Wire-bond Package Finger Placement with Minimal Distance

Yu-En Lin, Che-Hsu Lin, Yi-Yu Liu
Department of Computer Science and Information Engineering,
National Taiwan University of Science and Technology, Taipei City, Taiwan, R.O.C.
b10615030@mail.ntust.edu.tw, b10615020@mail.ntust.edu.tw, yyliu@mail.ntust.edu.tw

Abstract

Abstract— Semiconductor packaging is the final stage of fabrication, which encapsulates one or more integrated circuit chips to avoid physical damage and to provide interconnections to outside world. There are a variety of packaging design styles, such as wire-bond, flip-chip, 3-D TSV, and so on. In this work, we propose a framework to handle the finger placement problem in wire-bond package design style. First, a minimum-cost-maximum-flow-based global finger placement algorithm is used to derive initial finger locations with overall minimum distance. After that, legalization steps are proposed taking bond-wire crossing constraint and pad row sequence constraint into consideration. Finally, incremental finger placement refinement algorithm is adopted to generate final layout with minimal displacement compared to the initial global placement. With the proposed framework, legalized finger locations are determined in polynomial time for package substrate layout engineer to start with.

I. Introduction

In the past, the semiconductor packaging industry is driven by new scaling CMOS technology in order to meets the issues in advanced process nodes. So as to achieve the Moore's Law, the transistor counts of semiconductor chips would double roughly every 18 months, the primarily solution is to shrink the size of transistors [1]. However, this scaling slows down owing to the feature size of transistors near atomic scale. The industry has been forced to seek alternative approaches. For instance, the concepts of heterogeneous integration, 2.5D/3D packaging are considered to drive the semiconductor industry to another generation. In the past, most of IC designers paid less attention to packaging technology. IC design house determines package design specifications for outsourced semiconductor assembly and test (OSAT) company. The OSAT is required to finalize package substrate layouts in time taking miscellaneous constraints, such as signal integrity, power integrity, unit cost, into account. However, as the slow down of IC manufacturing technology, IC design houses are now cooperating with the packaging company for better optimized die-package-board pathfinding and co-design. Therefore, the packaging is what the next generation IC designers must understand. [8]. There are a lot of package design styles, such as wire-bond, flip-chip, 3-D TSV, and so on. In order to realize a competitive design, packaging style selection requires to have number of I/Os, signal integrity, power integrity, performance rating, thermal dissipation requirements, and unit cost in mind. In this paper, we focus on wire-bond packaging design style.

The rest of this paper is organized as follows. Section II introduces the wire-bond package and design rules. Section III gives an overview of our finger placement flow. Section IV presents a dynamic-programming approach to legalization the violation. The experimental results are illustrated in Section V. Finally, Section VI concludes our work.

II. Wire-bond Package Design

Wire-bond package is one of widely used design styles that connects device to outside world using conductive bonding wires. The common wire material is gold and copper for better ductility and lower resistivity. Wire-bond technology provides a low-cost packaging option for a design with medium number of I/O (e.g., less than 800). Figure 1 illustrates the structure of wire-bond package. There are two terminals in each bonding wire, the bond pad and bond finger. The bond pads normally are situated on the periphery of the package die and bond fingers are placed on the substrate. Before designing the substrate routing, bond-pad locations must be determined and optimized taking substrate routing capacity into account. The bond-pad layout can be designed in either one-line or multi-line based on the package die specification. Several bond-pad layout design rules, such as center pad pitch and corner pad pitch, are enforced to ensure wire-bond yield. A bond finger is the wire-bond terminal on the substrate side to bridge a bond-pad signal on the die side to a substrate bump ball. The relationship between bond pad and bond finger could be one-to-one, one-to-many, or many-to-many. Consequently, there are several bond-finger types available for the aforementioned relationship. In addition, there are two important bond-finger design rules to avoid bonding wire short circuit. (1) Two distinct bonding wires terminated at the same finger

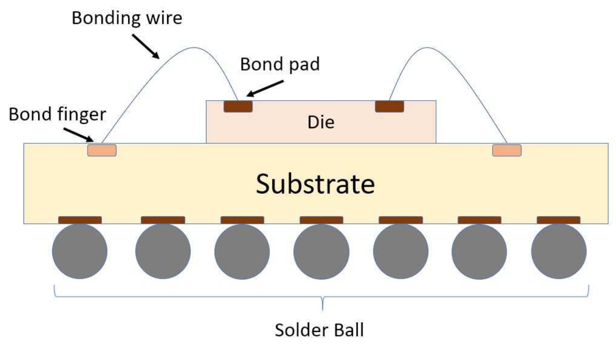
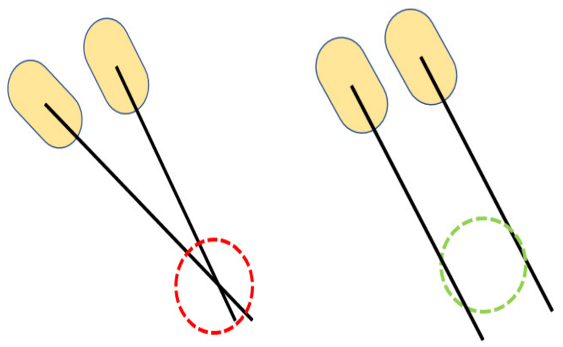


Fig. 1. Anatomy of a wire-bond package

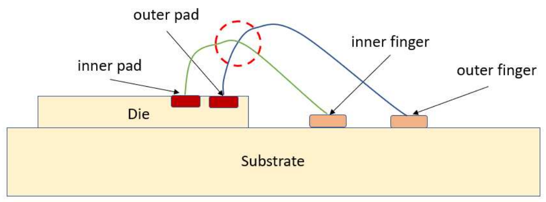


(a) Invalidation (b) Validation

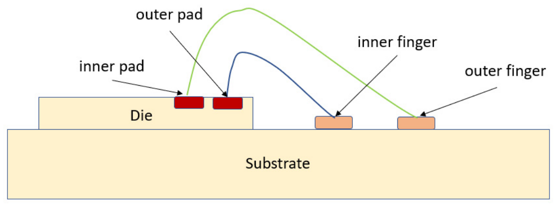
Fig. 2. Design rule 1: Wire crossing constraint.

row cannot be across each other. (2) In multi-line bond-pad design, the bond-finger row sequence must be in reversed order of the corresponding bond-pad row sequence. The second design rule is important when pad layout is multi-line in order to avoid short circuit between different bonding wires. Figures 2 and 3 illustrate the bond-finger design rules.

Given a netlist, die bond-pad locations, substrate solder-ball locations, and package design rules, the wire-bond finger placement problem is to determine proper bond-finger locations for follow-up package substrate routing design to start with. In practice, the IC design houses decide the connection relationship between bond pads and balls. The package companies need to design appropriate finger positions without violating design rule, routing congestion, power integrity, and so on. Currently, experienced layout engineers are required to manually design the substrate layout in order to fulfill complex design rules. In the paper, we propose a framework to automate bond finger placement with minimum distance. Our preliminary placement result generates a quick and legalized reference for substrate layout engineers to start with.



(a) Invalidation



(b) Invalidation

Fig. 3. Design rule 2: Reversed order of bond-pad row sequence to the corresponding bond-finger row sequence in multi-line bond-pad design.

III. Finger Placement

First of all, our algorithm is consist of pre-processing, global finger placement, legality analysis, and detailed finger placement stages, summarized in Figure 4. Pre-processing stage deals with multi-pads corresponding multi-balls finger merging and pad segment identification and grouping. Global finger placement stage converts finger placement problem to a network-flow problem with global view to determine appropriate finger locations. Legality analysis detects bonding wire crossing since we do not consider any design rule in global finger placement stage. We adopt longest common subsequence (LCS) to highlight potential violations. Finally, detailed finger placement stage moves fingers by the least distance way or increases the least cost to satisfy the design rules. In this section, we introduce pre-processing, global finger placement, and legality analysis stages.

We need to meet the following two design rules mentioned earlier:

(1) Two distinct bonding wires terminated at the same finger row cannot be across each other. (2) In multi-line bond-pad design, the bond-finger row sequence must be in reversed order of the corresponding bond-pad row sequence.

A. *Pre-processing Stage*

We use two diagonals to divide a package die into UP, DOWN, LEFT, and RIGHT sectors. First, we use coordinate to separate pads into four groups within each sector. For corner bond pads, we additionally use proportion to solve this problem. For power/ground nets,

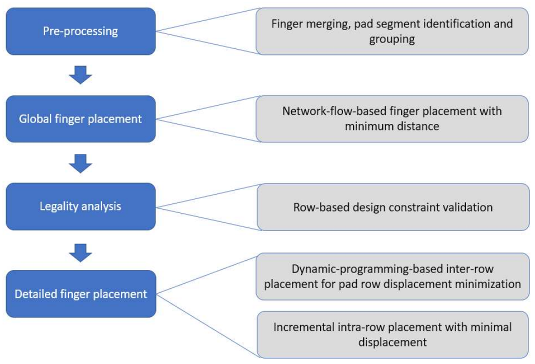

Pre-processing
Finger merging, pad segment identification and grouping
Global finger placement
Network-flow-based finger placement with minimum distance
Legality analysis
Row-based design constraint validation
Detailed finger placement
Dynamic-programming-based inter-row placement for pad row displacement minimization
Incremental intra-row placement with minimal displacement

Fig. 4. The algorithm flow.

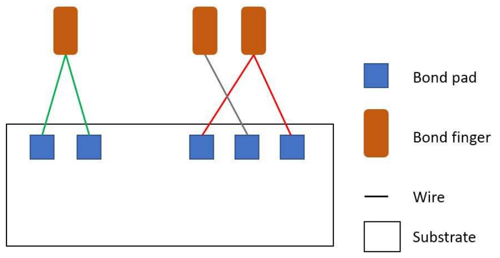


Fig. 5. Power/ground pad and finger merging.

multiple bond pads connect to the same source. To avoid duplicating massive identical bond fingers of the same power/ground net, we decide to merge some bond pads in the same group. The basic idea is to merge nearby bond pads without incurring illegal wire crossing. These merged bond pads could share one large finger instead of using multiple individual fingers. Figure 5 illustrates two situations. The left-hand-side wires, colored in green, is legal merging; while the right-hand-side wires, colored in red and gray, is illegal merging.

Next, we introduce a virtual segment which literal meaning is part of a line to record a start, end position, and all of pad's name within this segment. The pad segment is formed based on the distance of adjacent pads. First, we sort pad according to x-position. Then, the x and y-position of a pad is used to determine which segment it belongs to. There are two threshold parameters which are equal to pad-to-pad width for detecting whether two pads close enough. Figure 6 is an example, each blue rectangle represents a segment. The result is that 11 bond pads are divided into 5 segments.

After that, we group overlapped segments to simulate the relationship of inner pads and outer pads. In this stage, we want to find the relationship between segments. Because each segment is horizontal, the overlapping happens when they are in different y-position. In each segment group, the design rule 2 must be enforced to maintain the reversed order of bond-finger row sequence and the corresponding bond-pad row sequence. First, we sort

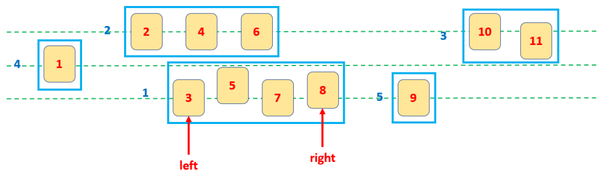


Fig. 6. Example for detecting pad segments.

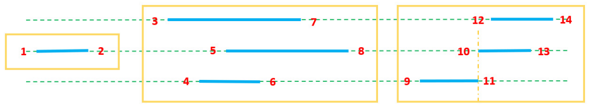


Fig. 7. Example for grouping overlap segments

the endpoints of all pad segments according to x-position, breaking ties by putting left endpoints before right endpoints. Then we construct a segment group when all segments in the group are finished. Therefore, there must not be any overlap between the segment groups. Figure 7 draws an example of segment grouping, orange rectangle represents segment group. The 7 pad segments are grouped into 3 groups.

Finally, some of the segments in segment group can be merged to one segment to increase the feasible solutions and decrease the possibility of failure in follow-up inter-row placement. In previous step, a segment group represents that each segment has at least one overlap with other segments in the same segment group. Nevertheless, not all pairs of segments in the same group overlap with each other. Therefore, we can merge two non-overlapped segments whenever there is no other segment between them.

B. *Global Finger Placement*

In order to obtain the finger placement with least wire length, we transform the finger placement problem into a minimum cost maximum flow problem. First, there are two virtual points representing source and sink. Second, source will link to each pad and cost are set to zero, then pads will link to legal fingers and cost are set to the distance between pad and finger plus the distance between finger ball. Finally, each fingers will link to sink and cost are also set to zero. Figure 8 shows our network flow model. The edge flow constraint in the network graph is set to one, because one pad only correspond to one finger, and vice versa. The number of target flow exactly equals to the number of (merged) pads, and the total cost represents that sum of the distance between pad and finger plus the distance between finger ball.

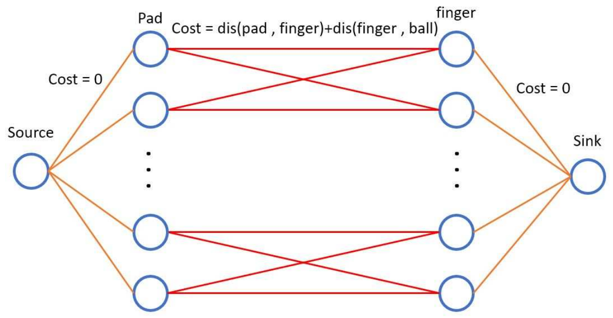


Fig. 8. Network Flow Model for Global Finger Placement

C. Legality Analysis

We use longest common subsequence (LCS) to highlight the potential design rule 1 violations after global finger placement. If two subsequences overlaps, there are two wire crossings. This violation problem is resolved in Section IV.

IV. Detailed Placement

After the legality analysis step is finished, the violations incurred in the global placement stage should be resolved. In order to meet the aforementioned design rules, we use two methods to separately fix the violations of two design rules. In this section, we propose inter-row placement and intra-row placement.

A. Inter-row placement

The inter-row placement enforces design rule 2. All of the pre-processing steps except pad merging are prerequisites of this step. We model the inter-row placement as follows. Giving m segments, their relationship, and n finger rows, each segment must be assigned to one finger row. The objective is to obtain best assignment solution without violating design rule 2. The following formulas illustrate the problem:

$$o[m,n] = min(o[m,n-1], c[m,n]+o[m-1,n-1]), m <= n$$

$$o[m,n] = \infty, m > n$$

Above formulas mean that m segments assigned to n finger rows is minimum of m segments assigned to $n-1$ finger rows or $m-1$ segments assigned to $n-1$ finger rows and m^{th} segment assigned to n^{th} finger row when m is smaller than or equal to n;otherwise, it is equal to unlimited because the segment number can't larger than finger row number.

Algorithm 1 describes the dynamic programming approach, $o[m,n]$ array records the totally cost of assignment, $c[m,n]$ array records the cost of m^{th} assigned into

Algorithm 1: Assign pad segment

```
1  create o[0..m, 0..n] and p[1..m, 1..n] tables;
2  for $i = 0$ to $n$ do
3      $o[0,i] = 0$;
4  end
5  for $i = 1$ to $m$ do
6      $o[i,0] = \infty$;
7  end
8  for $i = 1$ to $m$ do
9      for $j = 1$ to $n$ do
10         if $o[i,j-1] <= c[i,j] + o[i-1,j-1]$ then
11             $o[i,j] = o[i,j-1]$;
12             $p[i,j] = LEFT$;
13         else
14             $o[i,j] = c[i,j] + o[i-1,j-1]$;
15             $p[i,j] = UP - LEFT$;
16         end
17     end
18 end
```

n^{th} row, and $p[m,n]$ array records the path of the solution to backtrack the optimum solution. Line 1 to 7 create and initialize the $o[0,n]$ and $o[m,0]$ value. Line 10 judges whether adding one more finger row can find better solution of $o[m,n]$. Line 11 to 13 is the situation it doesn't meet the line 10 condition. On the other hand, line 14 to 15 is the situation it meets the line 10 condition.

B. Intra-row placement

After inter-row placement, the finger rows of all pad segments are determined, i.e., the y-positions of all fingers are fixed without any wire-looping design rule violation. In intra-row placement, we only need to solve the design rule 1 violations. We found that the wire-crossing problem in finger placement is analogous to the standard cell legalization problem. Hence, we adopt the classic framework, Abacus [4], to legalize the intra-row bond-finger locations. Notice that, the fingers could be adjusted within the same finger row, i.e. in x-position, so that the design rule 2 could be kept. Therefore, the intra-row placement is to eliminate all wire-crossing in the same finger row with minimal total displacement. Table I lists the parameters and our problem is:

$$min\sum_{i=1}^{N}[[Fx(i)-Px(i)]^2-[Fx'(i)-Px(i)]^2 + [Fx(i)-Bx(i)]^2-[Fx'(i)-Bx(i)]^2] \quad (1)$$

$$s.t. Fx(i)-Fx(i-1) >= Fw(i-1) \quad i = 2,...,N \quad (2)$$

The objective Equation (1) describes the difference between before and after intra-row placement total x-direction wire distance. The constraint Equation (2) assures that there is no overlap between the wires if and

TABLE I
Parameters of intra-row placement

Property	Explanation
Fx'(i)	finger x-position in inter-row placement
Fx(i)	finger X-position in intra-row placement
Fw(i)	finger width
Px(i)	pad x-position
Bx(i)	ball X-position

only if the order of fingers is same as the order of corresponding pads.

Spindler et al. point out that the constraints "≥" is time consuming, so we follow this observation to only consider the solution with "=" constraints [4]. Therefore, Equation (2) is transformed to:

$$Fx(i) = Fx(1) + \sum_{j=1}^{i-1} Fw(j) \qquad i = 2, ..., N \tag{3}$$

Inserting Equation (3) in Equation (1) that only depends on one variable, Fx(1). Consequently, we can obtain the minimum value by using first-order derivative to zero, which gives:

$$2\sum_{i=1}^{N} Fx(1) - [[Px(1) + Bx(1)] + \sum_{i=2}^{N}[Px(i) + Bx(i) - 2\sum_{k=1}^{i-1} Fw(k)]] = 0$$

Since the equation is the same to Abacus, we could use the dynamic-programming approach for intra-row placement. Notice that the objectives of Abacus and our formulation are quite different. Our goal is to increase lowest total wire distance from pad to finger and finger to ball, while Abacus is to optimize movement of all cells within one row after global placement, respectively. Once when the finger locations are determined by intra-row placement, the entire finger placement framework is finished.

V. Experimental Results

We develop our framework using C++ programming language on a Linux Ubuntu18.04 workstation with 2.80 GHz Intel i7-7700U CPU and 8 GB memory. Our finger placement layout is visualized by using Gtk and Cairo libraries [12, 13]. There are 6 industrial designs in our experiments.

Table II compares the results of our framework to those of manual designs. Besides of Euclidean wire length and Manhattan distance, the overall layout affinities, and runtime are also reported. The angle affinity, denoted as A_a, is the cosine of two-vector angle. The initial point of the

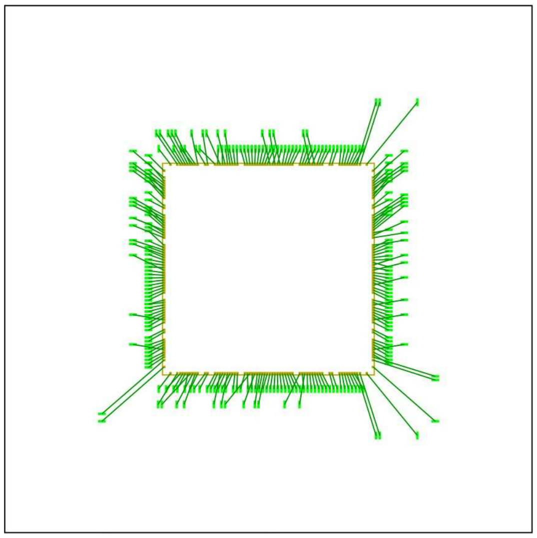


Fig. 9. Industrial 3 of WBFP

vector is the die pad. The terminal points of the two vectors are finger locations $WBFPV$ and $manualV$), respectively. Hence, the angle affinity reports the design similarity in terms of two-vector angle.

$$A_a = \frac{InnerProduct(WBFPV, manualV)}{|WBFPV||manualV|}$$

The length affinity, denoted as A_l, is the ratio of finger displacement to the maximum distance in the trapezium, (i.e., the boundary of each finger). The maximum distance is the maximum value of the diagonal (*trapzDiag*) and the longest side of the trapezium (*trapzLSide*). The finger locations determined by our finger placement framework and by experienced layout engineers are denoted as *WBFPF* and *manualF*, respectively. *trapzLSide*). Hence, the length affinity reports the design similarity in terms of wire length.

$$A_l = 1 - \frac{|WBFPF - manualF|}{MAX(trapzDiag, trapzLSide)}$$

In our experiment, the results of Industrial 1 to Industrial 3 achieve 12% to 50% distance reduction. Nevertheless, the results of industrial 4 and industrial 5 are inferior to those of manual designs. The reason is that during the legalization process, the initially optimized fingers were moved to reach design rule 1, thus sacrificed the best result. Figure 9 and 10 draws the finger placement results of 2 benchmark designs from industry, respectively.

VI. Conclusion

We have proposed a framework to tackle bond finger placement problem. In order to obtain a global view, a

TABLE II
Experiment result of similarity and distance

Benchmark	Euclidean distance (μm)			Manhattan distance (μm)			Affinities		Runtime
	WBFP	Manual	Imp.	WBFP	Manual	Imp.	A_a	A_l	
Industrial 1	105099	119711	12.2%	129584	194678	33.4%	97.5%	96.5%	0.5s
Industrial 2	174757	208093	16.0%	212164	426010	50.1%	92.2%	96.5%	1.3s
Industrial 3	778238	958131	18.7%	993231	1221930	18.7%	97.4%	86.0%	2.5s
Industrial 4	810251	728718	-11.1%	992839	1030370	3.6%	93.2%	92.9%	4.4s
Industrial 5	2198840	2170720	-1.2%	2731780	2929010	6.7%	90.7%	88.6%	6.4s
Industrial 6	1791910	1802230	0.5%	2299500	3175900	27.5%	93.5%	88.9%	5.7s
Average			5.9%			23.3%	94.1%	90.6%	3.5s

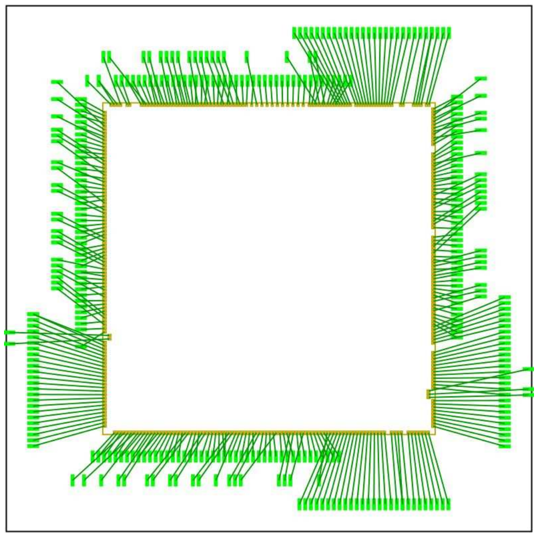


Fig. 10. Industrial 4 of WBFP

network-flow-based algorithm is used to minimize pad-to-finger and finger-to-ball distance. After that, inter-row and intra-row legalization and optimization stages are proposed to enforce wire-bond design rules with minimal displacement compared to the initial global placement. The finger placement of our proposed framework provides a quick and legalized reference for substrate layout engineers to start with. The experimental results demonstrate the potential of this framework in terms of competitive solution quality and design time reduction.

References

[1] N. H. E. Weste, D. M. Harris, *Integrated Circuit Design*, 2011.

[2] *Minimum-cost flow - Successive shortest path algorithm*, Available: `https://cp-algorithms.com/graph/min_cost_flow.html`

[3] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, *Introduction to Algorithms, 3rd Edition*, 2009.

[4] Peter Spindler, Ulf Schlichtmann and Frank M. Johannes, *Abacus: Fast Legalization of Standard Cell Circuits with Minimal Movement*, Available: `https://dl.acm.org/doi/10.1145/1353629.1353640`

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

[6] *Electronic design automation*, Available: `https://en.wikipedia.org/wiki/Electronic_design_automation`

[7] *Wire bonding*, Available: `https://en.wikipedia.org/wiki/Wire_bonding`

[8] Junko Yoshida, *Wanted: Process Engineers Versed in Packaging*, 2011, Available:*https://www.eetimes.com/wanted-process-engineers-versed-in-packaging.html*

[9] David White, Cadence *AI Challenges for Next-Gen EDA*, 2019, Available: `https://www.eetimes.com/ai-challenges-for-next-gen-eda/`

[10] Don Scansen *Chiplets, Gum and other Things that Stick*, 2020, Available: `https://www.eetimes.com/chiplets-gum-and-other-things-that-stick/`

[11] Ravi Mahajan, Intel *The Case for Heterogeneous Integration*, 2018, Available: *https://www.eetimes.com/the-case-for-heterogeneous-integration/*

[12] *The GTK Project - A free and open-source cross-platform widget toolkit*, Available: `https://www.gtk.org/`

[13] *Cairo Tutorial*, Available: `https://www.cairographics.org/tutorial/`