INTEGRATION the VLSI journal 58 (2017) 1–8

Contents lists available at ScienceDirect

INTEGRATION, the VLSI journal

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

A routing framework for technology migration with bump encroachment

Po-Yi Wu[a], Wai-Kei Mak[a], Ting-Chi Wang[a], Cheng Zhuo[b,*], Kassan Unda[c], Yiyu Shi[c]

[a] *Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan, ROC*
[b] *College of Information Science & Electronic Engineering, Zhejiang University, Hangzhou, China*
[c] *Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN, USA*

ARTICLE INFO

Keywords:
Redistribution layer routing
Technology migration
Power/ground domain
Bump encroachment

ABSTRACT

Technology migration plays a critical role in the time-to-market competition. Most existing works focus on layout compaction or hardware description language re-synthesis, and pay little attention to the I/O interface in flip chips. The complication of bumping process as well as electrical and reliability considerations prevent the bumps from scaling with transistor sizes. On the other hand, the number of signal bumps cannot be reduced and sometimes even increases due to the demands for wider bandwidth and various peripheral devices. As a result, the allocated die area for I/O can no longer afford the number of bumps a chip requires. This issue, known as bump encroachment, puts a stringent requirement on the redistribution layer (RDL) routing. In this paper, we first formulate the problem of RDL routing with bump encroachment, and then propose a network flow based algorithm to efficiently address it. Experimental results on a few benchmarks with parameters extracted from industrial designs show that compared with a maze routing-based approach, our algorithm can achieve up to 72% wire length reduction.

1. Introduction

In the highly competitive semiconductor industry, time-to-market plays a vital role in product revenue. For every new generation of technology process, it has become a common practice to reuse the IPs from earlier technology nodes for the first design in the new process. Meanwhile the microarchitecture is maintained the same. Such practice is commonly referred to as technology migration [1]. After that, with better understanding of the new process for higher yield, follow-up designs may come with new microarchitecture with lower cost and higher performance. One well-known example is Intel's Tick-Tock model [2].

Due to its criticality, many existing works have studied the automated flows for technology migration [1,3–8]. However, the prior works are either focusing on layout compaction, or hardware description language re-synthesis (details will be discussed in Section 2.1). Very little attention has been placed on the reuse of I/O interface., i.e., the bumps in flip-chip designs as shown in Fig. 1.

The bumping process typically includes seed layer deposition, bump plating and seed etch [9]. Seed etch is often a wet etch process, during which etchants will remove the Cu seed layer and the adhesion promoter such as Ti. Seed etch becomes very challenging when the bump size scales below ~25 μm in diameter due to under-cut, i.e. the Cu seed layer under the bump (or even the bottom part of the bump) is smaller than the bump. A small under-cut will reduce the contact area between the bump and the substrate and will influence its mechanical and electrical performance. Moreover, electrical and reliability considerations, like current carrying capacity, also limit the size of bumps. Thus, bump size cannot keep pace with transistor size shrinking in technology scaling [10]. On the other hand, the number of signal bumps for I/Os are gradually increasing due to the growing demands on wider bandwidth and various peripheral devices.

As a result, the allocated die area for I/O can no longer maintain the desired scaling ratio in order to afford the number of required bumps. For example, for a 32 nm CORE™ chip, the north bridge size is around 10 mm*1 mm. The common packaging technology has bump pitch around 150–200 μm, with 300–400 signal bumps placed on top of the north bridge. According to the Moore's Law, for 22 nm chip the north bridge area is desired to be halved and the size needs to be scaled to 7 mm*0.7 mm. In order to maintain the same number of bumps, the bump pitch and size need to be around 105–140 μm, which is very challenging for the commercial packaging technology without incurring significant additional overhead. Moreover, if we assume bumps remain similar size, then, in order to fit the reduced die size, the number of bumps has to be reduced by 38% by Year 2020 for high performance logic designs, as shown in Table 1 [11]. In general, the reduced number of bumps may either lead to technology migration failure or compromise of I/O bandwidth.

* Corresponding author.
E-mail address: czhuo@zju.edu.cn (C. Zhuo).

http://dx.doi.org/10.1016/j.vlsi.2017.01.003
Received 30 June 2016; Received in revised form 4 December 2016; Accepted 3 January 2017
Available online 07 February 2017
0167-9260/ © 2017 Published by Elsevier B.V.

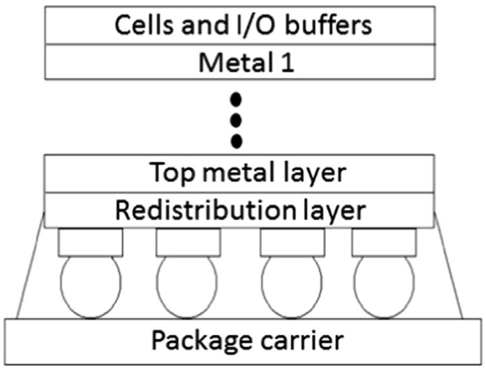


Fig. 1. Flip-chip design with solder balls (bumps).

Table 1
Effective channel length and normalized bump number for high performance logic designs till 2020 (source: International Technology Roadmap for Semiconductors [11]).

Year	2015	2016	2017	2018	2019	2020
L_{eff} (nm)	17	15.3	14.0	12.8	11.7	10.6
#Bumps (normalized)	1	0.90	0.83	0.75	0.69	0.62

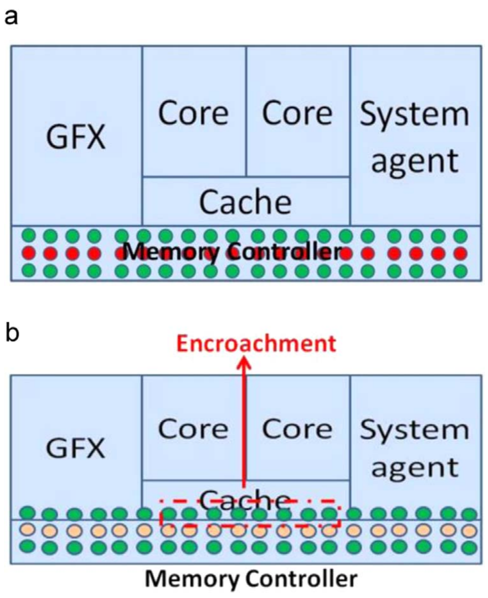


Fig. 2. Example plots of bump encroachment in technology migration. (a) Bump map of memory controller in last technology node (e.g. 22 nm). (b) Bump map of memory controller in current technology node (e.g. 14 nm) where bumps of memory controller encroaches cache region.

In order to minimize the impact on scaling and enable the migration for sub-22 nm chips, one new approach, used in industry for both microprocessors and SoCs [12,13], is to allow some bumps for I/O to be placed on top of non-I/O regions (for example, core and cache) which is called **bump encroachment**, and reassign some power and ground bumps to signals to trade power integrity for design migratability. An illustrative plots of bump encroachment during technology migration is shown in Fig. 2, which shows the memory controller bumps do not scale as much as the die and hence have to encroaches the cache area in order to maintain a similar number of bumps. Although this approach provides a solution to bump scaling during technology migration, it also brings several additional challenges for routing in the redistribution layer (RDL):

- What is the best way to choose and reassign some of the power/ground bumps as signal I/Os?
- How to connect signal traces to their respective bumps in RDL with such change?
- How to ensure each portion of I/O are still powered?

To the best of authors' knowledge, these problems have not been well addressed or studied in the prior works.

This problem falls into the category of routing in the redistribution layer (RDL), a dedicated extra metal layer redistributing and connecting the I/O pads to the bumps. With details discussed in Section 2.2 about the prior works of arts on RDL routing, our problem is fundamentally different:

- Our work is designed for the scenario of bump encroachment, which is a relatively new technology used in a few sub-22 nm chips to address the issues of bump scaling. This technology is believed to be more widely used in future sub-14 nm designs due to the incapability of bump scaling and design cost concerns. However, the routing challenges with bump encroachment have not been publicly discussed yet.
- Prior RDL routing works only consider I/O routing or I/Os and P/G routing separately. In practice, such approaches may introduce unnecessary blockage for the I/O or P/G routing on the redistribution layer and enforce power supply transmission through the lower stacks, which jeopardizes power integrity. To resolve the aforementioned issues, the proposed work considers I/O and P/G routing simultaneously.
- Existing RDL routing works only consider the connection between I/O pads to bumps, without much discussion on the routing trace connection. In practice, RDL is typically designed after bump freezing and IP designs. Thus, for an IP, there are signal traces and P/G traces on the lower stacks, both of which need to be connected to the bumps on RDL. This is challenging but rarely mentioned in the previous works. The proposed work discusses a solution to tackle this challenge.
- Typically the bump patterns in the previous works are assumed to be rectangular or chessboard manner. However, in recent commercial chips [12,13], bumps are placed in a staggered way to facilitate routing. The reasons behind the use of a staggered bump pattern (instead of rectangular pattern) are: (a) within the same unit area and design requirements, the staggered bump pattern allows more bumps compared to the rectangular pattern; (b) by staggering, the power and signal bumps may see more ground bumps around, in other words, more return paths and smaller inductance, which is in favor of power and signal integrity of the design.

The main contributions of this paper are two folds: This is the first work to discuss and formulate the problem of bump encroachment in technology migration; and an efficient 3-stage algorithm is proposed to address the problem.

Experimental results on a set of benchmarks with parameters extracted from industrial designs show that compared with a maze routing-based approach, our algorithm can achieve up to 72% wirelength reduction.

The remainder of this paper is organized as follows. Section 2 shows the background of technology migration as well as our motivation. We formally state and formulate the problem in Section 3. We describe our approach to solve it in Section 4. Experimental results are shown in Section 5 and concluding remarks are given in Section 6.

2. Background

2.1. Technology migration

Depending on how the technology migration is done, the reuse of a design can be either hard or soft. Hard-design reuse directly converts a chip's physical layout into the new technology. Billions of polygons

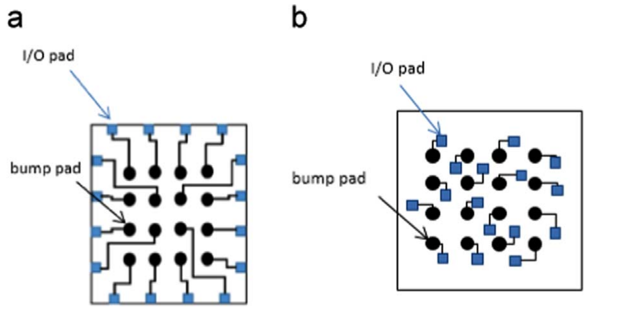


Fig. 3. (a) RDL routing for a peripheral-I/O flip-chip (b) RDL routing for an area-I/O flip-chip.

have to be modified in order to satisfy the design rules of the target technology, which is a very challenging task. Several studies have been performed for hard-design reuse [1,5–8].

On the other hand, soft-design reuse is based on the hardware description language, which will be resynthesized in the target technology [3,4]. Compared with the hard-design reuse, it has the benefits of flexibility and portability.

However, it cannot guarantee power and timing closure, since all the physical design effort invested in the previous design will be lost. This is especially an issue for high performance or low power designs, where the engineering efforts in physical design are considerable.

2.2. RDL routing

Two types of flip-chip structures, peripheral I/O and area I/O, are commonly used. The peripheral I/O structure has I/O pads distributed around boundaries, while the area I/O structure has I/O pads distributed internally. See Fig. 3 for an illustration of the two types of flip-chip structures, where bumps are arranged as a rectangular array.

Flip-chip RDL routing has been studied in the literature and can be categorized into three classes: the free-assignment problem, the pre-assignment problem, and the unified-assignment problem [14,15].

The free-assignment problem has no pre-defined net assignments between I/O pads and bumps, so an RDL router has the freedom to connect each I/O pad to a bump as long as the bump is not connected with another net. Routing algorithms based on network flow and Voronoi diagram have been proposed for the free-assignment problem on both flip-chip structures [16,17,21,22]. In contrast, the pre-assignment problem has pre-defined net assignments between I/O pads and bumps, and therefore it is more restricted than the free-assignment problem in terms of solution space. Integer linear programming, dynamic programming, and greedy method have been adopted to develop routing algorithms for the pre-assignment problem on the peripheral I/O structure [18,19,20,21,23].

The unified-assignment problem is a hybrid version of the free-assignment and pre-assignment problems, in which some I/O pads have pre-defined net assignments while others do not. A method utilizing Delaunay triangulation and Voronoi diagram for the area-IO unified-assignment problem is proposed in [22]. As shall be seen in Section 3, the RDL routing problem addressed in this paper is different from any of the prior works and therefore no existing algorithm can be directly applied to solve our problem.

3. Problem description

In this section, we define our RDL routing problem for technology migration with bump encroachment. We consider block-based SoC design with multiple power domains. The routing region of each block is defined after bump encroachment. We assume that the top metal layer (see Fig. 1) consists of traces for different power/ground (P/G) domains, as well as traces for I/O signals as in Fig. 4. All the traces in the top metal layer are horizontal. The traces within each routing region have to be connected to bumps located above the routing region through the redistribution layer where the redistribution layer allows Manhattan routing. Each signal trace is required to be connected to a separate bump.

To minimize the maximum noise in the P/G domains, the number of power/ground bumps for each P/G domain within a routing region is pre-determined on the basis of the ratio of their load capacitances. For example, assume we have 30 bumps and 3 P/G domains whose load capacitances are 10nF (C), 20nF (2C) and 30nF (3C) respectively. So, we allocate 5 bumps to first P/G domain, 10 to the second one and 15 to the third power grid. This effectively scales the resistance as the resistance is inversely proportional to the number of power/ground bumps allocated to that P/G domain.

Given a layout of signal traces and P/G traces of various P/G domains in the top metal layer, the objective of RDL routing is to connect all signal traces and P/G traces to bumps through the redistribution layer with minimum total wire length subject to the pre-determined bump budget for each P/G domain. We assume that the supply of bumps is equal to the bump demand by all signal traces and P/G domains.

For example, Fig. 4(a) shows an industrial block-based SoC design where one of the routing regions (routing region A) has two power domains (red and blue) and a ground domain (green). There are 6 signal traces (black) and 10 bumps in the routing region. The pre-determined numbers of P/G bumps for the red, blue, and green P/G domains are 1, 1, and 2, respectively. Note that Fig. 4(b) is a simplified illustration, in the top metal layer there are typically scores of tracks between two adjacent bump rows. Except for the short signal traces, the top metal layer is fully occupied by long interleaving P/G traces of different P/G domains.

Now the differences between our problem formulation and those in the existing RDL routing works discussed in Section 2.2 are pretty clear. Existing works assume that there is a given set of I/O pads on the redistribution layer to be connected to bumps. Here we are given a distribution of signal traces and P/G traces in the top metal layer. The number of P/G traces is significantly more than the number of signal traces. Both the P/G traces and signal traces need to be connected to bumps. Besides, we assume that the bumps are arranged in a staggered

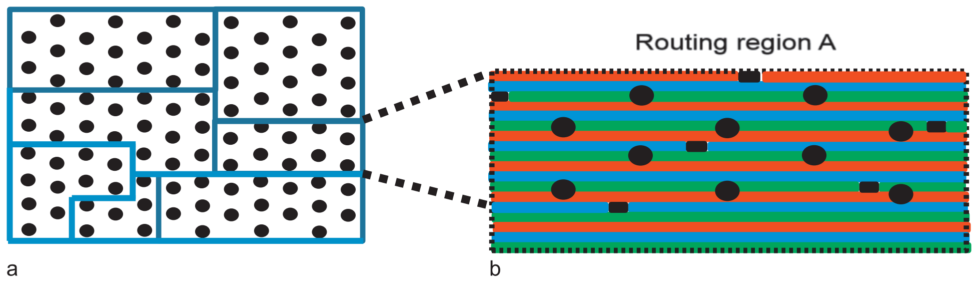


Fig. 4. (a) A block-based SoC design. (b) Signal traces and P/G traces in the top metal layer of routing region *A*. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

fashion, similar to the recent package design practices in industry (for example, Intel 32 nm and 22 nm microprocessors).

4. Algorithm

A major distinction of our problem from all previous works is that we need to connect all P/G traces belonging to the same P/G domain in a routing region to a few bumps and each P/G domain can have up to hundreds or thousands of traces distributed all over the region. Unless these P/G traces are routed carefully, it is highly likely that their routes will obstruct the routing of the signal traces to the remaining bumps. Here we propose an efficient approach to handle such large number of traces in a routing region.

We note that since all the traces within a routing region have to be connected to the bumps located over the region, we may process each region independently. Our RDL routing algorithm consists of three main steps. Initially, we connect the majority of the P/G traces of each P/G domain using a P/G trunk in such a way that it is guaranteed not to hinder the routing of the signal traces. Next, we simultaneously route the P/G trunks and the signal traces to the bumps using network flow to minimize the RDL wirelength. Finally, we perform a simple post-processing step to connect any orphan P/G traces to its own P/G domain. Below we first describe our routing model, then we present the details of each step of our RDL routing algorithm.

4.1. Preliminaries and notation

To facilitate RDL routing, we impose a routing grid onto the routing region such that the grid pitch corresponds to the redistribution layer's wire pitch. Subsequently, the routing on the redistribution layer is assumed to follow the grid segments. See Fig. 5(a) for an example (the P/G traces of the routing region are not shown in Fig. 5 for better visualization). Without loss of generality, we assume that all the signal traces and P/G traces run horizontally in the rest of the paper and all bump pads are located at grid points.

Recall that all the signal traces and P/G traces reside on the top metal layer which is above the redistribution layer as in Fig. 1. We note that the pitch on the top metal layer are typically 5–10 times smaller than that on the redistribution layer, so the signal traces and P/G traces on the top metal layer will not be aligned with the imposed RDL routing grid. Besides, the horizontal signal traces on the top metal layer are sparse and very short. As shown in Fig. 5(a), typically a signal trace passes over only one or two vertical RDL routing tracks and a via can be inserted at that one or two locations for the signal trace on the top metal layer to access the RDL routing grid. Without loss of generality, we assume that one via location has been picked for each signal trace as in Fig. 5(b). On the other hand, the P/G traces on the top metal layer are dense and long with the majority of the P/G traces spanning from one end of a routing region to the other end of the routing region horizontally (see Fig. 4(b)). As a result, a P/G trace passes over many vertical RDL routing tracks and has many candidate locations to drop a via to access the RDL routing grid. The following notation is used in the rest of the section. There are K different P/G domains which are denoted by α_k $(k = 1,2,\ldots,K)$. There are N_S signal traces in the top metal layer. The total number of available bumps is N and the bump budget allocated to P/G domain α_k is N_k $(k = 1,2,\ldots,K)$.

4.2. P/G trunk insertion

In the first step of our RDL routing algorithm, we connect the majority of P/G traces in each domain using a P/G trunk in the RDL routing grid in such a way that it is guaranteed not to hinder the routing of the signal traces. Recall that there are over hundreds of P/G traces for each P/G domain and most of them span from the left end of the routing region all the way to the right end of the routing region. This motivates us to stitch together the P/G traces belonging to the same P/G domain by a *P/G trunk* that runs from the top to the bottom of the routing region. But there will still be some P/G traces not crossed by the P/G trunk of its domain, especially when a routing region is rectilinear but not rectangular, these P/G traces will be connected by the post-processing step described in Section 4.4.

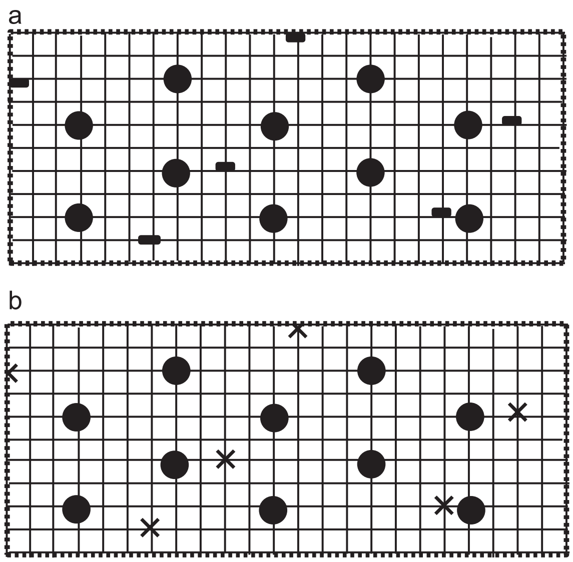


Fig. 5. (a) RDL routing grid imposed onto routing region A. (Note that the signal traces are located on the top metal layer above the RDL and the bumps are located below the RDL.) (b) Via locations for signal traces.

It should be noted that even if we decide to construct P/G trunks running from the top to the bottom of the routing region, there is a huge number of ways to do that. There is an enormous number of combinations of K P/G trunks for K P/G domains, some of which are feasible for routing completion and some are not. For example, the combination of the three P/G trunks shown in Fig. 6 is infeasible, since there is no way to route the rightmost signal trace to any bump. Intuitively, we desire the following properties for the constructed P/G trunks. First, we want a P/G trunk to be short so that it consumes less routing resources. Second, we want a P/G trunk to be close to some bumps that it may be connected to some bumps easily. Third, the P/G trunks of different P/G domains should not cross one another. Fourth, the K P/G trunks for K P/G domains must be distributed in such a way that allows routing completion, i.e., each P/G trunk τ_k will be able to connect to N_k distinct bumps while at the same time the N_S signal traces will be able to connect to N_S distinct bumps. Here we propose to construct P/G trunks in the shape of a rightward dogleg immediately above some bump as in Fig. 7 which satisfy the first two properties. Moreover, we describe an effective procedure to insert K rightward dogleg P/G trunks that satisfy the last two properties. In particular, our procedure always connects each rightward dogleg P/G trunk in the solution to some bump(s) on its left.

The procedure for P/G trunk insertion is shown in Fig. 8 and we explain its details below. We process the bump columns from left to right. Suppose P/G trunks have already been successfully inserted for P/G domains α_1 to α_{d-1} when we come to bump column c. We define the active signals as the signals to the right of the P/G trunk of domain α_{d-1} but to the left of bump column c. We define the active bumps as the bumps to the right of the P/G trunk of domain α_{d-1} but to the left of or on bump column c. Let ρ be the largest integer such that the number of active signals plus $\sum_{k=d}^{d+\rho-1} N_k$ is no more than the number of active bumps (let ρ be 0 if the number of active signals plus N_d is more than the number of active bumps). Then, an upper bound[1] on the

[1] In the unlikely case that the number of P/G domains is more than the number of vertical routing tracks W between two adjacent bump columns, it should be checked that ρ plus the number of P/G trunks inserted across the bump column preceding c does not

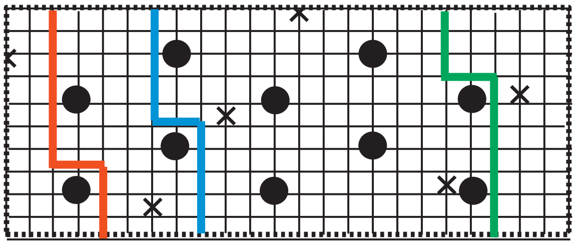


Fig. 6. An infeasible set of P/G trunks for routing region *A*.

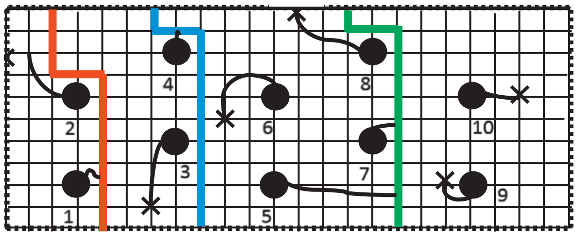


Fig. 7. A feasible set of P/G trunks for routing region *A* that will allow every signal and P/G trunk to connect to bumps.

```
d=1; /*initialization*/
for (each bump column $c$ from left to right)
  compute $\rho$;
  i=0;
  for (each bump $b$ from bottom to top)
    if (rightward dogleg trunk candidate $\tau_b$
induced by $b$ is feasible)
      insert $\tau_b$ for domain $\alpha_{d+i}$;
      i++;
  d=d+i;
```

Fig. 8. Procedure for P/G trunk insertion.

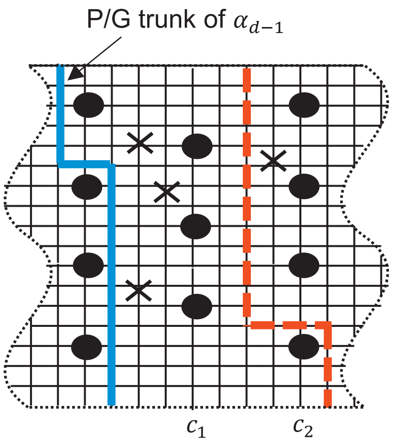


Fig. 9. Determination of rightward dogleg P/G trunk candidate for domain α_d.

number of rightward dogleg P/G trunks that we may actually insert across bump column *c* is ρ. Consider the example in Fig. 9, when column c_1 is reached, the number of active signals is 3 and the number of active bumps is 4. Assume N_d=2, then ρ=0. In other words, no P/G trunk should be inserted across column c_1. Next column c_2 is reached, the number of active signals becomes 4 and the number of active bumps becomes 8. Assume N_d=N_{d+1}=2, then ρ=2.

We process the bumps in bump column *c* from bottom to top. Let *i* be the number of rightward dogleg P/G trunks that we have already successfully inserted across column *c* when we process bump *b* in column *c*. Then, the rightward dogleg P/G trunk candidate induced by bump *b* is made up of (1) a lower vertical wire segment *i*+1 grid pitches to the right of bump *b*, (2) an upper vertical wire segment *max*(ρ-*i*, 1) grid pitches a to the left of bump *b*, and (3) a short horizontal wire segment one grid pitch above bump *b* that connects the lower and upper vertical wire segments.

The rightward dogleg P/G trunk candidate induced by bump *b* is feasible if its insertion will form a new sub-region such that the sub-region's bump demand is exactly equal to the sub-region's bump supply. The sub-region's left boundary is the P/G trunk of domain α_{d+i-1} and the right boundary is the trunk candidate under consideration. The sub-region's bump demand is equal to the number of signals inside it plus N_{d+i}. The sub-region's bump supply is equal to the number of bumps inside it. If the rightward dogleg candidate induced by bump *b* is feasible, we will insert it. For example, when we process the bottom bump of column c_2 in Fig. 9, the number of rightward dogleg P/G trunks that we have already inserted across column c_2 is 0 (i.e., *i*=0) and ρ=2 as explained earlier. So, the rightward dogleg P/G trunk candidate induced by the bottom bump of column c_2 consists of a lower vertical wire segment 1 (=*i*+1) grid pitch to its right, an upper vertical wire segment 2 (=*max*(ρ-*i*, 1)) grid pitches to its left, and a horizontal wire segment one grid pitch above it (shown as the red dotted line in Fig. 9). Hence, this trunk candidate is feasible as its insertion will form a new sub-region with bump demand equals to 3 (=number of signals inside sub-region) plus 2 (=N_d) and with bump supply equals to 5.

Let us illustrate how the procedure works on routing region *A*. We have N_S=6 signal traces in the top metal layer. The total number of available bumps *N*=10. Without loss of generality, we let the red, blue, and green P/G domains be α_1, α_2, and α_3, respectively. The bump budgets allocated to the three P/G domains are N_1=1, N_2=1, and N_3=2. In the following, bump *q* refers to the *q*-th bump reached by our procedure which processes the bump columns from left to right and the bumps in each column from bottom to top.

The P/G trunk insertion procedure starts from the leftmost bump column with d=1. It finds that ρ=1 for the leftmost bump column. The rightward dogleg trunk candidate induced by the bump 1 is infeasible since it would form a new sub-region containing one bump but the sub-region's bump demand would be two (one for the single signal trace in the sub-region and one for the red P/G domain). Next, we consider the trunk candidate induced by bump 2, it is feasible since it forms a new sub-region containing two bumps and the sub-region's bump demand is also two (one for the single signal trace in the sub-region and one for the red P/G domain) as in Fig. 7. So, we insert the trunk candidate induced by bump 2. Then, we reach the second bump column and *d* becomes 2. The procedure finds that ρ=1 for the second bump column. The trunk candidate induced by bump 3 is infeasible. But the trunk candidate induced by bump 4 is feasible, so we insert it. Then, we reach the third bump column and *d* becomes 3. The procedure finds that ρ=0 for the third bump column, so the trunk candidate induced by bump 5 or 6 must be infeasible. Then, we reach the fourth bump column and *d* is still 3. The procedure finds that ρ=1 for the fourth bump column. The trunk candidate induced by bump 7 is infeasible. But the trunk candidate induced by bump 8 is feasible, so we insert it as in Fig. 7. Note that in Fig. 7, the sub-region with the trunk candidate induced by bump 8 as the right boundary contains four bumps and the sub-region's bump demand is also four (two for the two signal traces in the sub-region and two for the green P/G domain).

It is clear that by our construction, each time a new rightward dogleg P/G trunk is inserted, the number of bumps in the sub-region to the left of the P/G trunk is equal to the bump demand by all the signals in the sub-region plus the bump demand by the intended P/G domain.

(*footnote continued*)
exceed *W*. If it does, we will abort and report a routing failure (or try again with a different ordering of P/G domains).

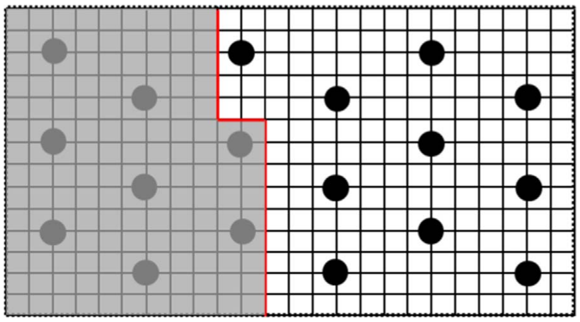


Fig. 10. The shaded region and unshaded region divided by the P/G trunk for α_{m-1}.

But it remains to prove that we can successfully insert a rightward dogleg P/G trunk for each P/G domain. We prove it in Lemma 1 below.

Lemma 1. It is guaranteed that a rightward dogleg P/G trunk will be successfully inserted for each P/G domain using the described procedure.

Proof. . We prove the lemma by contradiction. Suppose we run the procedure on some routing region for which the supply of bumps is equal to the total bump demand by all signal traces and P/G domains, and rightward dogleg P/G trunks have been successfully inserted for P/G domains $\alpha_1, \alpha_2, \ldots, \alpha_{m-1}$. But it cannot find a feasible rightward P/G trunk candidate for P/G domain α_m.

Assume the rightward dogleg P/G trunk candidate for α_{m-1} is induced by the λ-th bump (i.e., the λ-th bump reached by our procedure which processes the bump columns from left to right and the bumps in each column from bottom to top). It divides the routing region into a shaded region which is on the left and an unshaded region which is on the right as in Fig. 10. By our construction, the bump demand by all the signals in the shaded region plus the bump demand by P/G domains $\alpha_1, \alpha_2, \ldots, \alpha_{m-1}$ is exactly equal to the supply of bumps in the shaded region. Hence, the bump demand by all the signals in the unshaded region plus the bump demand by P/G domains $\alpha_m, \alpha_{m+1}, \ldots, \alpha_K$ must be exactly equal to the supply of bumps in the unshaded region.

Let *left_bumps(j)* and *left_signals(j)* denote the number of bumps and the number of signals inside the unshaded region and to the left of the rightward dogleg P/G trunk candidate induced by the *j*-th bump ($j=\lambda+1, \lambda+2, \ldots, N$). Consider $f(j)=left\_bumps(j)-left\_signals(j)-N_m$ as a function defined on $j=\lambda+1, \lambda+2, \ldots, N$. If there exists $j^*(\lambda+1\le j^*\le N)$ such that $f(j^*)=0$, then the rightward dogleg trunk candidate induced by the j^*-th bump is feasible and our procedure would have inserted it for P/G domain α_d.

We note that $f(\lambda+1)\le 0$ since $left\_bumps(\lambda+1)=1$, $left\_signals(\lambda+1)\ge 0$ and $N_m\ge 1$. On the other hand, $f(N)\ge 0$ since *left_bumps(N)* is equal to the total number of signals inside the unshaded region plus the bump demand by P/G domains $\alpha_m, \alpha_{m+1}, \ldots, \alpha_K$. So, either we have case 1: $f(j)=0$ for some $j=\lambda+1, \lambda+2, \ldots, N$, or case 2: $f(j)<0$ and $f(j+1)>0$ for some $j=\lambda+1, \lambda+2, \ldots, N-1$.

We show that case 2 cannot be true. Suppose that $f(j)<0$ for some $j=\lambda+1, \lambda+2, \ldots, N-1$. Then, we have

$$\begin{aligned}
& f(j) < 0 \\
& \Rightarrow left.bumps(j) - leftsignals(j) - N_m < 0 \\
& \Rightarrow (j-\lambda) - leftsignals(j) - N_m < 0 \\
& \Rightarrow (j+1-\lambda) - leftsignals(j+1) - N_m \le 0 \\
& \qquad (\text{as } leftsignals(j+1) \ge leftsignals(j)) \\
& \Rightarrow leftbumps(j+1) - leftsignals(j+1) - N_m \le 0 \\
& \Rightarrow f(j+1) \le 0
\end{aligned}$$

In other words, we cannot have $f(j)<0$ and $f(j+1)>0$ for any $j=\lambda+1, \lambda+2, \ldots, N-1$.

As a result, case 1 must be true which contradicts the assumption that the rightward dogleg P/G trunk candidate induced by the *j*-th bump is infeasible for $\lambda+1\le j\le N$.

4.3. Connecting P/G trunks and signal traces to bumps by network flow

In the second step of our algorithm, we connect the inserted P/G trunks and the signal traces to bumps through the RDL routing grid. We must obtain a routing solution with no crossing on the RDL grid. Note that the P/G trunks inserted in the previous step naturally divides the RDL routing grid into separate routing sub-regions. Hence, we can route each sub-region independently. For instance, the P/G trunks in Fig. 7 yield four routing sub-regions. There is exactly one P/G trunk in each sub-region except the rightmost sub-region which has none.

We use the minimum cost flow approach to compute a feasible routing of the signal traces and the P/G trunk (if any) in each sub-region to the sub-region's bumps. The flow network for a sub-region is constructed as follows. For each grid point *p* of the RDL routing grid within the sub-region, we create a node *p* in the flow network with unit capacity and zero cost. For any two neighboring grid points *p* and *q* of the RDL routing grid within the sub-region, we create a bi-directional edge (*p*, *q*) with unit capacity and unit cost. We add a source node *s* and a sink node *t*. For each signal trace *i* in the sub-region, we add a unit capacity edge from *s* to p_i where p_i is the grid point closest to signal trace *i*. If there is a P/G trunk in the sub-region, we add a node *u*. We add an edge from *s* to *u* with capacity *n* which is equal to the number of bumps that the corresponding P/G domain is entitled to. We also add a unit capacity edge from *u* to *p* for each grid point *p* that the P/G trunk passes through. Finally, we add a unit capacity edge from each grid point that covers a bump to sink node *t*. All edges of the network have zero cost except the bidirectional edges between adjacent grid points which have a cost of 1. And node capacities are associated with grid nodes only. Fig. 11(b) shows the flow network constructed for the sub-region in Fig. 11(a).

As typical minimum cost flow solvers do not handle node capacity, we convert the node capacity into edge capacity by the node splitting technique. Each grid node *p* is split into an incoming node p_1 and outgoing node p_2. All incoming edges to node *p* will go to p_1 instead, and all outgoing edges from node *p* will go from p_2 instead. Nodes $p_1$and p_2 are connected by a directed edge (p_1, p_2) whose capacity and cost are set equal to node *p*'s capacity and cost. See Fig. 12 for an example.

Then, a minimum cost maximum flow from source node *s* to sink

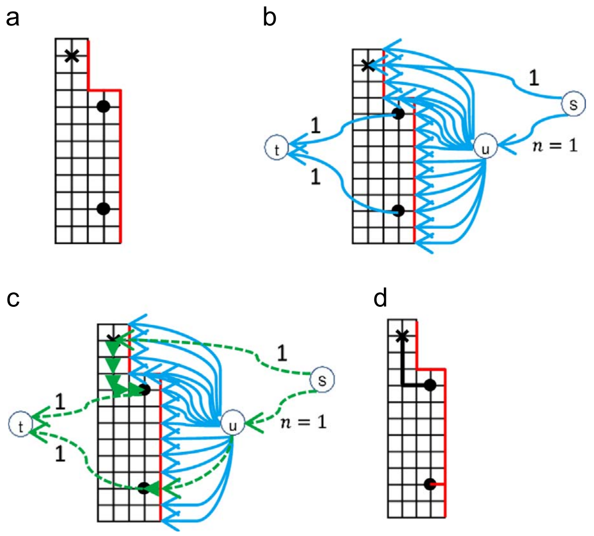
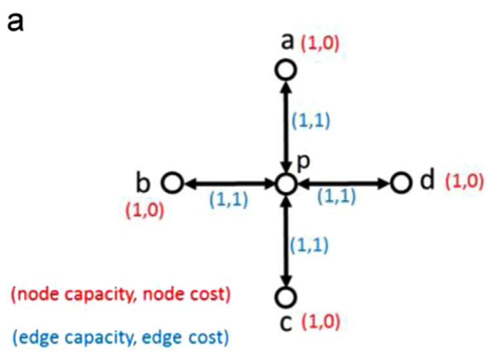


Fig. 11. (a) Routing grid in leftmost sub-region of routing region *A*. (b) Corresponding flow network. (c) The flow result by minimum cost maximum flow algorithm. (d) The routing corresponding to the flow result in (c).



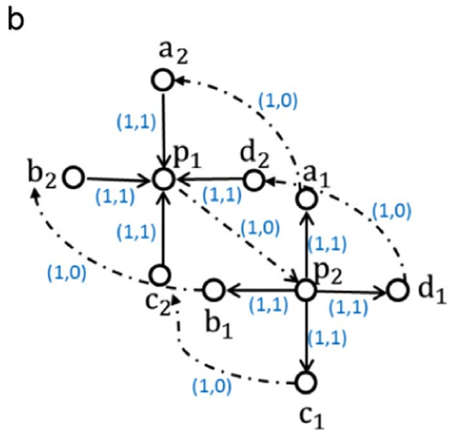


Fig. 12. (a) A grid point p and its neighboring grid points in the initial flow network. (b) Each grid point v is split into an incoming node v_1 and an outgoing node v_2 connected by a directed edge (v_1,v_2) with unit capacity and zero cost.

node t in the constructed flow network will correspond to a feasible routing solution from the signal traces and the P/G trunk (if any) to bumps with the minimum wirelength. Note that by the Integrality Theorem of minimum cost flow [23], there exists an integral optimal solution if all edge capacities in a network are integers. In our constructed network, all edge capacities are integers. Moreover, in this case using algorithms like the network simplex method guarantees to return an integral optimal flow [25].

Fig. 11(c) shows a minimum cost maximum flow where the dashed green edges carry non-zero flow. The corresponding routing result from the signal trace and the P/G trunk in the leftmost sub-region of routing region A to bumps with the minimum wirelength is shown in Fig. 11(d).

Now, for each P/G domain α_i, all its horizontal P/G traces in the top metal layer that pass over domain α_i's P/G trunk τ_i in the redistribution layer can be connected to the P/G trunk by dropping vias. The via connections between the red P/G traces and the red P/G trunk are shown in Fig. 13. Any orphan P/G traces of domain α_i that do not pass over domain P/G trunk τ_i will be connected by a post-processing step to be presented in Section 4.4. Similarly, signal traces in the top metal layer are also connected to their corresponding routing wires in the RDL by dropping vias. Fig. 13 shows an example for the signal trace

Table 2
Benchmark information.

Testcase	#Routing regions	MNB	MNS	MNPG
C1	12	20	11	4
C2	15	20	15	4
C3	20	25	12	5
C4	30	25	15	4
C5	35	20	11	5

Table 3
Comparisons of wirelength and runtime between the maze routing-based method and our method.

	Wirelength		Runtime (s)	
Testcase	Heuristic WL	Our WL	Heuristic WL	Our method
C1	163086(1)	45140 (−72%)	1.50(1)	0.14 (−91%)
C2	109758(1)	36822 (−66%)	1.54(1)	0.17 (−89%)
C3	92554(1)	34462 (−63%)	1.61(1)	0.22 (−86%)
C4	94072(1)	34220 (−64%)	1.61(1)	0.25 (−84%)
C5	92246(1)	34701 (−62%)	1.58(1)	0.16 (−90%)

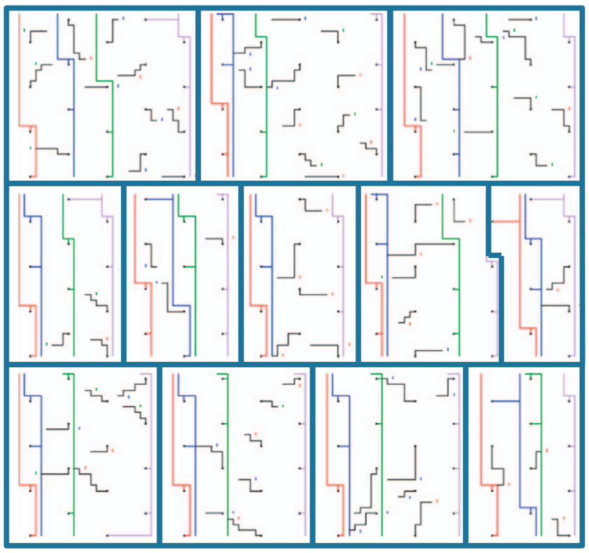


Fig. 14. The RDL routing result of C1. (Four P/G domains are represented in red, blue, green, and violet.). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

near the bottom right corner.

4.4. Post-processing

We perform a post-processing step to connect any orphan P/G traces not connected to P/G trunks in step 1. For each sub-region, we use A*-search on the RDL routing grid to connect each orphan P/G trace to a nearest P/G trace that is already connected to the required P/G trunk. We note that a P/G trace can access the RDL routing grid by

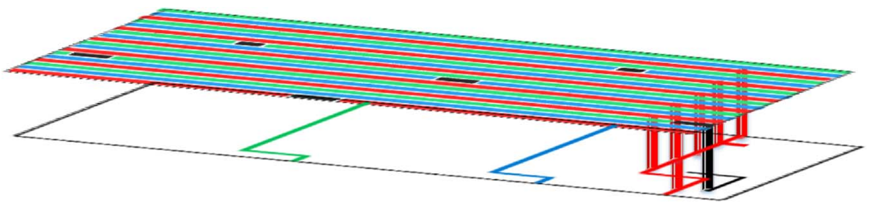


Fig. 13. Dropping vias from the red P/G traces and the bottom right signal trace. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 4
Wirelength and runtime by our method and the maze routing based method assuming technology migration.

Normalized		Wirelength		Runtime (s)	
Area	#Bumps	Heuristic	Our	Heuristic	Ours
1	1200	279312(1)	119088(−58%)	1.61(1)	0.55(−63%)
0.81	972	176616(1)	95592(−46%)	1.62(1)	0.43(−73%)
0.64	768	296208(1)	82392(−72%)	1.6(1)	0.45(−72%)
0.49	588	N/A	86880	NA	0.48

dropping a single via.

5. Experimental results

We implemented our algorithm in the C++ programming language and performed the experiments on a Linux machine with Intel(R) Xeon CPU @2.6 GHz and 64 GB memory. CS2 [24] is employed as our network flow solver.

As none of the existing RDL routing methods directly apply to our problem formulation, for fair comparison, we also implemented a maze routing-based approach. The maze routing on the same routing grid as shown in Fig. 5. Each signal trace is initially connected to the closest grid point and maze routing for the signal starts at that grid point. Each P/G domain is connected by calling maze routing-based approach repeatedly till all its P/G traces are connected with the pre-determined number of P/G bumps. In the interest of wirelength, our heuristic tries to route the signal traces before the P/G domains. If it fails, then it will delay routing some signal traces till all the P/G domains are routed.

We constructed a set of five test cases based on the extracted parameters from 45 nm industrial designs. The detailed information is provided in Table 2. MNB, MNS, and MNPG denote the maximum number of bumps per routing region, the maximum number of I/O signals per routing region, and the maximum number of P/G domains per routing region, respectively. In the experiment, the bump pitch is 180 µm unless otherwise stated, the wire pitches in the RDL and the top metal layer are 22 µm and 2 µm, respectively.

We first compare the wirelength on all the benchmarks between our method and the maze routing-based method. The results are shown in Table 3. From the table we can see that our method outperforms the maze routing-based method in all the cases, with up to 69% shorter wirelength.

In Fig. 14, we show the routing result of testcase C1. The design has a total of four P/G domains represented in red, blue, green, and violet. The colored short vertical connections are added by the post-processing step to connect the few orphan P/G traces.

In addition, we study the impact of technology migration for a particular design. We shrink the chip area and decrease the number of bumps in this chip from 1200 to 588. To keep the total number of bumps being equal to the bump demand, we also reduce the bump budget of each P/G domain progressively. The experimental result is shown in Table 4. There are two factors interacting which affect the wirelength. Chip area reduction can naturally reduce the required wirelength. But when the chip area shrinks too much, the reduction in number of bumps and amount of routing resources in the redistribution layer will start to hurt routability. So, the wirelength by the heuristic method increased drastically when the normalized area is reduced from 0.81 to 0.64. When the normalized area is shrunk to 0.49, the heuristic method even fail to find a routing solution. On the other hand, our method can successfully compute a routing solution in all cases and consistently produces solutions with smaller wirelength.

6. Conclusions

Technology migration plays a critical role in the time-to-market competition. Most existing works focus on layout compaction or hardware description language re-synthesis, and pay little attention to the I/O interface in flip chips. The complication of bumping process as well as electrical and reliability considerations prevent the bumps from scaling with transistor sizes.

On the other hand, the number of signal bumps cannot be reduced and sometimes even increases due to the demands for wider bandwidth and various peripheral devices. As a result, the allocated die area for I/O can no longer afford the number of bumps a chip requires. This issue, known as bump encroachment, puts a stringent requirement on the redistribution layer (RDL) routing. This is the first work to address the RDL routing problem in the context of bump encroachment in technology migration.

We proposed an efficient 3-stage algorithm for connecting the signal traces and P/G traces on the top metal layer to the bumps simultaneously. Compared to a maze routing-based approach, our algorithm can reduce the RDL wirelength by up to 72%.

References

[1] R. Nitzan, S. Wimer, AMPS and SiClone integration for implementing 0.18 µm to 0. 13 µm design migration, in: Proceedings of the Synopsys Users Group (SNUG) Conference, San Jose CA, 2002.
[2] ⟨http://www.intel.com/content/www/us/en/silicon-innovations/intel-tick-tock-model-general.html⟩
[3] R. Saleh, et al., System-on-chip: reuse and integration, in: Proceedings of the IEEE 94(6): (2006) 1050–1069.
[4] S.-Y. Chiang, Foundries and the dawn of an open IP era, Computer 34 (4) (2001) 43–46.
[5] T. Lengauer, CompactionCombinatorial Algorithms for Integrated Circuit Layout, Wiley, Chichester, 1990, pp. 579–643.
[6] M. Reinhardt, Automatic Layout Modification: Including Design Reuse of the Alpha cpu in 0.13 µm soi Technology, Kluwer, Norwell, MA, 2002.
[7] E. Shaphir, R.Y. Pinter, S. Wimer, Efficient Cell-Based Migration of VLSI Layout, Springer, US, 2014.
[8] S. Wimer, Planar CMOS to multi-gate layout conversion for maximal fin utilization, Integration 47 (2014) 115–122.
[9] J. De Vos, L. Bogaerts, T. Buisson, C. Gerets, G. Jamieson, K. Vandersmissen, A. La Manna, E. Beyne, Key elements for sub-50 µm pitch micro bump processes, in: Proceedings of the IEEE 63rd Electronic Components and Technology Conference (ECTC), 2013, pp. 1122–1126.
[10] W. Zhang, Z. Mai, L. Bogaerts, M. Gonzalez, G. Vakanas, A. La Manna, E. Beyne, Mechanical characterization of micro-bump for aggressive bump scaling, in: Proceedings of the Electronic System-Integration Technology Conference (ESTC), 2012, pp. 1–5.
[11] ⟨http://www.itrs2.net/itrs-reports.html⟩
[12] ⟨http://www.intel.com/content/www/us/en/processors/core/desktop-6th-gen-core-family-datasheet-vol-1.html⟩
[13] ⟨http://www.intel.com/content/www/us/en/embedded/products/bay-trail/atom-e3800-family-datasheet.html⟩
[14] H. Leey, Y. Chang, P. Lee, Recent Research Development in Flip-Chip Routing, in: Proceedings of the ICCAD, 2010.
[15] J. Fang, M.F. Wong, Y. Chang, Flip-Chip Routing with unified Area-I/O pad assignments for package-board Co-Design, in: Proceedings of the DAC, 2009.
[16] X. Liu, Y. Zhang, G.K. Yeap, C. Chu, J. Sun, X. Zeng, Global routing and track assignment for flip-chip designs, in: Proceedings of the DAC, 2010, pp. 90–93.
[17] J.-W. Fang, C.-H. Hsu, Y.-W. Chang, An integer linear programming based routing algorithm for flip-chip design, in: Proceedings of the DAC, 2007, pp. 606–611.
[18] P.-W. Lee, C.-W. Lin, Y.-W. Chang, An efficient pre-assignment routing algorithm for flip-chip designs, in: Proceedings of the ICCAD, 2009, pp. 239–244.
[19] J.-T. Yan, Z.-W. Chen, RDL pre-assignment routing for flip-chip designs, in: Proceedings of the GLSVLSI, 2009, pp. 401–404.
[20] J.-W. Fang, K.-H. Ho, Y.-W. Chang, Routing for chip-package-board co-design considering differential pairs, in: Proceedings of the ICCAD, 2008.
[21] J.-T. Yan, Z.-W. Chen, IO connection assignment and RDL routing for flip-chip designs, in: Proceedings of the ASP-DAC, 2009, pp. 588–593.
[22] J.-W. Fang, I. Lin, P.-H. Yuh, Y.-W. Chang, J.-H. Wang, A routing algorithm for flip-chip design, in: Proceedings of the ICCAD, pp. 753–758, 2005.
[23] H. Hsu, M. Chen, H. Chen, On effective flip-chip routing via pseudo single redistribution Layer, in: Proceedings of the DATE, 2012.
[24] CS2. min cost flow solver. [Online]. Available: ⟨http://www.igsystems.com⟩
[25] R.J. Vanderbei, Linear Programming: Foundations and Extensions, Springer, New York, 2007.