Wire-bonding Finger Placement for FBGA Substrate Layout Design with Finger Orientation Consideration

Yu-En Lin and Yi-Yu Liu
Department of CSIE, National Taiwan University of Science and Technology, Taipei 106, Taiwan
{d11315004, yyliu}@mail.ntust.edu.tw

***Abstract*—Wire bonding is a mature packaging technique that enables chip pins to transmit signals to bonding fingers on the substrate through bonding wires. Such commodity technology is also essential in supporting the rapid development of the system in package and heterogeneous integration technologies. However, the automation tools are relatively deficient compared to other packaging techniques, resulting in tremendous manual design time and engineering effort due to numerous wire-bonding design constraints. This paper addresses the finger placement problem and serves as the first work considering the orientation constraint of fingers. The finger placement flow is divided into three stages. First, an integer linear programming (ILP) formulation is developed to allocate each net finger row. After that, we utilize mixed-integer quadratic programming (MIQP) to place the bonding fingers and consider the wire crossing constraint. Finally, the locations of the bonding finger are refined by considering both the bonding finger orientation angle and the finger spacing constraints. The final layouts generated by our integrated finger placement and substrate routing framework outperform manual designs in terms of the design time, the total wirelength, and the routing completion rate.**

I. Introduction

As semiconductor manufacturing continues miniaturization, the size of transistors has reached physical limits. Three approaches have been proposed to ensure the sustainable development of the semiconductor industry: More Moore, More than Moore, and Beyond CMOS [1]. More than Moore can be achieved through heterogeneous integration. Instead of integrating all functions on a single chip (SoC), System in Package (SiP) and heterogeneous integration (HI) involve integrating modules from various processes into the same package. From the system perspective, these solutions offer better performance, power efficiency, area utilization, and cost-effectiveness. The wire-bonding Fine-pitch Ball Grid Array (FBGA) package is the main packaging technology in mature processes thanks to the low manufacturing cost and high yield rate. Therefore, such commodity technology is also essential in supporting the rapid development of SiP and HI [2]. However, the automation tools are relatively deficient compared to other packaging techniques, resulting in tremendous manual design time and engineering effort due to numerous wire-bonding design constraints. Wire-bond packaging uses bonding wires to connect a chip to the substrate, and a FBGA is the interface between the substrate and the PCB. Compared to traditional Surface Mount Technology (SMT), FBGA offers higher connection density and lower assembly cost. The wire-bonding FBGA package is depicted in Figure 1. The signal originates from the die pad, traverses through the bonding wire to the bonding finger, and then proceeds through metal traces and vias on the substrate to reach the bump ball, finally transmitting the signal through the bump ball.

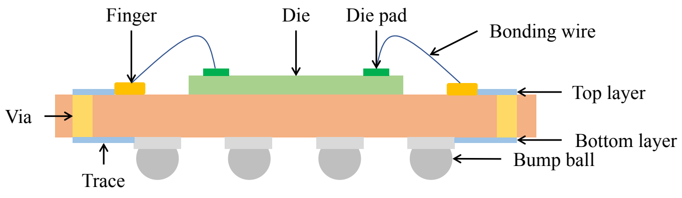


Fig. 1: Cross-section view of a wire-bonding FBGA package.

Research on wire-bonding FBGA packages can be divided into placement and routing problems. Placement determines the finger assignment or locations, and routing deals with establishing the connections between fingers and bump balls. For routing, Wu et al. propose a wire-bonding FBGA package router that considers via and routing trace width mismatch compared to traditional routers [3]. To tackle the placement problem, Lu et al. focus on the finger assignment problem with considerations of routing congestion and signal integrity [4]. They assess routing congestion impact during candidate selection and use a simulated annealing-based refinement, balancing routability and signal integrity. Mak et al. explore pad assignment in a die-stacking SiP design style [5]. The die-stacking architecture uses bonding wires for inter-die pad connections. To address wire crossings resulting from improper pad assignment, a modified left-edge algorithm followed by ILP-based refinement is proposed.

Lin et al. propose the first work to address the finger placement problem for wire-bonding FBGA packages with multiple pad rows and finger rows [6]. The finger placement is formulated as a minimum-cost maximum-flow problem to obtain the initial result followed by dynamic programming and the Abacus method to fix the wire crossings [7]. As drawn in Figure 2a, the proposed algorithm guarantees no crossings when two nets share the same pad row and finger row. However, wire crossing could be generated when two nets with the same pad row and different finger rows as drawn in Figure 2b. In addition, finger row sharing is disallowed when the pad rows are different, as drawn in Figure 2c; this reduces solution space and thus degrades solution quality. Furthermore, placing fingers involves numerous design rules in practice. However, none of the previous works comprehensively consider all these rules. This paper proposes finger placement work for wire-bonding FBGA packages by comprehensively considering all the complicated design rules in industrial designs. Specifically, this is the first work considering the orientation angle constraint of fingers, which will be detailed in the following. Table I summarizes the comparison of design rule considerations between this work and related works.

The rest of this paper is organized as follows: Section II in-

TABLE I: Comparison of design rule considerations with related works.

Work	Objective	Finger spacing	Wire crossing	Multiple pad/finger rows	Feasible region	Orientation angle
[4]	Pad/finger assignment	√				
[5]	Pad assignment	√	√			
[6]	Finger placement	√	△	√	√	
Ours	Finger placement	√	√	√	√	√

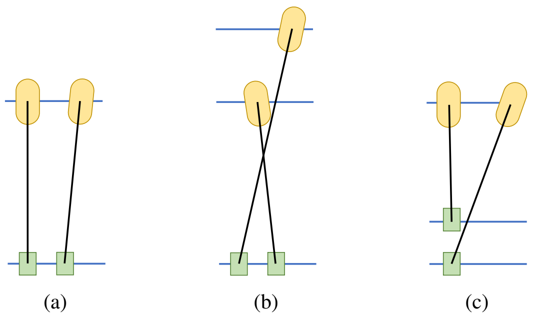


Fig. 2: Wire crossing considerations and limitations in previous work [6]. (a) Two nets with same pad and finger row. (b) Different finger rows. (c) Different pad rows.

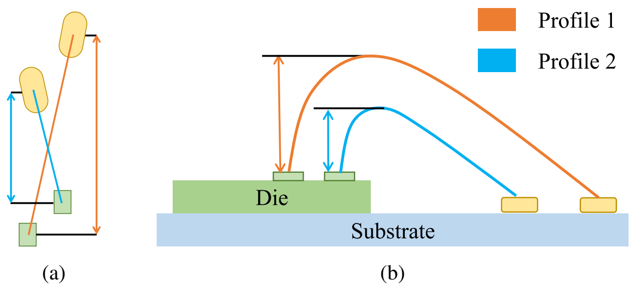


Fig. 3: Profiles of multiple bonding wire loop heights. (a) Top view. (b) Cross-section view.

troduces the wire-bonding design rules and problem formulation. Section III outlines our overall algorithm flow and explains the finger row assignment method. After that, the proposed crossing-free finger placement and finger placement refinement are demonstrated in Section IV. The experimental results are provided in Section V. Finally, Section VI concludes our work.

II. Preliminaries

In the following, Section II-A introduces the design rules to be considered in a wire-bonding package design, and the problem formulation is described in Section II-B.

A. Wire-bonding Design Rules

Theoretically, fingers can be placed anywhere on the substrate, except within the die area [8]. Nevertheless, the finger placement is subject to various design constraints, as fingers directly affect the corresponding bonding wire layouts. We consider four design rules utilized in real industrial designs in this paper as follows:

1. Finger spacing constraint: The distance between two fingers must be greater than or equal to a specified value.
2. Feasible region constraint: The finger connected by each pad must reside within a specified region to meet the bonding wire manufacturing limitations. More details will be provided in Section III-B.
3. Orientation angle constraint: The orientation of each finger should closely align with the angle of the connected bonding wire. The angle between each finger and the corresponding bonding wire cannot exceed a specified value.
4. Wire crossing constraint: No two bonding wires should intersect, except when their nets meet the wire looping height condition, as explained below.

Any bonding wire intersections are forbidden except the blue and orange bonding wires in Figure 3a which are considered feasible because of the wire-bonding 3D structure. Specifically, we define the range of each bonding wire as the orthogonal span from its pad row to its finger row, represented by the double arrow segment in Figure 3a. If the range of one bonding wire is completely inside the range of the other or vice versa, the bonding wire crossing is acceptable. The reason can be explained with Figure 3b, which shows that bonding wires exhibit different wire loop height profiles based on the pad and finger row configuration. Therefore, the bonding wires in Figure 3a can avoid intersections owing to different wire loop heights.

B. Problem Formulation

Given a wire-bonding FBGA substrate design and the netlist where die pads and bump balls are designated for each net in advance. The goal is to minimize the overall bonding wirelength while aiming for improved routing completion rates in the subsequent routing stage. Compliance with the design rules introduced in Section II-A is mandatory during placement.

III. Proposed Flow and Finger Row Assignment

In this section, the proposed algorithm flow is briefly introduced in Section III-A. Section III-B introduces the finger row and candidate generation method. The formulation used for the finger row assignment stage is presented in Section III-C.

A. Algorithm Overview

Figure 4 outlines the algorithm flow. The wire-bonding FBGA substrate layout design automation encompasses three stages: finger placement, via placement, and package routing. The finger placement stage determines the finger position of each net. Subsequently, valid via locations are generated and selected for each net in the via placement stage. Finally, the connections between the fingers and bump balls are established in the routing stage. This paper primarily focuses on automating finger placement, depicted in the right-hand side of Figure 4, which is divided into four stages:

1. Finger row generation: Generate available finger rows and candidates based on the pad locations.
2. Finger row assignment: Determine the finger row for each net finger to be placed.
3. Crossing-free finger placement: Specify the x-coordinate of each net finger and consider the wire crossing and the finger spacing constraints.

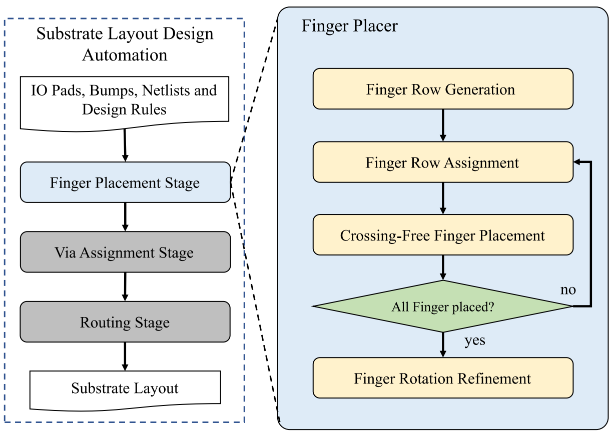


Fig. 4: The framework of the proposed finger placer automation.

4. Finger placement refinement: Fine-tune finger positions to simultaneously satisfy the finger orientation angle and the finger spacing constraints.

The algorithm iterates back to the finger row assignment stage if we are unable to obtain a crossing-free finger placement. Fingers are then reassigned to different rows by expanding the candidate width, and the process repeats until all fingers can be successfully placed. Expanding the candidate width limits the number of fingers in each finger row at the cost of bonding wirelength to alleviate the congestion.

B. Finger Row Generation

Initially, we partition the entire design into four disjoint domains according to the four chip boundaries. Without loss of generality, we demonstrate the finger placement procedure of the top boundary of a chip in the following examples. Firstly, the number of available finger rows is maximized to the number of die-area-exterior bump ball rows. Furthermore, the y-coordinate of each finger row is aligned with the corresponding bump ball row to maximize the number of through-hold via candidate locations. Since the vias connecting fingers and bump balls are huge and non-overlapped to both fingers and balls, the alignment of finger and bump ball rows substantially limits routing resource wastage to enhance the routability. Subsequently, finger candidates are generated one by one based on the candidate width within each finger row. For each net, the feasible finger candidates are those centers located within its feasible region. In Figure 5, the green region specifies the feasible region of the net because the net pad can only be connected with a bonding wire within a specified span angle, the green rectangles lying in the green region are considered feasible finger candidates.

C. Finger Row Assignment

In this stage, we determine the finger row for each net finger by choosing the finger candidate. Our objective is to minimize the overall bonding wirelength and estimated routing wirelength. Consequently, we formulate the finger row assignment problem as an ILP formulation, and the used notations are detailed in Table II. We introduce the required constraints and the objective function.

1) Net Constraint:

$$\sum_{i=1}^{row_{\boldsymbol{F}}} fr_{n,i} = 1, \forall n \in \boldsymbol{N} \tag{1}$$

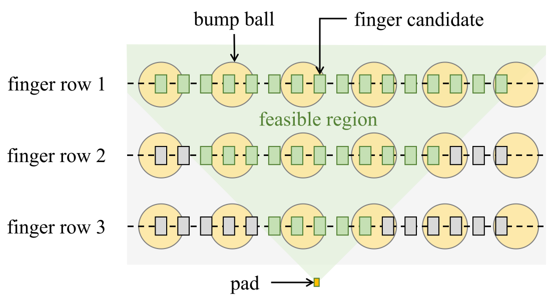


Fig. 5: Finger row and candidate generation and feasible finger candidates.

TABLE II: Notations for finger row assignment.

Notation	Definition
$\boldsymbol{N}$	Set of pad-bump nets.
$\boldsymbol{P}$, $\boldsymbol{B}$, $\boldsymbol{F}$	Set of die pads, bump balls, and finger candidates.
$n_p.x$, $n_p.y$	x and y coordinates of the pad of net n.
$n_b.x$, $n_b.y$	x and y coordinates of the bump ball of net n.
$row_{\boldsymbol{P}}$	Number of pad rows.
$row_{\boldsymbol{F}}$, $col_{\boldsymbol{F}}$	Number of finger rows and columns.
$f_{i,j}.x$, $f_{i,j}.y$	x and y coordinates of the finger candidate at finger row i and column j.
$fr_{n,i}$, $fc_{n,j}$	Binary variable indicating whether net n selects finger row i(finger column j).
fr_{limit}	Maximum number of finger rows can be used.

$$\sum_{j=1}^{col_{\boldsymbol{F}}} fc_{n,j} = 1, \forall n \in \boldsymbol{N} \tag{2}$$

$$\begin{gathered} fr_{n,i} + \sum_{j \notin \boldsymbol{FC}_n^i} fc_{n,j} \leq 1, \\ \forall i : 1 \leq i \leq row_{\boldsymbol{F}}, \forall n \in \boldsymbol{N} \end{gathered} \tag{3}$$

Equations (1) and (2) represent net n choosing one and only one finger row and column by restricting the summation of binary variables to equal one, respectively. $\boldsymbol{FC}_n^i$ is all feasible finger column indices when net n selects finger row i. Equation (3) ensures that the selected finger candidate must lie within the feasible region, that is, $fc_{n,j}$ is in $\boldsymbol{FC}_n^i$.

2) Finger Candidate Constraint:

$$\begin{gathered} \sum_{n \in \boldsymbol{N}} (fr_{n,i} \land fc_{n,j}) \leq 1, \\ \forall i : 1 \leq row_{\boldsymbol{F}}, \forall j : 1 \leq col_{\boldsymbol{F}} \end{gathered} \tag{4}$$

Simultaneous selection of the same finger candidate by multiple nets leads to a short circuit. Equation (4) ensures that any finger candidate is used by at most one net at a time.

3) Maximum Finger Row Constraint:

$$\sum_{i=1}^{row_{\boldsymbol{F}}} \left(\bigvee_{n \in \boldsymbol{N}} fr_{n,i} \right) \leq fr_{limit} \tag{5}$$

Using more finger rows to place fingers increases the variety of wire loop heights required, thereby raising packaging costs and reliability. Therefore, Equation (5) ensures that the number of used finger rows does not exceed the fr_{limit}.

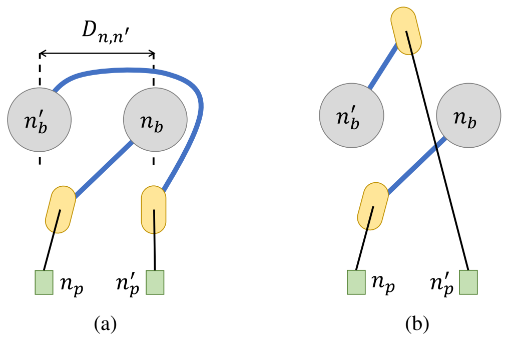


Fig. 6: Example of the routing detour for bad finger row assignment.

4) Wire Looping Control Constraint:

$$\sum_{i'=1}^{i} fr_{n',i'} + fr_{n,i} \leq 1, \quad \forall i : 1 \leq row_{\boldsymbol{F}}, \forall n' \in WLC(n), \forall n \in \boldsymbol{N} \tag{6}$$

We define a set for each net n, denoted as $WLC(n)$, where each net in $WLC(n)$ has its pad located within the feasible region of net n. We consider that net n and the nets in $WLC(n)$ have relatively high probabilities of causing bonding wire crossings because of closely positioned net pads. Therefore, our strategy is to utilize the flexibility of different wire loop heights as described in Section II-A. By labeling the finger rows from top to bottom, Equation (6) ensures that the ranges of the nets in $WLC(n)$ are completely inside the range of net n.

5) Objective Function:

$$Minimize \sum_{n \in \boldsymbol{N}} \sum_{i=1}^{row_{\boldsymbol{F}}} \sum_{j=1}^{col_{\boldsymbol{F}}} w_{n,i,j} \times (fr_{n,i} \wedge fc_{n,j}) + \sum_{(n,n') \in \boldsymbol{RS}} D_{n,n'} \times \bigvee_{i=1}^{row_{\boldsymbol{F}}} (fr_{n,i} \wedge fr_{n',i}) \tag{7}$$

Equation (7) represents the objective function consisting of two parts. The first aims to minimize the overall wirelength, while the latter addresses routing detours. The overall wirelength is the summation of the estimated wirelength $w_{n,i,j}$ of each net n, which is calculated by adding the Euclidean distances between n_p and $f_{i,j}$ and between $f_{i,j}$ and n_b. Routing detours may occur when two nets share the same pad row and bump ball row, and the sequences of their pads and bump balls are reversed. Therefore, we define the set $\boldsymbol{RS}$ to represent such net pairs. Figure 6a shows when both fingers are on the same finger row, one of these two nets requires a detour. On the other hand, in Figure 6b, allocating fingers to different finger rows mitigates the routing detour. Hence, we incorporate this concern into the objective function and add the penalty for each net pair in $\boldsymbol{RS}$ when selecting the same finger row. The penalty value $D_{n,n'}$ is defined as the difference in the x-coordinates between the bump balls of these two nets as the estimated detour wirelength.

IV. Finger Placement and Refinement

Although the ILP formulation above has assigned a finger candidate to each net, a generated result is not immune to bonding wire crossings. Therefore, we further perform finger placement and refinement in this stage by leveraging the finger row assignment results. Section IV-A introduces crossing-free finger placement to ensure placement results comply with design rules. After that, the methods addressing the finger orientation angle constraints are discussed in Section IV-B.

TABLE III: Notations for crossing-free finger placement.

Notation	Definition
$n_f.x$, $n_f.y$	x/y-coordinate of the finger of net n.
$n_p.x$, $n_p.y$	x/y-coordinate of the pad of net n.
$n_b.x$, $n_b.y$	x/y-coordinate of the bump ball of net n.
$n_f.min$, $n_f.max$	Leftmost/rightmost x-coordinate of the finger of net n.
f_{pitch}	Minimum pitch between two fingers.

A. Crossing-Free Finger Placement

We determine the x-coordinate of each finger at this stage instead of using the selected finger candidate to achieve a crossing-free finger placement result. An MIQP formulation is developed to reach this goal. Table III lists the notations used in the formulation, and the constraints and the objective function are detailed in the following.

1) Finger Feasible Region Constraint:

$$n_f.x \geq n_f.min, \forall n \in \boldsymbol{N} \tag{8}$$

$$n_f.x \leq n_f.max, \forall n \in \boldsymbol{N} \tag{9}$$

Each net finger must stay within its feasible region. Equations (8) and (9) ensure that the x-coordinate of each finger remains within the interval $[n_f.min, n_f.max]$. The value of $n_f.min$ and $n_f.max$ is determined by the net pad location and the assigned finger row from the previous stage drawn in Figure 5.

2) Wire Crossing Constraint:

$$n_f.x + f_{pitch} \leq n'_f.x, \forall (n,n') \in \boldsymbol{N_S^S} \tag{10}$$

$$n_f.x \leq \frac{(n_f.y - n_p.y)\left(n'_f.x - n_p.x\right)}{n'_f.y - n_p.y} + n_p.x, \quad \forall (n,n') \in \boldsymbol{N_S^D} \tag{11}$$

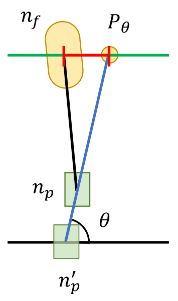
$$L_n^{n'} + R_{P_\theta}^{n'} = 1, \forall (n,n') \in \boldsymbol{N_D^S} \tag{12}$$

$$L_{P_{\theta'}}^{n'} + R_{P_\theta}^{n'} = 1, \forall (n,n') \in \boldsymbol{N_D^D} \tag{13}$$

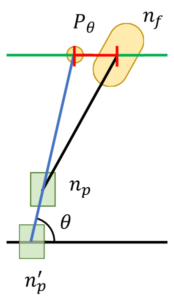
$$\begin{aligned} L_n^{n'} = 1 &\implies n'_f.x + f_{pitch} \leq n_f.x \\ L_n^{n'} = 0 &\implies n'_f.x - f_{pitch} \geq n_f.x \end{aligned} \tag{14}$$

$$\begin{aligned} R_{P_\theta}^{n'} = 1 &\implies n'_f.x \geq P_\theta.x \\ R_{P_\theta}^{n'} = 0 &\implies n'_f.x \leq P_\theta.x \end{aligned} \tag{15}$$

Net pairs are categorized into four conditions: $\boldsymbol{N_S^S}$, $\boldsymbol{N_S^D}$, $\boldsymbol{N_D^S}$, and $\boldsymbol{N_D^D}$ based on whether their pad rows (denoted in subscript) and finger rows (denoted in superscript) are the same (denoted as S) or different (denoted as D). In *the same pad row and*



(a) n_f is left of P_θ.



(b) n_f is right of P_θ.

Fig. 7: Bonding wire crossing intervals for net set $\boldsymbol{N_D^S}$.

finger row condition, Equation (10) ensures that the net pair finger sequence complies with the corresponding pad sequence. Based on the mathematical relationships of similar triangles, Equation (11) prevents wire crossing under *the same pad row and different finger rows* condition. Equation (12) avoids the net pair n, n' from crossing under *the same finger row and different pad rows* condition, drawn in Figure 7. The wire crossing occurs when n'_f falls within the red interval between finger n_f and point P_θ. Point P_θ is the intersection of the extended line segment $\overline{n'_p n_p}$ and the net finger row. To avoid finger n'_f being in the red interval, two auxiliary binary variables, $L_n^{n'}$ and $R_{P_\theta}^{n'}$, are introduced to specify whether n'_f is to the left of n_f and to the right of P_θ in Equations (14) and (15), respectively. Equation (12) avoids net finger n'_f in the red interval by setting either $L_n^{n'}$ or $R_{P_\theta}^{n'}$ to be 1. Similarly, Equation (13) avoids net pair from crossing under *different finger rows and pad rows* condition. We omit the detailed illustration for brevity.

3) Objective Function:

$$minimize \sum_{n \in \boldsymbol{N}} (n_p.x - n_f.x)^2 + (n_b.x - n_f.x)^2 \qquad (16)$$

The objective function minimizes the wirelength along the x-axis. Equation (16) calculates the squared differences between the x-coordinates of pads and fingers and between fingers and bump balls. The indicator and logical AND/OR constraints can be converted into linear and Special-Ordered Set(SOS) constraints using auxiliary variables [9]. Therefore, the formulation is solved with MIQP.

B. Finger Placement Refinement

We assume each bonding finger is not rotated before this stage. Adjusting bonding finger orientation angles can result in finger spacing constraint violations. A complicated problem is induced because applying a finger rotation to meet the orientation angle constraint could require finger displacement to avoid the spacing violation. However, moving a finger might also cause another orientation angle violation that requires further rotation. Therefore, minimizing rotation angles can simultaneously minimize finger displacement. To meet this goal, fingers are sorted based on their x-coordinates in ascending order and are processed one at a time sequentially. We only rotate fingers when they violate the orientation angle constraints. If the distance between the rotated and the preceding fingers violates the finger spacing constraint, the currently rotated finger is displaced rightward. In Figure 8, the green line represents the distance between the fingers of net

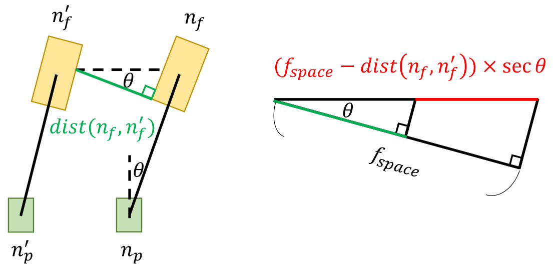


Fig. 8: Horizontal displacement calculation.

TABLE IV: The specifications of industrial designs.

Benchmarks	Nets	Pads	BGA	pr	fr_{avail}	fr_{man}
Industrial1	59	75	8 × 8	1	1	1
Industrial2	95	113	10 × 10	1	1	1
Industrial3	255	256	16 × 16	1	4	1
Industrial4	188	370	20 × 20	1	3	2
Industrial5	285	517	22 × 22	2	8	4
Industrial6	301	483	23 × 23	2	7	3

n and n', and θ is the angle between the bonding wire of net n and the vertical dashed line. The red line indicates the rightward displacement for the finger of net n if its orientation angle does not change. Equation (17) presents the formulation for the required displacement, inspired by the similar triangles in Figure 8.

$$move_{n_f} = min\left(f_{space} - dist\left(n_f, n'_f\right), 0\right) \times sec\theta \qquad (17)$$

Notice that the finger orientation angle of net n might be adjusted again after a rightward displacement to avoid the displacement-induced orientation angle constraint violation. Therefore, Equation (17) is valid only when there is no further displacement-induced orientation angle constraint violation. To avoid indefinite rotations and displacements on the same finger, Equation (18) illustrates the heuristic computation method for the x-coordinate of the net finger n_f considering Equation (17). Specifically, a small constant α is added to ensure a larger displacement stride. Consequently, the finger orientation and displacement are refined sequentially to resolve both the finger spacing and the orientation angle constraints.

$$n_f.x_{(t+1)} = n_f.x_{(t)} + move_{n_f} + \alpha \qquad (18)$$

V. Experimental Results

We implement our methodology using the C++ programming language, and the experiments are conducted on the Linux workstation equipped with a 2.9GHz Intel Xeon Gold 6326 processor and 256GB of memory. To address the ILP and MIQP problems, we employ Gurobi Optimizer 10.0.1 for the solution [10]. The industry provides 6 benchmarks, and the corresponding design information is presented in Table IV. Column "Nets" gives the total number of nets. "Pads" and "pr" represent the number of die pads and pad rows, respectively. "BGA" specifies the row and column of the ball grid array. "fr_{avail}" and "fr_{man}" show the number of available finger rows and those used in the manual design, respectively. For fairness, the number of finger rows used by "Ours" matches that of "fr_{man}". Finally, the angle between

TABLE VI: Estimated wirelength, routing completion rate, and runtime comparisons.

Benchmarks	Manual		RbR+ [6]			RbR+CFFP			Ours		
	ETWL	CR	ETWL	CR	T(s)	ETWL	CR	T(s)	ETWL	CR	T(s)
Industrial1	119,308	86.9%	117,242	92.0%	1.5	116,815	90.7%	0.8	116,815	90.7%	1.1
Industrial2	223,570	96.0%	198,271	96.5%	1.8	198,369	96.5%	1.1	198,369	96.5%	0.9
Industrial3	1,072,902	96.9%	1,047,397	98.8%	3.9	1,056,243	99.6%	1.9	984,780	98.8%	6.1
Industrial4	642,512	84.6%	753,841	74.9%	5.8	768,910	73.1%	3.0	533,116	80.9%	3.0
Industrial5	2,199,735	74.5%	2,421,427	72.1%	9.7	2,430,734	72.5%	4.1	2,008,293	81.5%	78.0
Industrial6	2,154,819	85.9%	3,141,228	84.2%	8.3	3,152,493	84.2%	3.6	2,107,423	81.8%	13.0
Average	1.09	87.5%	1.19	86.4%	1.11	1.20	86.1%	0.60	1.0	88.2%	1.0

TABLE V: Bonding wire crossing comparisons.

Benchmarks	Manual	[6]	RbR+ [6]	RbR+CFFP	Ours
Industrial1	0	0	0	0	0
Industrial2	0	0	0	0	0
Industrial3	0	0	0	0	0
Industrial4	41	479	207	0	0
Industrial5	31	977	0	0	0
Industrial6	18	812	0	0	0

each finger and its bonding wire cannot exceed $15°$ based on industrial practice.

Firstly, Table V reports bonding wire crossing violations in different placement methods. "Manual" denotes the results achieved through manual placement by experienced substrate layout engineers. "RbR+ [6]" employs a row-by-row (RbR) assignment method, which assigns all nets in the same pad row to the same finger row and ensures the wire looping constraint in Figure 3, before the finger placement proposed in [6]. "RbR+CFFP" combines the row-by-row assignment with our proposed crossing-free finger placement (CFFP). In single-finger-row designs, *Industrial1-3*, no crossings are observed; while in multi-finger-row designs, though [6] and "Manual" suffer from numerous crossings, our method still guarantees no crossings. Although row-by-row assignment guarantees different wire loop heights in different pad rows drawn in Figure 3, it is worthwhile noting that there are still 207 crossings in "Industrial4" of "RbR+ [6]". The reason behind this anomaly is that the die pad count in one pad row exceeds the finger candidate count of the corresponding finger row. Therefore, multiple finger rows are inevitable to accommodate the bonding fingers of the corresponding die pad row and [6] cannot avoid crossings from one pad row to multiple finger rows drawn in Figure 2b. In contrast, our proposed crossing-free finger placement can effectively prevent crossing violations as shown in Table V. In addition, since [6] is unable to tackle the finger orientation angle constraint, we do not consider [6] in the following experiment.

Table VI demonstrates the effectiveness of the proposed placement flow. "ETWL" stands for estimated total wirelength, including both estimated bonding wirelength and routing wirelength. The estimated bonding wirelength is the Euclidean distance between pads and fingers, while the estimated routing wirelength refers to the connections between fingers and bump balls using FLUTE [11]. "CR" is the routing completion rate by taking the finger placement results as inputs for the router [3]. "T(s)" denotes the runtime of finger placement in seconds. The larger solution space exploration in the ILP-based finger row assignment stage, caused by the increased "fr_{avail}", results in runtime overhead in "Industrial5 and 6" with our approach. However, our method substantially reduces estimated total wirelength in 9%, 19%, and 20%, respectively, compared to the "Manual", "RbR+ [6]", and "RbR+CFFP". Furthermore, our finger placement results achieve the best routing completion rate in the follow-up substrate routing stage.

VI. Conclusion

In this paper, we tackle the design automation problem for finger placement. We divide the finger placement problem into three main stages. Firstly, we utilize finger row assignment to determine each net finger row. Next, we employ crossing-free finger placement to address the wire crossing constraint during placement. Finally, the finger placement refinement fine-tunes finger locations to simultaneously consider the finger orientation angle and the finger spacing constraints. The experimental results indicate that the finger placement results obtained by our integrated framework achieve 9% wirelength reduction and 0.7% routing completion rate improvement compared to manual finger placement results. The final layout of our integrated finger placement and substrate routing framework can be efficiently adopted as an effective reference design for layout engineers to start with a new package design project.

References

[1] Yao-Wen Chang, "EDA for More-Moore and More-than-Moore Designs: Challenges and Opportunities," *IEEE/ACM International Conference On Computer Aided Design*, 2020. Keynote.

[2] Heterogeneous Integration Roadmap, Chapter 8: Single Chip and Multi-Chip Integration. https://eps.ieee.org/technology/heterogeneous-integration-roadmap.html.

[3] Jun-Sheng Wu, Chi-An Pan, and Yi-Yu Liu, "ILP-based Substrate Routing with Mismatched Via Dimension Consideration for Wire-bonding FBGA Package Design," *ACM Transactions on Design Automation of Electronic Systems*, pp.1-26, vol.28, i.5, 2023.

[4] Chao-Hung Lu, Hung-Ming Chen, Chien-Nan Jimmy Liu, Wen-Yu Shih, "Package routability- and IR-drop-aware finger/pad assignment in chip-package co-design," *Design, Automation & Test in Europe Conference & Exhibition*, pp.845-850, 2009.

[5] Wai-Kei Mak, Yu-Chen Lin, Chris Chu, Ting-Chi Wang, "Pad Assignment for Die-Stacking System-in-Package Design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp.1711-1722, vol.31, i.11, 2012.

[6] Yu-En Lin, Che-Hsu Lin, Yi-Yu Liu, "Wire-bond package finger placement with minimal distance," *In Proceedings of Workshop on Synthesis And System Integration of Mixed Information Technologies*, pp.186-191, 2021.

[7] Peter Spindler, Ulf Schlichtmann, Frank M. Johannes, "Abacus: fast legalization of standard cell circuits with minimal movement," *Proceedings of International Symposium on Physical Design*, pp.47-53, 2008.

[8] Farhang Yazdani, "Foundations of Heterogeneous Integration: An Industry-Based, 2.5D/3D Pathfinding and Co-Design Approach," *Springer*, 2018.

[9] H. Paul Williams, "Model Building in Mathematical Programming, 5th Edition," *Wiley*, 2013.

[10] Gurobi Optimizer 10.0.1. http://www.gurobi.com/.

[11] Chris Chu, "FLUTE: fast lookup table based wirelength estimation technique," *IEEE/ACM International Conference on Computer Aided Design*, pp. 696-701, 2004.