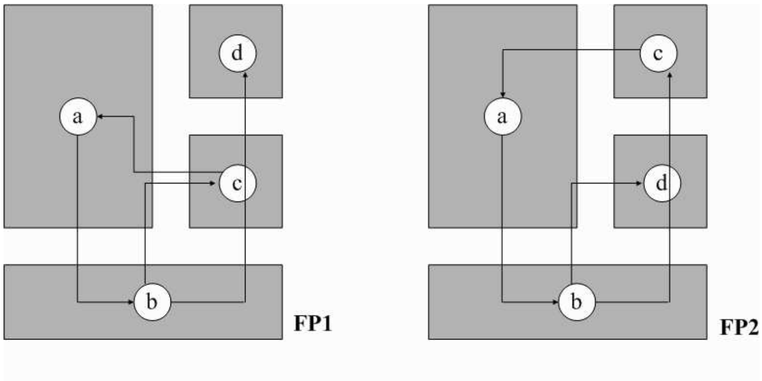


Figure 2.7: Floorplanning result FP1 & FP2[20].

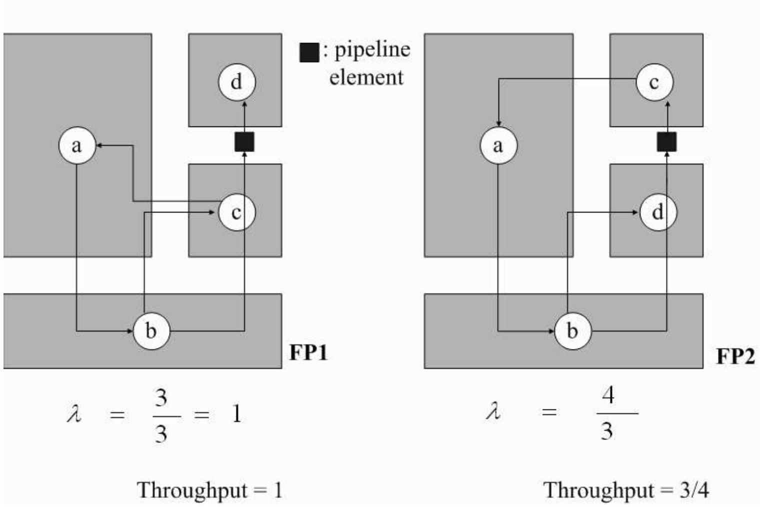


Figure 2.8: Floorplan FP1 & FP2 after pipelining[20].

2.4 Power Integrity (IR Drop)

IR-drop is mainly due to the resistance of the on-chip Power-Ground network. When large current flows through, un-acceptable voltage drop happens. As technology advances, the metal width decreases while the resistance of the power wire increase substantially. Moreover, the threshold voltage scales nonlinearly, raising the ratio of the threshold voltage to the supply voltage and making the voltage (IR) drop in the P/G network a serious challenge in modern IC design [13]. The voltage drop may cause timing uncertainty and slew rate slow-down, hence affecting performance and increasing power consumption. As [22] pointed out that 5% IR drop in supply voltage may slow down circuit performance by as mush as 15% or more.

2.4.1 The IR-Drop Constraints Definition

For every P/G pin I, its corresponding voltage V_i must satisfy the following constraint:

$V_i \geq V_{min,k}$ for each power pin i of module k,
$V_i \leq V_{max,k}$ for each ground pin i of module k,

where $V_{min,k}$ ($V_{max,k}$) is the minimum (maximum) voltage required at the injection point of a P/G network for module k.

2.4.2 IR-Drop Estimation

We adopt the efficient, yet sufficiently accurate P/G network analysis method in [15] to get a quick estimation of IR-drop in floorplanning stage. Below is the brief description of this method.

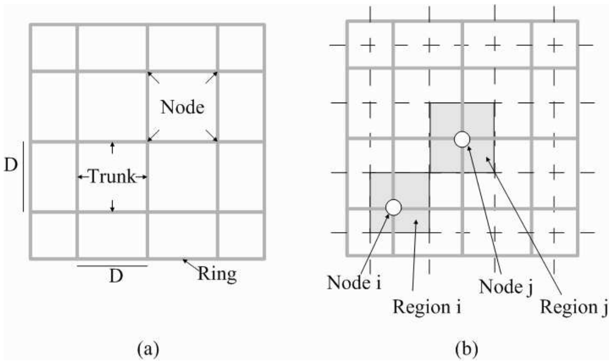


Figure 2.9: (a) A uniform P/G mesh. (b) A floorplan with a P/G mesh divided into regions[15].

The method first generate a conceptual P/G mesh network for the floorplan. We assumed that the pitch D_{pitch} of the power lines is given, then we can get a uniform mesh. Figure 2.9(a) shows a uniform mesh. To reduce the complexity, the method make a reasonable approximation by attaching all current sources to the intersection nodes of the vertical and horizontal power lines. That is, every P/G pin is connected to its nearest node with a power strap, and the length of the strap is the Manhattan distance between the P/G pin and the node. For convenience, we divide the floorplan into n regions, where n is the number of the nodes. The divided floorplan is illustrated in Figure 2.9(b). The border line of two regions is the center line between the two nodes such that the node is the nearest one for any point in the region.

For hard modules, P/G pin is connected to its nearest node and absorbs current from that node. For soft modules, the current it absorb from nodes is determined by the area that it overlapped with. Figure 2.10 shows an example of this. Finally, the mesh P/G network is modeled as the resistive P/G network model[14], and current absorbs from node is modeled as current source. As the Figure 2.11 illustrates, The static analysis of a P/G network is formulated as follows [14]:

$$\mathbf{Gx} = \mathbf{i}, \tag{2.7}$$

where **G** is the conductance matrix for the resistor, **x** is the vector of node voltages, and **i** is the vector of current loads. The dimensions of **i** and **x** are equal to the number of nodes in the P/G network, and **G** is a sparse positive definite matrix for a general resistor network.

After solving the linear equations, we will get the voltages of all the nodes. Then the voltage of each pin is estimated through this equation[15]:

$$V_j = V_i - I_j max(r_h \frac{Dx_{ij}}{w_{hstrap}}, r_v \frac{Dy_{ij}}{w_{vstrap}}), \tag{2.8}$$

where V_j is the pin voltage, V_i is his corresponding node voltage, r_v and r_h are the respective sheet resistivity of the vertical and horizontal metal layers, w_{hstrap} and w_{vstrap} are the widths of the respective vertical and horizontal straps, Dx_{ij} and Dy_{ij} are the respective vertical and horizontal distances between pin j and node i.

2.5 Problem Formulation

In this section, we give our problem formulation. We define the simultaneous buffer / flip-flop station planning and voltage drop minimization in floorplan design as follows.

- **Problem:** simultaneous buffer / flip-flop station planning and voltage drop minimization in floorplan design.

- **Objective:** Minimize area overhead, IR-drop violation, system latency, and system throughput, subject to timing requirements.

- **Inputs:** An initial floorplan of m modules, the current consumption of each modules, two pin nets, and their timing requirements, buffer library, technology file.

- **Outputs:** A floorplan with buffer / flip-flop station planning.

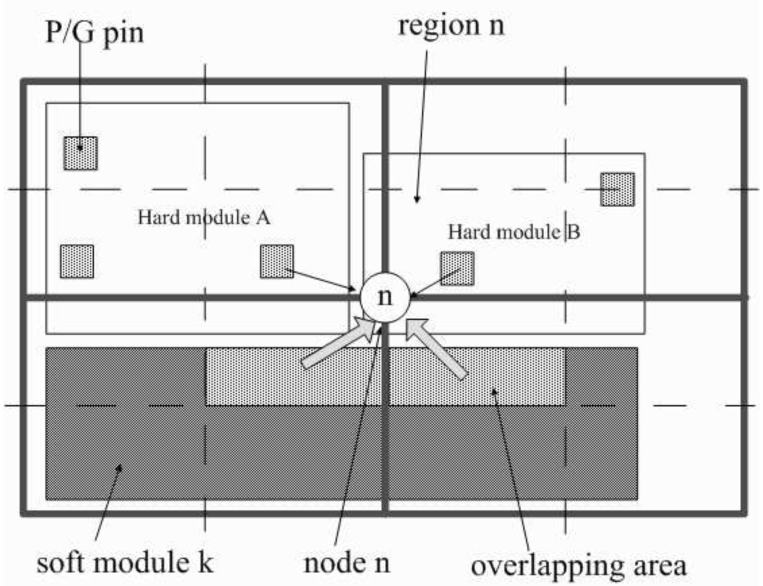


Figure 2.10: An example of the P/G analysis. The dashed lines denote the boundaries of the regions. and the gray area denotes the overlap of the soft module *k* and the region *n*. Node n is absorbed current by the hard modules A , hard module B and soft module k. Assumed that the overlapped area with node n is the 0.3 total area of soft module k and the current consumption by module k is 10uA, then the current absorbs from node n by module k is 0.3*10 uA[15].

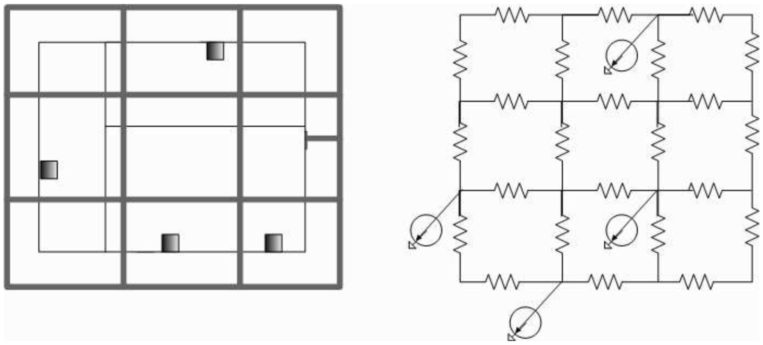


Figure 2.11: A global power mesh and its equivalent circuit model[15].

Chapter 3

Buffer / Flip-Flop Station Planning with Voltage Drop Consideration

In this section, we detail the buffer block planning and interconnect pipelining for an intermediate floorplan. For a given floorplan, we first do IR-drop estimation to know the distribution of voltage drops. Then for the nets needs only buffers, we will calculate its IFR and plan it considering the voltage drop. Nets that required more than one clock cycle time will be pipelined with minimum latency. Finally, we will do the IR-drop re-estimation to make sure that the buffer and flip-flops planning does not cause any voltage drop.

3.1 IR-Drop Estimation and Nets Separation

We start our methodology with the IR-drop estimation using the method described in Section 2.4, assumed that the pitch of the P/G mesh is given. After the IR-drop estimation, we get the voltages of all the nodes in the P/G mesh. Figure 3.1 shows an example. Then, we separate all the nets into two groups. The nets that its timing constraint will be satisfied by using only buffer will be in one group. And the nets that must be pipelined by flip-flop will belong to the other group. After the separation job, we first deal with group which only needs buffers.

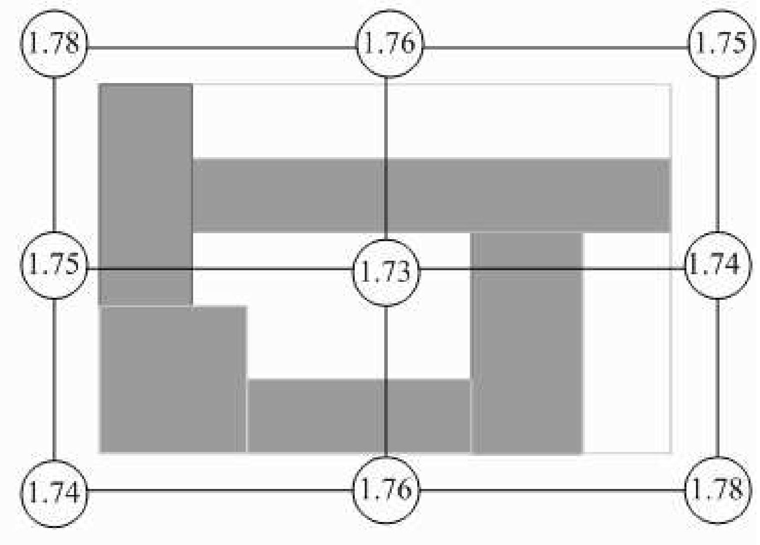


Figure 3.1: Voltages on each P/G mesh node after IR-drop estimation.

3.2 Buffer Block Planning Considering IR-Drop

For all the nets in this group, we first calculate the IFRs of each net and then insert buffers into buffer blocks which are overlapped with the IFRs. In the following, we first define the buffer block. Then we present a methodology to retrieve all existing buffer blocks efficiently.

3.2.1 Buffer Block Generation

3.2.1.1 Review of the B*-Tree Representation

A B*-tree [4] is an ordered binary tree whose root corresponds to the module on the bottom-left cornet for modeling a non-slicing floorplan. Given a admissible placement which every module can neither move down nor move left, we can construct a unique B*-tree T in a linear time. Further, given a B*-tree, we can also obtain an admissible placement by packing the blocks in a linear time with a contour structure [21].

In a B*-tree T, the root of T represents the block on the bottom-left corner;

i.e. the x- and y-coordinates of the block associated with root (x_{root}, y_{root}) = (0,0). If node n_j is the left child of node n_i, block b_j is place on the right-hand side and adjacent to block b_i in the admissible placement; i.e. $x_j = x_i + w_i$, where w_i is the width of block b_i. Otherwise, if node n_j is the right child of node n_i, block b_j is placed above block b_i with equivalent x-coordinate; i.e. $x_j = x_i$. With the contour structure, we can compute the y-coordinate of block b_j in constant time[21].

Figure 3.2 illustrates an admissible placement and its corresponding B*-tree representation. Using the depth-first search (DFS) procedure, we can recursively construct the admissible placement. First, we pick the root n_0 and place block b_0 on the bottom-left cornet. Then we traverse the left child of n_0, n_1, and place block b_1 on the right of b_0. We then traverse the left child of n_1, n_2, and place block b_2 on the right of b_1. Therefore, since n_1 and n_2 do not have right child, we traverse the right child of n_0, n_3, and place block b_3 above block b_1. Then, we traverse the left child of n_3, n_4, and place block n_4 on the right of n_3. Because all nodes are visited, we get an admissible placement.

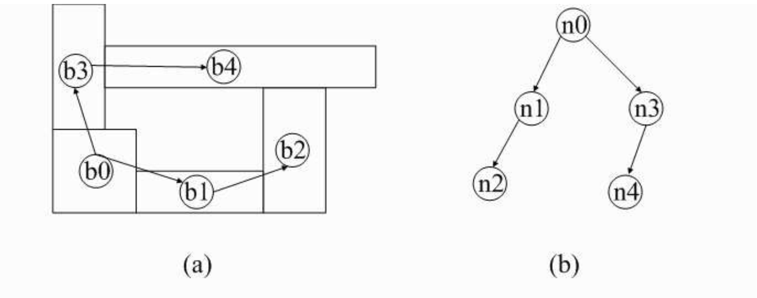


Figure 3.2: (a) An admissible placement. (b) The corresponding B*-tree[4][21].

3.2.1.2 Buffer Block Definition

Because the buffers are made of transistors, the areas occupied by existing modules are considered obstacles during buffers allocation. The empty area in a floorplan is called *deadspace* which can be used to place buffers. In addition to deadspace, there are *channels* between adjacent modules. Using channels to place buffers must pay additional penalty since the channel needs to be expanded for inserting buffers, and thus the entire circuit area will increase. Both deadspace and channels are proper locations for buffer insertion. We use the term *buffer blocks* to represent those deadspace and channels that are occupied by buffers and flip-flops.

3.2.1.3 Reverse U Boundary

We develop a data structure called reverse *U-shaped boundary*, which not only can retrieve all information of deadspace and channels precisely, but also can rebuild the B*-tree for both modules and buffer blocks at the same time. Nevertheless, we need the information of buffer blocks to refine the circuit area at the final stage. The up side of the Reverse u boundary is called up boundary and the left and right side of the u boundary is called left and right boundary respectively. As illustrates in Figure 3.3, initially every module and chip have its own reverse U boundary. We first packing the B*-tree to get the x-coordinate and y-coordinate of modules, and the Height and Width of the chip. Then, we place modules one by e one by the construction order of B*-tree and do the following steps to catch all the deadspace below the module and locate all the channels along the reverse U boundary of the module.

The steps of finding the deadspaces and channels:

Step 1. Place the module according to its x-coordinate and y-coordinate.

Step 2. Check if there is any up boundary which is overlapped with the bottom of the module. If the y-coordinate of the overlapped up boundary is equal to the bottom of the module, the overlapped segment is a channel. If the the y-coordinate of the overlapped up boundary is lower, the overlapped segment is a deadspace.

Step 3. Those overlapped segments of the up boundaries that are found in step 2 will be banished.

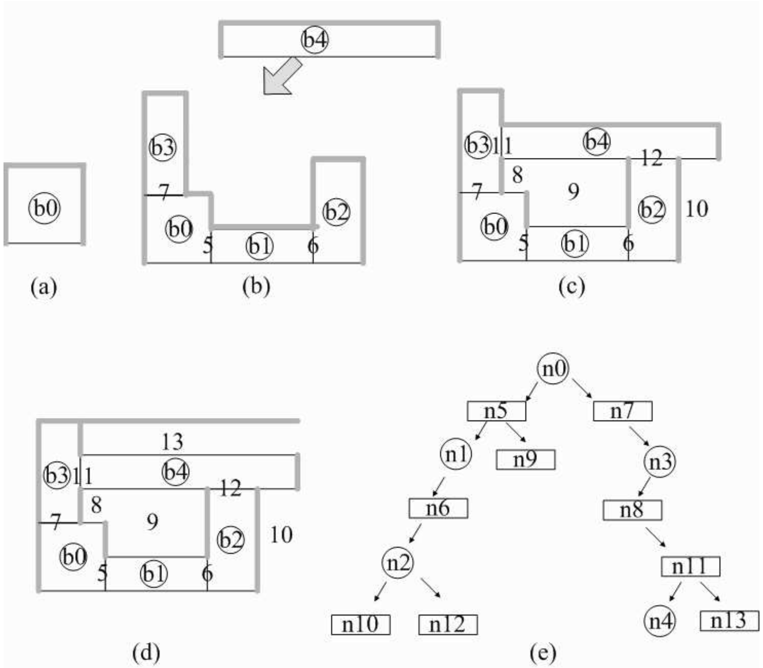


Figure 3.3: An example of buffer block retrieval with reverse u boundary. (a) Reverse u boundary of a module. (b) The reverse u boundaries before placing module b4. (c) After module b4 is placed, we got channel 11, 12, and deadspaces 8, 9, 10. (d)(e) The final result and its corresponding B*-tree.

Step 4. Check if there is any left (right) boundary which is overlapped with this module. The overlapped segment is a channel.

Step 5. Those overlapped segments of the left (right) boundaries that are found in step 2 will be banished.

Step 6. Place next module and repeat above steps.

3.2.2 Buffer Insertion with Fixed Buffer Block location

In the Section 2.4, we have described the soft module current model, and now we will apply the concept to decide the voltage of each buffer blocks. The voltage of buffer block will be the average value of the voltages of all the nodes which are overlapp with this buffer block. As Figure 3.4 illustrate, the voltage of the buffer

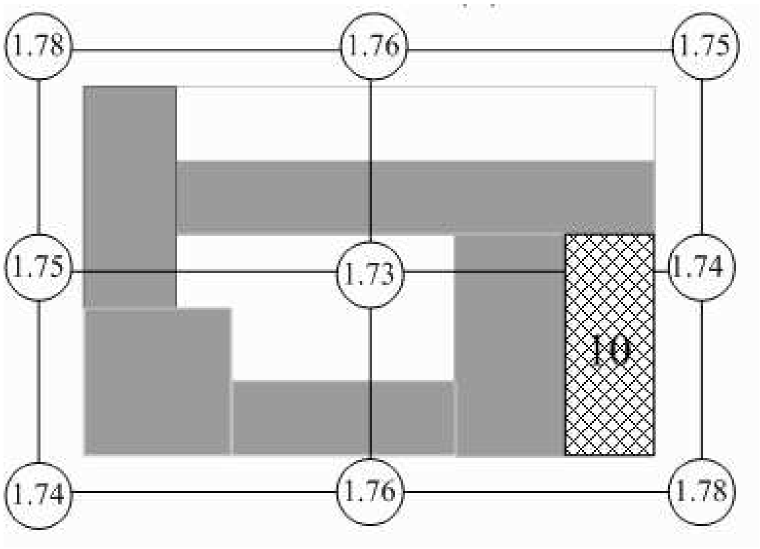


Figure 3.4: Deadspace(buffer block) 10 is overlapped with the right below two nodes. Assumed that the subarea of the deadspace that correspond to node with voltage 1.74 is 2/5 of its area, and that correspond to node with voltage 1.78 is 3/5 of its area, the voltage of deadspace 10 is 1.74*0.4+1.78*0.6 = 1.764V.

block is 1.764V. After this, we can get the distribution of the voltage drop of each buffer block that will effect our buffer insertion.

For the nets that needs only buffer to satisfy its constraint, we calculate the IFRs of each nets. The IFR of each net usualy have more than one buffer blocks which are overlapped with. For the consideration of area overhead minimization, we usually pick the buffer block which have more IFRs overlapped with than others to insert buffer. But to avoid causing IR-drop constraint violation, we cluster buffers on those buffer blocks which have less voltage drop. This will not only have the effect of minimizing area overhead but also ease the voltage drop. If there is any net that its IFRs have no buffer block to insert, then we will set the net as failed on timing constraint. When all the nets in this group is done with the buffer insertion, we move to deal with the other group.

3.3 Interconnect Pipelined for Minimum Latency

As described in Section 1, many global signal traveling across the chip need several clock cycles to reach their destinations, thus requiring the adoption of pipelined interconnects, i.e., latent wiring structures in which normal buffers are interleaved with memory elements such as latches and flip-flops. Unlike the buffer insertion, the insertion of flip-flop increases not only the area but also latency. The increment of latency will impact directly the system performance. Thus we need to deal with insertion of flip-flops carefully. We adopt the methodology described in Section 2.2 to minimize the latency of pipelined interconnect.

At the floorplanning stage, no routing information is available. We assumed that each net would be routed with a shortest path within the bounding box containing the two terminals. Because the position to insert buffer and flip-flop is limited by the buffer blocks (deadspaces and channels), so the routing path is limited by those as well. To find all the possible path and positions, we first find the buffer block bounded by the two terminals. And then we insert positions on those buffer block by equal distance. All the possible paths will be found by connecting positions by monotonic route. That is, position always connects to the next one which is on the shortest path between this position and the source. Figure 3.5 shows an example of this.

Because some paths are include by others, those paths are redundant. As the Figure 3.6 shows, there are three paths for the two positions. Because The solution space of path 1 and path 2 are included by path 3, the path 1 and path 2 are redundant. To avoid the redundant path, position always connect to the next such that there is no other position are bounded by this two positions.

The interconnect pipelined for minimum latency for an intermediate floorplan is a recursive bottom up approach. We start this approach from the sink. At the

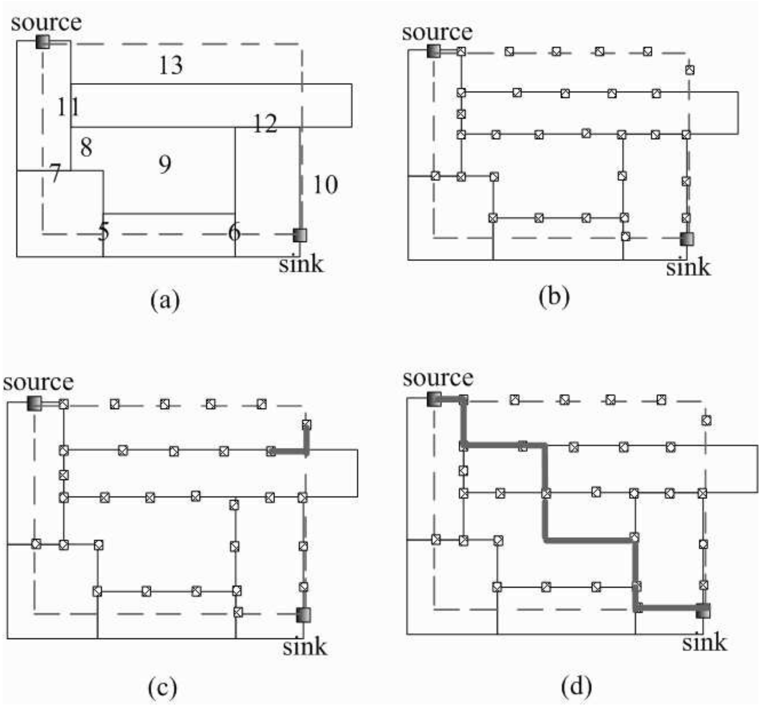


Figure 3.5: Example of positions and path. (a) The deadspaces and channels bounded by the two terminals. (b) Put positions in those buffer block (c) An example of non-monotonic route. (d) A possible path and positions on it.

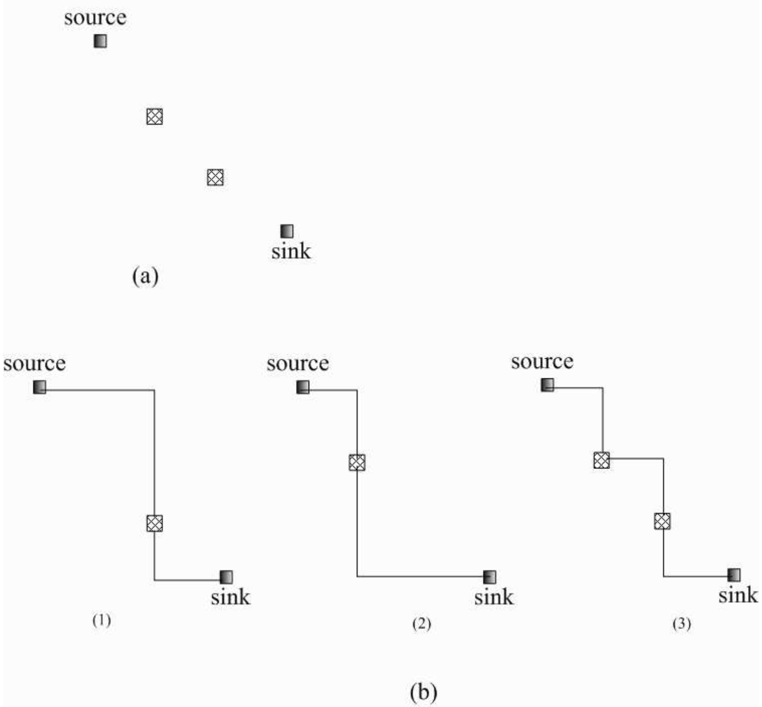


Figure 3.6: (a) Sink and source with two positions. (b) All the possible paths. Path (1) and (2) are redundant..

sink, we find the positions to connect to and use the Wire() operation that are described in Section 2.3 to propagate the solution to the connected positions. For those position received the solution, we do the Buffer() and FF() operation on it and keep finding the positions to propagate to. After the recursion stop, we will get all the possible paths and optimized solution with minimum latency. To minimize the area overhead, if a net have more than one path that have minimum latency and satisfy its timing constraint, we pick the path that its corresponding buffer blocks have much buffer and flip flop inserted. Figure 3.7 is an example, the upper path inserts a flip-flop in the deadspace 13 and the other path inserts a flip-flop in the deadspace 9. If deadspace 13 has more buffer or flip-flop inserted than that deadspace 9 has, we choose the upper path and insert a flip-flop in the deadpace 13; otherwise, we choose the below path and insert a flip-flop in the deadspace 9.

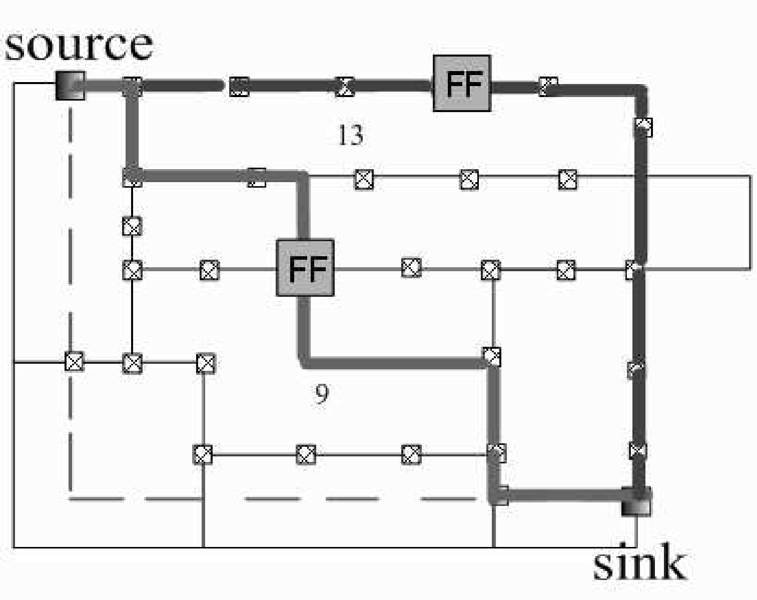


Figure 3.7: To minimize area overhead, we pick the path that its corresponding deadspace have much more buffer or flip flop inserted. If deadspace 13 have much buffer or flip flop inserted than deadspace 9, we choose the upper path; otherwise we choose the below path.

3.4 Packing and IR-Drop Re-Estimate

After the buffer insertion and pipelining interconnect for all nets are done, we then treat a buffer block as a soft module, insert the node into to the B*-tree. And we will do the IR-drop estimation again to check if there is any violation since we have additional buffers and flip-flops and the modules are slightly moved.

Chapter 4

Simultaneous Buffer / Flip-Flop Station Planning and Voltage Drop Minimization in Floorplan Design

In this section, we will present our algorithm for this problem. The algorithm is based on simulated annealing. After perturbing the floorplan, we invokes the procedure described in the preceding section.

4.1 Solution Perturbation

We represent a feasible nonslicing floorplan, without overlapping modules with the B*-tree representation. We adopt the following three operations to perturb a B*-tree to another.

- Op1: Rotate a module.
- Op2: Move a module to another place.
- Op3: Swap two modules.

4.2 Cost Function

As given in Section 2.5, the objective of our problem is to find a floorplan with planned buffer blocks such that all timing requirements and IR-drop constraints are satisfied and the area growth and system latency are minimized and system throughput is maximized. Therefore, a floorplan Γ is evaluated by its cost combined as follows.

$$cost(\Gamma) = area(\Gamma) + \alpha\frac{\mid t_n \mid}{normalized \mid t_n \mid} + \beta\frac{\sum_{\forall pv_i \in P_v} v_{pv_i}}{\sum_{\forall p_i \in P} V_{lim,p_i}} + \gamma\frac{\sum_{N_i \in N_{cn}} \lambda_{N_i}}{\mid N_{cn} \mid} + \omega\frac{\sum_{C_i \in C_{cc}} \tau_{C_i}}{\mid C_{cc} \mid}$$

where α, β, γ, and ω are user specified parameter, t_n is the number of net meet timing requirement, v_{pv_i} is the amount of voltage drop at the pin pv_i, P is the set of all P/G pins, P_v is the set of violating P/G pins, and V_{lim,p_i} is the allowed amount voltage drop of the P/G pin pi, N_{cn} is the most critical nets of latency, λ_{N_i} is the latency of net N_i, C_{cc} is the most critical cycles of throughput, and τ_{C_i} is the cycle mean of C_i.

The first part of cost is the area consumed by the floorplan, including existing buffer-block. The second part is the cost of the timing penalty paid for unsatisfied nets. The third part is the cost of the IR-drop violation. The fourth and fifth is the cost of system latency and system throughput respectively.

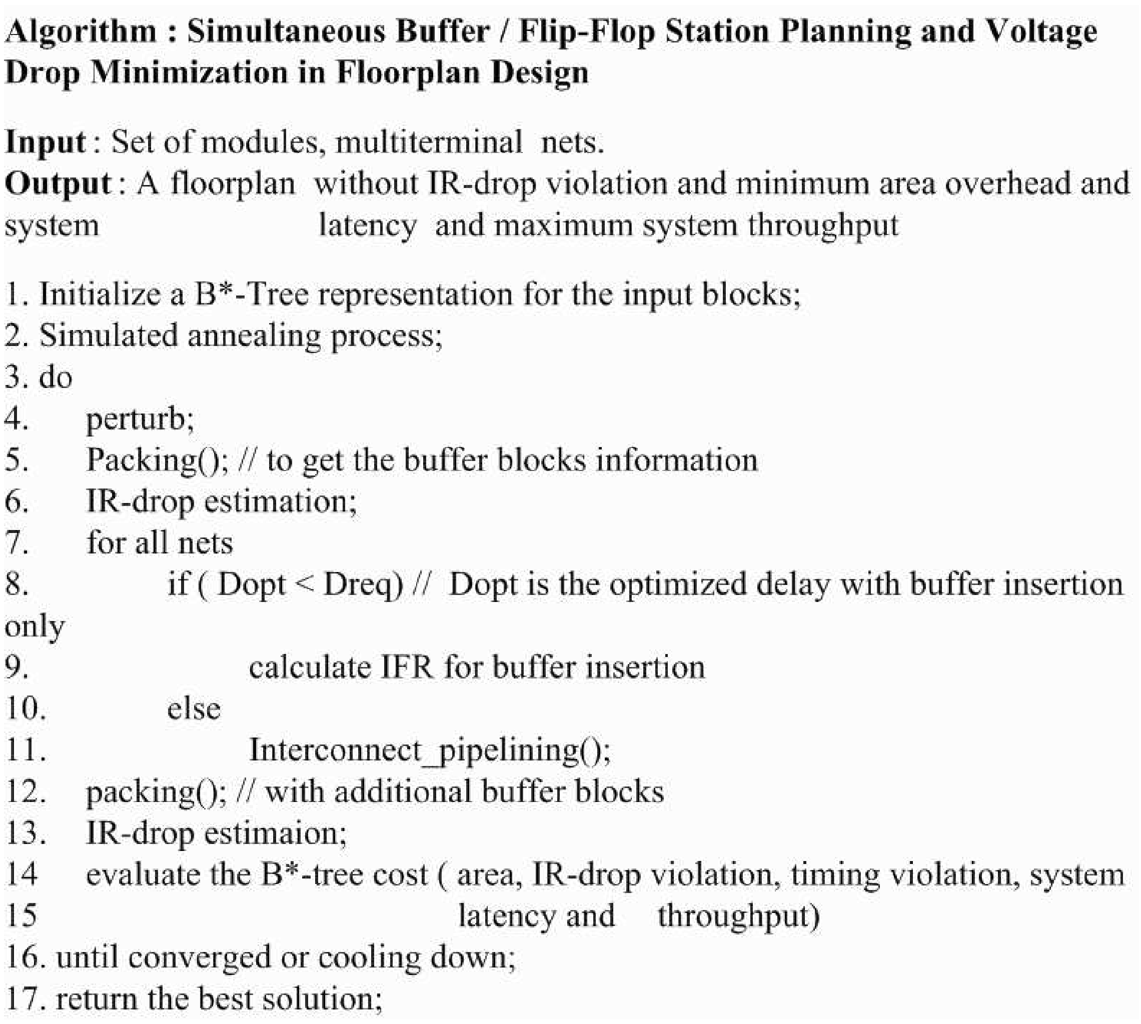
4.3 Annealing Schedule

The annealing schedule controls the acceptance rate of uphill move, neighboring solutions with higher cost. The initial temperature is set as $\Delta_{avg}/\ln(p)$, where Δ_{avg} is the average cost change of a random sequence of moves, and p is the initial probability of accepting uphill moves. In the beginning, the temperature is high; hence, p is initially set very close to 1. After each iteration, the temperature is reduced by a factor $\rho < 1$. The annealing process ends up when the temperature cools down below ε.

4.4 Overall Algorithm

The simulated annealing process begins from a random feasible Γ. We insert the buffer and flip-flop according to the method described in Section 3. Then we perturbs the floorplan using the three perturbations. After each move, buffers and flip-flops are planned according to the new floorplan. The process terminates when the solution is frozen, the temperature is too low, or the runtime is too long.

The flow of our algorithm is summarized in Figure 4.1. In line 1, we first get a initial floorplan by random assign the B*-tree. In lines 3-16, we perturbs the floorplan from one to another until the solution is converged or cool down.



Algorithm : Simultaneous Buffer / Flip-Flop Station Planning and Voltage Drop Minimization in Floorplan Design

Input : Set of modules, multiterminal nets.
Output : A floorplan without IR-drop violation and minimum area overhead and system latency and maximum system throughput

```
Initialize a B*-Tree representation for the input blocks;
Simulated annealing process;
do
    perturb;
    Packing(); // to get the buffer blocks information
    IR-drop estimation;
    for all nets
        if ( Dopt < Dreq) //  Dopt is the optimized delay with buffer insertion only
            calculate IFR for buffer insertion
        else
            Interconnect_pipelining();
    packing(); // with additional buffer blocks
    IR-drop estimaion;
    evaluate the B*-tree cost ( area, IR-drop violation, timing violation, system
                                        latency and    throughput)
until converged or cooling down;
return the best solution;
```

Figure 4.1: simultaneous buffer / flip-flop station planning and voltage drop.

Chapter 5

Experimental Results

We implemented our approach in the C++ Programming language and the platform is AMD Opteron (tm) 2.8G with 2.0GB memory. We experiment with our approach on MCNC[1] circuit banchmark. Table 5.1 lists the technology file and buffer library used in our experiments that are based on 0.18-μm in the NTRS'97 roadma[2]. The intrinsic delay and input capacitance of a flip flop is 10% that buffer has. Our IR-drop constraint is 5% of the supply voltage. Thus the IR-drop constraints are $V_{min} = 1.71$ for the power and $V_{max} = 0.9$ for the ground. We give each circuit two power pads and randomly assigned the peak current on each P/G pin of the modules as [15] did. The current of buffer and flip flop are assigned as the proportion of its area to the smallest module area in each circuits. The vertical and horizontal power wire pitches are both 600μm.

The first experiment compare the results of planning buffer block without IR drop consideration and our methodology that insert buffer block and also consider IR-drop. In this experiment, the two-terminal nets obtained by splitting from multiterminal nets and the timing requirement of each net are generated by [9] from 1.05-1.20D_{opt} as the [11] did. The experimental result are summarized in Table 5.2 . The first column shows the circuit name and the algorithm used. The second column shows the number of nets meeting timing requirements (# nets meet) and

Table 5.1: Parameter of 0.18-μm Technology in the NRTS'97 RoadMap[11].

Parameter	Description (unit)	Value
r	wire sheet resistance ($\Omega/\Box$)	0.068
r_w	wire unit-length resistance of 0.9 μm width (Ω/μm)	0.075
ω	wire width (μm)	0.9
c_w	wire unit-length capacitance of 0.9 μm width (Ω/μm)	0.118
C^L	load capacitance (fF)	23.4
R^D	driver resistance (Ω)	180
D_b	intrinsic buffer delay (ps)	36.4
C_b	buffer input capacitance (fF)	23.4
R_b	buffer output resistance (Ω)	180
A_b	buffer size (μm^2)	400

that of total nets in a circuit (Tot. # nets). The third column gives the percentages of nets meeting the timing constraints. Column4 lists the number of buffers inserted (# buffers). Column 5 gives the worst voltage of modules in each circuit. Column 6 gives the percentages of extra areas over the given floorplans for buffer insertion.

The result shows that our methodology have almost the equal result on the timing requirement and the area overhead and also does not violate any IR-drop constraint.

In the second experiment, unlike the first experiment, the timing constraint for every net are given the same constraint to reflect that some nets need pipelining . Because the ami33 circuit is much smaller than others, its timing constraint is given as half value of other circuits. We compare our methodology "simultaneous buffer / flip-flop station planning and voltage drop minimization in floorplan design" called method A (M. A) to the case that does not consider IR-drop called method B (M. B) and the case that does not consider latency and throughput called method (M. C). The experimental result are summarized in Table 5.3. Column 4 lists the number of nets that only needs buffer (B. net). Column 5 lists the number of nest that are pipelined (FF net). Column 6 lists the number of buffer inserted (# B). Column

Table 5.2: Compare the result of planning buffer block and the result of planning buffer block considering IR-drop, where column 2 is the number of nets meet timing requirement / number of total nets, column 3 is the percentage representation of column 2, column 4 is the number of buffer inserted, column 5 is the smallest voltage of the P/G pin among the modules in each circuit, column 6 is the area overhead caused by buffer inserted.

Circuit & Algorithm	#nets meet / Tot. #nets	nets meet timing (%)	# buffers	worst voltage	Extra area (%)
apte					
BBP	122 / 172	70.9	231	1.658	0.89
ours	118 / 172	68.6	236	1.722	1.13
xerox					
BBP	373 / 455	81.9	452	1.678	0.00
ours	376 / 455	82.6	445	1.724	0.00
hp					
BBP	192 / 226	84.9	111	1.704	0.00
ours	189 / 226	83.6	117	1.712	0.00
ami33					
BBP	340 / 363	93.6	377	1.662	0.00
ours	341 / 363	93.9	358	1.714	0.00
ami49					
BBP	484 / 545	88.8	523	1.681	0.00
outs	494 / 545	90.6	531	1.721	0.00
Summary					
BBP		84.0	338	1.676	0.17
ours		83.8	337	1.718	0.22

7 lists the number of flip-flops inserted (# FF). Column 8, 9 , respectively, are the worst latency and system throughput.

From the results of M. A, M. B, and M. C, it shows that our methodology can find a path with minimum latency if there exist any one (each percentage of nets meet timing requirement is high than 98%). The results of M. A and M. B shows that M. A does not have any IR-drop violation thought M. B have less area overhead. The IR-drop violation and area overhead is a tradeoff between M. A and M. B. The Results M. C have almost equal overhead to M. A but have less system throughput and higher latency.

Table 5.3: Compare the result of planning buffer / flip flop considering IR drop (M. A) and without IR-drop consideration (M. B) and without latency and throughput consideration (M. C), where column 4 is the number of nets that need only buffer to satisfy its timing requirement, column 5 is the number of the nets have to be pipelined, column 6 is the number of buffer inserted, column 7 is the number of flip flop inserted, column 8 is the largest latency of nets, column 9 is the largest throughput of cycles.

Cir. & Algo.	#N. meet / Tot. #N.	meet (%)	#B. nets	#F.F. nets	# B.	# F.F.	l	Throu.	worst vol.	Extra area (%)
apte										
M. A	171 / 172	99.4	1	169	216	308	4	0.25	1.71	6.0
M. B	171 / 172	98.8	10	131	191	291	3	0.33	1.67	4.0
M. C	170 / 172	98.8	4	161	191	291	4	0.23	1.72	3.6
xerox										
M. A	454 / 455	99.7	13	313	175	361	2	0.5	1.72	2.4
M. B	450 / 455	98.9	17	314	189	353	2	0.5	1.68	2.8
M. C	450 / 455	98.9	17	314	156	337	2	0.5	1.71	2.7
hp										
M. A	225 / 226	99.5	6	127	47	128	2	0.5	1.71	2.4
M. B	224 / 226	99.1	13	127	48	127	1	0.5	1.70	2.
M. C	225 / 226	99.5	6	128	47	128	2	0.5	1.71	2.4
ami33										
M. A	363 / 363	100	0	135	0	135	1	0.75	1.71	0.0
M. B	363 / 363	100	0	154	0	154	1	0.75	1.70	0.0
M. C	363 / 363	100	0	155	0	155	1	0.6	1.73	0.0
ami49										
M. A	543 / 545	99.6	16	463	389	621	3	0.42	1.72	7
M. B	543 / 545	99.6	21	424	392	622	3	0.42	1.68	4
M. C	543 / 545	99.6	22	397	387	599	3	0.37	1.73	11
Summary										
M. A		99.6					2.4	0.484	1.714	3.7
M. B		99.2					2.0	0.5	1.686	2.56
M. C		99.3					2.4	0.44	1.72	3.94

Chapter 6

Conclusion

In this thesis, we propose a methodology to pipeline interconnect in floorplan to estimate the system latency and throughput to avoid extreme high latency global net. Also we consider the IR drop during the planning of buffers and flip flops. The experimental results shows that our methodology is effective. As the size of chip getting larger, and size of buffer getting smaller, we expect the methodology will become more important in the future.

Bibliography

[1] “www.cse.ucsc.edu/research/surf/gsrc/mcncbench.html”.

[2] “National Technoloogy Roadmap for Semiconductors”. ed: Semiconductor Industry Assoc., 1997.

[3] C. J. Alpert and A. Devgan. “Wire segmenting for improved buffer insertion”. In *Proc. of DAC*, pages 588–593. June 1997.

[4] Y. C. Chang, Y. W. Chang, G. M. Wu, and S. W. Wu. “B*-Trees : A new representation for non-slicing floorplans”. In *Proc. of DAC*, pages 458–463. April 2000.

[5] Y. H. Cheng and Y. W. Chang. “Integrating buffer planning with floorplanning for simultaneous multi-objective optimization”. In *Proc. of ASPDAC*, pages 431–434. January 2003.

[6] P. Cocchini. “A methodology for optimal repeater insertion in pipelined interconnects”. In *IEEE TCAD*, volume 32, No 12, pages 1613–1624. Dec 2003.

[7] J. Cong. “Challenges and opportunities for design innovations in nanometer technologies”. In *SRC Design Sciences Concept Paper*. San Jose, CA: Semiconductor Research Corp., 1997.

[8] J. Cong. “Post-Placement Voltage Island Generation”. In *An interconnect-centric design flow for nanometer technologies*, volume 89, pages 505–528. Apr 2001.

[9] J. Cong, T. Kong, and D. Z. Pan. “Buffer block planning for interconnect-driven floorplanning”. In *Proc. of ICCAD*, pages 358–363. Nov 1999.

[10] A. Jahanian and M. Saheb Zamani. “Multi-level buffer block planning and buffer insertion for large design circuits”. In *International Symposium on Very Large Scale Integrated Circuits*, pages 411–415. 2005.

[11] Iris H. R. Jiang, Y. W. Chang, J. Y. Jou, and K. Y. Chao. “Simultaneous floorplan and buffer-block optimization”. In *IEEE TCAD*, volume 23, No.5. May 2004.

[12] J. Lillis, C. K. Cheng, and T. T. Y. Lin. “Optimal wire sizing and buffer insertion for low power and a generalized delay model”. In *Proc. ICCAD*, pages 138–143. Nov 1995.

[13] S. Lin and N. Chang. “Challenges in power-ground integrity”. In *Proc. of ICCD*, pages 651–654. 2006.

[14] V. Litovski and M. Zwolinski. “VLSI Circuit Simulation and Optimization”. Champman & Hall, 1997.

[15] C. W. Liu and Y. W. Chang. “Floorplan and power/ground network co-synthesis for fast design convergence”. In *Proc. of ISPD*, pages 86–93. 2006.

[16] T. Okamoto and J. Cong. “Buffered Steiner tree construction with wire sizing for low power and a generalized elay model”. In *Proc. of ICCAD*, pages 44–49. Nov 1996.

[17] P. Sarkar, V. Sundararaman, and C. K. Koh. “Routability-driven repeater block planning for interconnct-centric floorplanning”. In *Proc. of ISPD*, pages 186–191. Apr 2000.

[18] X. Tang and D. F. Wong. “Planning buffer locations by network flows”. In *Proc. of ISPD*, pages 180–185. Apr 2000.

[19] L. P. P. P. van Ginneken. “Buffer placement in distributed RC-tree net works for minimal Elemore delay”. In *IEEE ISCS*, pages 865–868. 1990.

[20] L. Wang. “Throughput-Aware Floorplanning by Dynamically Considering Multiple Critical Cycles”. Master Thesis, Department of Electronics Engineering, National Chiao Tung University, 2007.

[21] G. M. Wu, Y. C. Chang, and Y. W. Chang. “Rectilinear Block Placement Using B*-Trees”. In *ACM TODAE*. April 2003.

[22] J. S. Yim, S. O. Bae, and C. M. Kyung. “A floorplan-based planning methodology for power and clock distribution in ASICS”. In *Proc. of DAC*, pages 766–771. 1999.

作者簡歷

潘信華，民國七十一年七月出生於台北市。民國九十四年六月畢業於輔仁大學電子工程學系，並於同年九月進入國立交通大學電子研究所就讀，從事 VLSI 實體設計方面相關研究。民國九十七年一月取得碩士學位，碩士論文題目為『在佈局階段同時對緩衝器與正反器做放置規畫以及電壓下降的最小化』。