Universal Methodology to Handle Differential Pairs During Pin Assignment

Tilo Meister[1], Jens Lienig[1], Gisbert Thomke[2]

[1]Dresden University of Technology, Dresden, Germany
[2]IBM Research and Development, Böblingen, Germany

Abstract. Differential signaling has been a major challenge in design automation. The routing of differential pairs requires a suitable pin assignment of the respective nets. However, current automatic pin assignment algorithms lack the ability to consider differential pairs. We present a methodology to include differential pairs during pin assignment. Our solution can be applied to automatic or manual pin assignment processes without changing the methodologies already in place. This universality is achieved by using any established pin assignment approach as a black box, which is extended by pre and post processing steps. Extensive studies in industrial design flows show that our differential pair methodology does not compromise pin assignment quality with the added benefit of effective differential pair allocations.

Introduction

Differential pairs are a common challenge during digital and analog/mixed-signal layout generation of modern electronic devices. The challenge is to route as closely together as possible a pair of wiring paths (the so-called *differential pair*) in order to improve the routing solution. The resulting routing geometry provides significantly better electrical characteristics than single ended signaling. For example, interference identically captured by both routing paths is filtered out. However, the routing of differential pairs requires an adequate pin assignment that has to be generated beforehand.

The pin assignment of a component, such as a chip, is the assignment of its I/O signals to its I/O pins, often referred to as pads (Fig. 1). Usually, this pin assignment is created after components are placed on the wiring substrate, such as a printed circuit board (PCB) or a multi chip module (MCM). Optimizing this pin assignment is a crucial stage because the routability of the substrate largely depends on both pin assignment and component placement. Due to rising I/O counts, the routability challenge has continued to increase rapidly in recent years, which puts enormous pressure on a well-performed pin assignment.

Furthermore, there has been a growing demand for differential pairs, which have to be considered during this stage. This is mostly due to more stringent electrical requirements of signals in modern applications. However, to the best of our knowledge, none of the published pin assignment approaches considers the implementation of differential pairs.

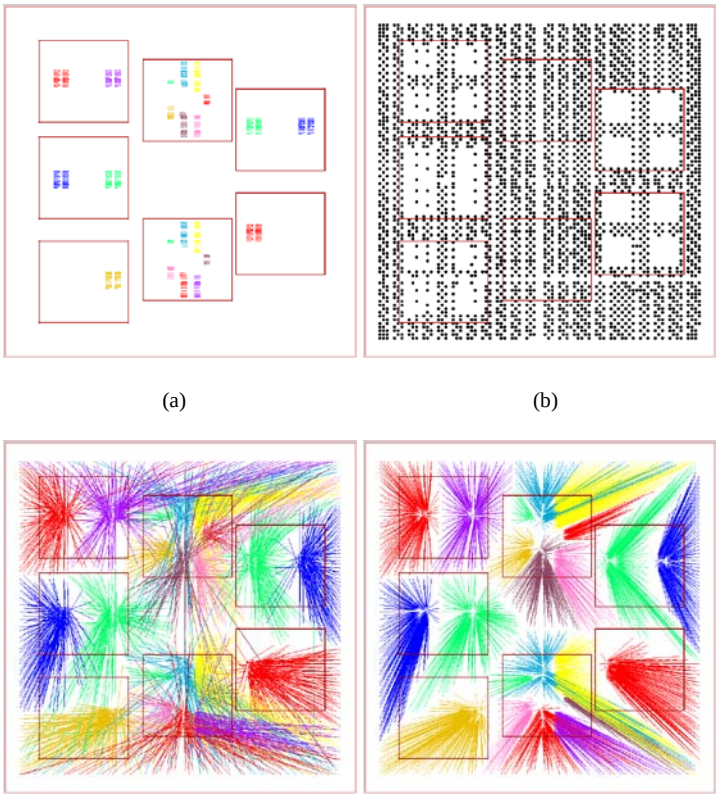


Fig. 1. Illustration of pin assignment with the I/O pins of seven chips (a) to be assigned to signals that connect a wiring substrate such as an MCM (b). The pin assignment based on the shortest Manhattan distances of the individual connections is depicted in (c) whereas (d) illustrates the pin assignment with minimum overall length of all Euclidean distances (*flylines*).

This chapter presents a universal methodology to extend pin assignment algorithms to consider differential pairs. This methodology requires no significant changes to the basic pin assignment algorithm, thereby respecting any individual pin assignment routines already in use. As shown below, this add-on approach has almost no impact on the quality of the created pin assignments while at the same time efficiently considering all differential pair requirements. Furthermore, the algorithm can be used for any given percentage (from zero to 100%) of differential pairs among the nets to be considered during pin assignment. As such, it allows a flexible inclusion of differential pair requirements in digital and analog/mixed-signal real-world design flows.

The remainder of this chapter is organized as follows. Pin assignment and differential pairs are introduced in the following two sections. The differential pair methodology is proposed in the section thereafter. The effectiveness of the proposed methodology is proven in the section presenting experimental results. At the end of this chapter, we present limitations of our approach, an outlook, and conclusions.

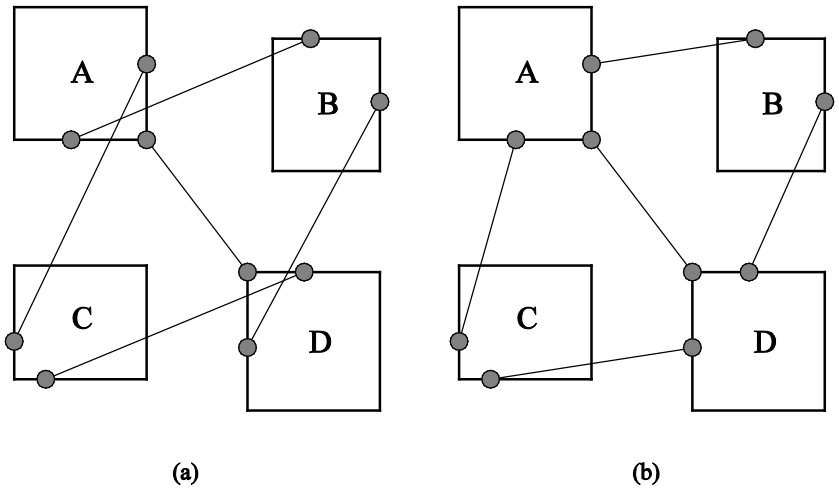


Fig. 2. Example with four components A-D to illustrate the influence of pin assignment on the routability. Pins are marked by circles ● on the outline of the components. Nets are shown as flylines. If no wiring is allowed below components A through D, the design with pin assignment (a) is not routable in one layer, whereas (b) allows single layer routing.

The Pin Assignment Problem

During logic design, logical pins are defined to be the signal interface between the different components of a design. During the subsequently performed layout synthesis, these logical pins, and thus the associated signals, have to be mapped to real, physical pins, which serve as the actual electrical joints between the components.

This mapping of logical pins (signals) to physical pins is called *pin assignment* and has great influence on the routability, electrical characteristics and the cost of the design (see example in Fig. 2). Hence, the objective of pin assignment is to assign signals to physical pins such that these circuit characteristics are fulfilled best for the individual designs.

Pin assignment has been studied for all system levels such as digital and analog/mixed-signal circuits (ICs), MCMs and PCBs. For ICs, the pin assignment of macro blocks is usually optimized with regard to routability during placement [3][4], buffer planning [5] or routing [6]. Pin assignment approaches for PCBs and MCMs can be found in [7][8][9][10].

Context

Pin assignment is closely related to both component placement and routing. All three design steps have in common that their individual optimal solutions depend on each other. Finding the overall optimal solution for these design steps would require incorporating them into one optimization task. Due to complexity and the related NP-

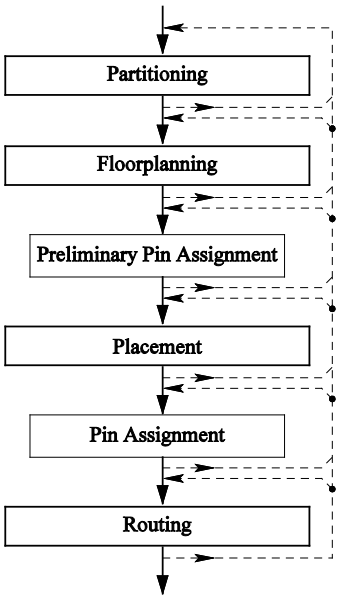


Fig. 3. Simplified physical design flow. The dashed arrows indicate iterations over the individual design steps.

hardness of physical design, this is unfeasible. Hence, the repeated sequential execution of these design steps is currently the only accomplishable approach to physical design.

Fig. 3 shows the major steps of the physical design flow [14][17], including pin assignment steps, of the design of electronic devices. Of these steps, placement is the first stage that requires an exact pin assignment, because the target function of placement depends on wire lengths and routing congestion. At this point a typical dilemma of physical design becomes obvious. The objectives of pin assignment are a minimal wire length and minimal routing congestion, which cannot be computed before determining the component placement. At the same time, placement depends on the chosen pin assignment. To come around this paradox either pin assignment has to be incorporated into placement [15] or a preliminary pin assignment has to be chosen before component placement is being optimized. Such a preliminary pin assignment is usually based on heuristics and the experience of designers and allows computing an optimized component placement.

Having optimized the placement for a specific preliminary pin assignment, it is then possible to improve the pin assignment for this optimized placement. To further improve design quality, it is possible to go back (one or more iterations) and revise the placement solution based on the optimized pin assignment (see Fig. 3).

A similar interdependency exists for pin assignment and routing. Routing largely depends on placement and pin assignment. Unfortunately, only after routing has been completed, which is extremely time consuming, it is possible to ultimately judge the quality of placement and pin assignment. Therefore, good estimates of the routability are essential for effective pin assignment algorithms.

It is further possible to integrate pin assignment into the global [16] and/or detailed routing phase. By integrating pin assignment into global routing, it can be adapted to the global requirements of routing, whereas a combination with detailed routing would support local adjustments of the pin assignment.

Pin Assignment Algorithms Used in This Work

We use four pin assignment algorithms to evaluate the differential pair methodology presented. Three of them are heuristics, which either reduce signal intersections or balance the lengths of nets within a bus. The fourth algorithm analytically minimizes net lengths and the number of signal intersections. All four algorithms assume that pin assignment is done in-between placement and routing (see Fig. 3). The details of the four pin assignment algorithms are described in [7].

By using these four algorithms in various configurations, we obtain seven different pin assignment procedures in order to evaluate the presented differential pair methodology. Specifically, the analytical algorithm can be utilized with different parameters to its cost function. Also, one of the heuristic algorithms can be used to modify pin assignment results of the remaining three algorithms.

Differential Pairs

A differential pair are two wires which are routed close together, have matched electrical characteristics, and are used to transmit one signal. This signal is encoded in the voltage difference between both wires. Differential pairs are essential for many electronic devices, because differential signaling has superior electrical characteristics to single ended signaling [1][2]. In particular, differential signaling leads to lower crosstalk and lower electromagnetic interference. Both noise emission and noise acceptance are minimized by differential pairs if both (1) the distance between the two routing paths is minimal and (2) the lengths and electrical characteristics of both paths are matched.

The basic functional principle of a differential pair is shown in Fig. 4. The differential sender encodes the signal $S = u(t)$ into the difference of two complementary signals $a \cdot S = u_p(t)$ and $-a \cdot S = u_n(t)$ propagating along the two routing paths n and p. Where a is the gain of the differential sender. If both routing paths have the same electrical characteristics and are routed close together, captured noise A can be presumed to be identical for both signals $u_{pA}(t) = u_p(t) + A$ and $u_{nA}(t) = u_n(t) + A$. The signals $u_{pA}(t)$ and $u_{nA}(t)$ are then translated back to the original signal S by subtracting $u_{pA}(t) - u_{nA}(t) = S_{rcv} = 2 \cdot a \cdot S$ which at this point filters out any noise A identically captured along both paths.

In case routing paths n and p are not routed close together and/or have different electrical properties, both tracks capture noise differently A_n and A_p leaving the received signal $S_{rcv} = 2 \cdot a \cdot S + (A_p - A_n)$ distorted with noise $(A_p - A_n)$. Additionally, if electrical characteristics of the tracks differ, propagation delays of the n- and p-signal may be different, resulting in a distortion of the transmitted signal as shown in Fig. 5.

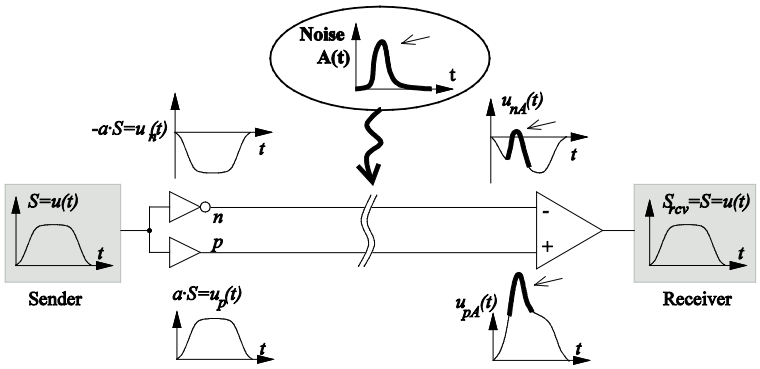


Fig. 4. Functional principle of a differential pair.

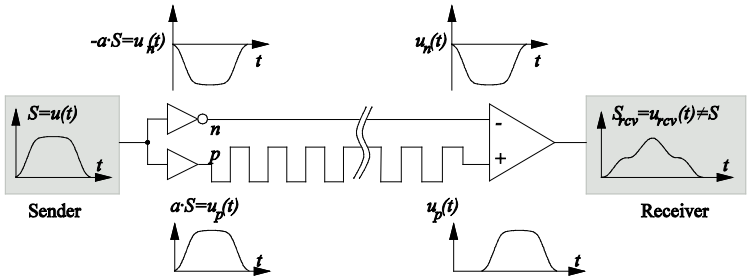


Fig. 5. Effect of unmatched propagation delays of a differential pair.

Assuming a signal frequency of 3GHz, a timing difference between signals *n* and *p* of only 167 ps shifts signals by half a clock. In a FR4 printed circuit board that timing difference is equal to a difference in wiring lengths of roughly 2.5cm. That is, the tolerance for propagation delays and wiring length differences for a differential pair at 3 GHz is in the domain of picoseconds and millimeters respectively.

The special wiring geometry of differential pairs requires suitable pin assignments. Specifically, for the two nets *n* and *p* of a differential pair, the pin assignment has to be chosen such that each pin of the routing path *n* has a so-called *parallel pin* at the same distance from the sender in the routing path *p* and vice versa. The distance be-

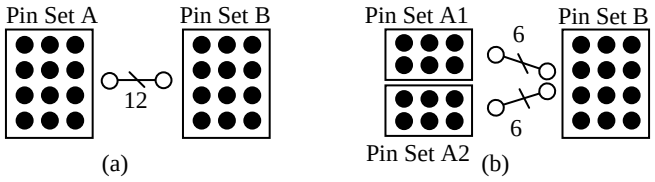


Fig. 6. Two pin assignment tasks for 12 two terminal nets. (a) 12 nets need an assignment to pins of sets A and B. (b) The nets of set B are to be assigned to two sets A1 and A2 and vice versa.

tween those parallel pins must not exceed a maximum distance d_{max}. This parameter is technology-dependent and for MCMs and PCBs usually ranges from one to two times the pin grid. For the sake of simplicity, we call the parallel pins of a differential pair a ***d**ifferential **p**in **p**air* (***DPP***). If the distance between the pins of the DPP is not greater than $d_{max,}$ we call it a *valid* DPP, else it is labeled an *invalid* DPP.

Differential Pair Methodology

In this section, we present our novel methodology to handle differential pairs during pin assignment. Our approach is used as an extension of any automatic or manual procedure in place that solves the pin assignment problem (Fig. 6 shows two example problems). The underlying basic pin assignment procedures, which are to be extended, are labeled ***PAA*** (**p**in **a**ssignment **a**lgorithm) throughout this paper.

Overview of the Algorithm

Our approach can be summarized in five steps (Fig. 7).

1. First, a transformation is applied to the original pins. This transformation embeds data about valid DPPs. We call the transformed pins *fat pins*.
2. Second, the PAA in place is applied to these fat pins.

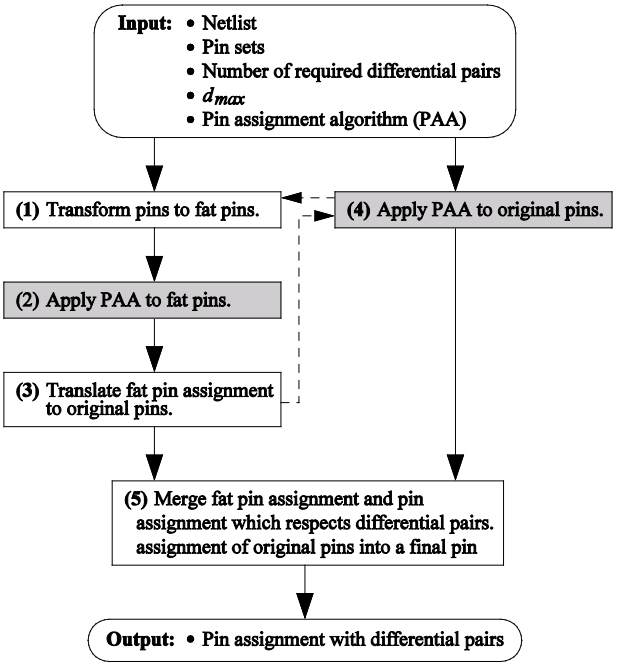


Fig. 7. Overview of our differential pair methodology.

3. Third, the pin assignment for the fat pins (*fat pin assignment*) is split up to the original pins. This back transformation returns a pin assignment only for a subset of the pins and nets.
4. Therefore in the fourth step, a pin assignment without differential pairs is created for the remaining unassigned nets with the same PAA as applied in the second step.
5. Finally, the two interim pin assignments created in steps (3) and (4) are merged into one final pin assignment, which respects all constraints of both the pin assignment problem and differential pairs.

This methodology is a framework that allows considering any number of differential pairs by utilizing any existing pin assignment algorithm (see above) without the need to modify the existing pin assignment algorithm itself. Steps (1), (3) and (5) are pre and post processing steps (white boxes in Fig. 7), while any already existing pin assignment procedure PAA can be plugged-in at steps (2) and (4) (gray boxes in Fig. 7).

The inputs for this framework are the netlist, the sets of pins, and an existing pin assignment algorithm. In addition, the designer specifies d_{max} for each set of pins and the number of differential pairs. The output is a pin assignment for all nets, which respects the constraints for as many differential pairs as specified by the designer. This pin assignment is topologically very similar to a pin assignment created by the basic pin assignment algorithm (PAA) alone.

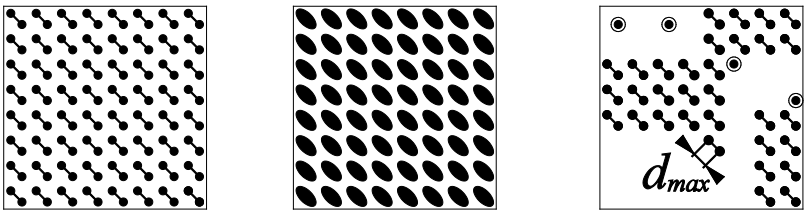
The individual steps as well as the indicated interactions (dashed arrows in Fig. 7) are presented in the following three subsections.

Combine Pin Pairs to Fat Pins

In order to generate the so-called fat pins (Step 1 in Fig. 7), valid DPPs are automatically determined among the original pins. This automatic selection of DPPs may be controlled by the designer by manually specifying an arbitrary number of DPPs. As outlined in this subsection, pins that cannot be combined to a valid DPP either ignored or are paired to invalid DPPs. As described above, these so-called invalid pin pairs cannot be used for differential signals in the final pin assignment. Nonetheless, allowing invalid DPPs at this point has a significant impact on the quality of the final pin assignment with differential pairs. The section presenting the experimental results (see below) shows the influence of invalid DPPs on the final pin assignment.

A maximum weighted matching (as shown in [11]) has to be calculated to find automatically as many DPPs as possible, with the least distance between the pins of the individual pairs. The implementation presented in [12] has a complexity of $O(p^3)$ (p number of pins). However, components with differential pairs have well-suited pin configurations such that DPPs can be determined effectively by heuristic, greedy algorithms. Therefore, we have developed two greedy algorithms, which are more time efficient than the slower optimal algorithms presented in [11][12].

The first algorithm (MOST_PAIRS) creates as many pin pairs as possible. The second algorithm (PREFERRED_PAIRS) focuses on pairs whose two pins are closest. The complexity of both algorithms is defined by the sorting algorithm, which is used to sort pins according to their distance to so-called *partner pins* and by the num-



(a) (b) (c)

Fig. 8. (a) Pin pairs of a small area array component. The black dots denote pins, the line between two pins represents a pin pair. (b) Fat pins created from the selected pin pairs. (c) Pin configuration which prevents some pins (circled) to be used for differential pairs.

ber of partner pins, respectively. Thereby, partner pins of one pin are those that are no further away than d_{max}. We use insertion sort, which has a complexity of $O(p^2)$ in the worst case. Still, the practical efficiency is much better since many pin pairs are of the same distance and most pins have the same number of partner pins.

Both algorithms first locate the next pin to be paired. In MOST_PAIRS, this is the pin with the least number of valid partner pins (distance $\leq d_{max}$) but at least one partner pin. In PREFFERED_PAIRS, it is the pin that has a valid partner pin that is closest amongst all possible pairs of pins. The located pin and its closest partner pin are then paired. This is repeated until no more pins can be paired. Fig. 8 (a) shows the automatically selected pin pairs for a small area array component.

As depicted in Fig. 8 (c), there may be pins, which cannot be paired to valid DPPs. Either those pins are ignored or they are paired to invalid DPPs by the same two strategies described above thereby ignoring d_{max}. Thus, we can create four different selections of pin pairs, which eventually lead to different pin assignments with differential pairs:

- PREFERRED_PAIRS with only valid DPPs
- PREFERRED_PAIRS with valid and invalid DPPs
- MOST_PAIRS with only valid DPPs
- MOST_PAIRS with valid and invalid DPPs

Which of the four variants are used depends on the number of differential pairs required (see Section Integrating Fat Pin Assignment with PAA).

Next, a *fat pin* is created for each computed pin pair, regardless whether it is valid or invalid. The coordinate of a fat pin is the arithmetic mean of the coordinates of its original two pins (see Fig. 8 b). Except for its coordinates, the new fat pin inherits all characteristics, such as design rules, from the two original pins. At the same time, specific nets are combined in order to ensure an identical number of nets and fat pins.

Fat Pin Assignment

Following fat pin creation, all fat pins are treated just like conventional pins and are fed to any PAA that solves the pin assignment problem (Step 2 in Fig. 7). The result-

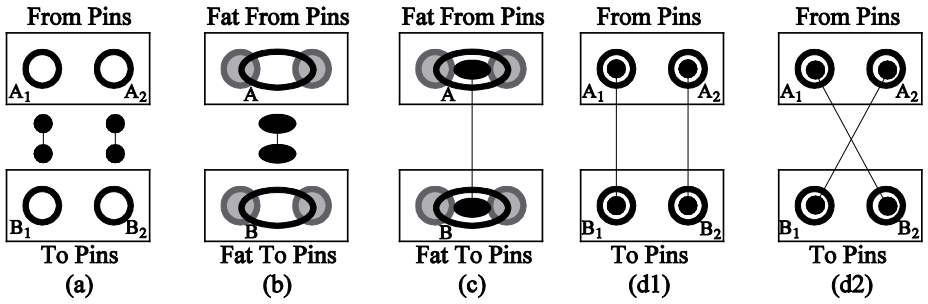


Fig. 9. Fat pin transformation and inverse transformation. (a) Pin assignment task for two nets. (b) Transformation from pins to fat pins. (c) Fat pin assignment. (d1) First alternative for inverse transformation. (d2) Second alternative for inverse transformation.

ing fat pin assignment is consequently transformed back to specify the assignment for the individual pins (Step 3 in Fig. 7).

The transformations illustrated in Fig. 9 are applied to each pin pair: Fig. 9 (a) shows the pin assignment task for two nets (lines) with two pins each (ending dots). A_1, A_2, B_1 and B_2 are the pins that are arranged in two separate sets. A_1 and A_2 are in the pin set named "From". B_1 and B_2 are in the pin set named "To". In Fig. 9 (b) pins A_1, A_2, B_1, and B_2 are transformed to fat pins A and B. Thus, only one of the two nets remains. Fig. 9 (c) shows the fat pin assignment by applying a PAA to the fat pin sets. Fig. 9 (d1) and (d2) denotes the two possibilities for the subsequent inverse transformation. Either pins A_1 and B_1 (Fig. 9.d1) or pins A_1 and B_2 (Fig. 9.d2) are assigned to the same net. We select the configuration with the smaller difference in the individual lengths and the shortest overall length of the flylines of both nets (which is (d1) in this example). This choice supports the matching of the net lengths of a differential pair.

If fat pins A and B are valid fat pins (A_1 and A_2, as well as B_1 and B_2, respectively, are no further apart than d_{max}), the two nets *can* be used for *either* a differential pair *or* for two single ended signals. Consequently, the number of nets which have all their pins assigned to valid fat pins defines the number of *possible differential pairs* in the final pin assignment because they can, *but need not*, be used as differential pairs.

Integrating Fat Pin Assignment with PAA

All unpaired pins and dropped nets are ignored and do not receive a pin assignment during fat pin assignment (Steps 1–3 in Fig. 7 and as described in the previous two subsections). To find the pin assignment for those pins and nets, the basic PAA is applied to original pins and nets (Step 4). The thus created pin assignment is integrated with the fat pin assignment to determine the final pin assignment with differential pairs (Step 5).

We propose two methods to integrate the two interim pin assignments. *Aggressive blending* creates more possible differential pairs than *defensive blending*, yet the results of defensive blending are better with respect to the objective function of the underlying PAA. Both methods can be used in combination with any of the four differ-

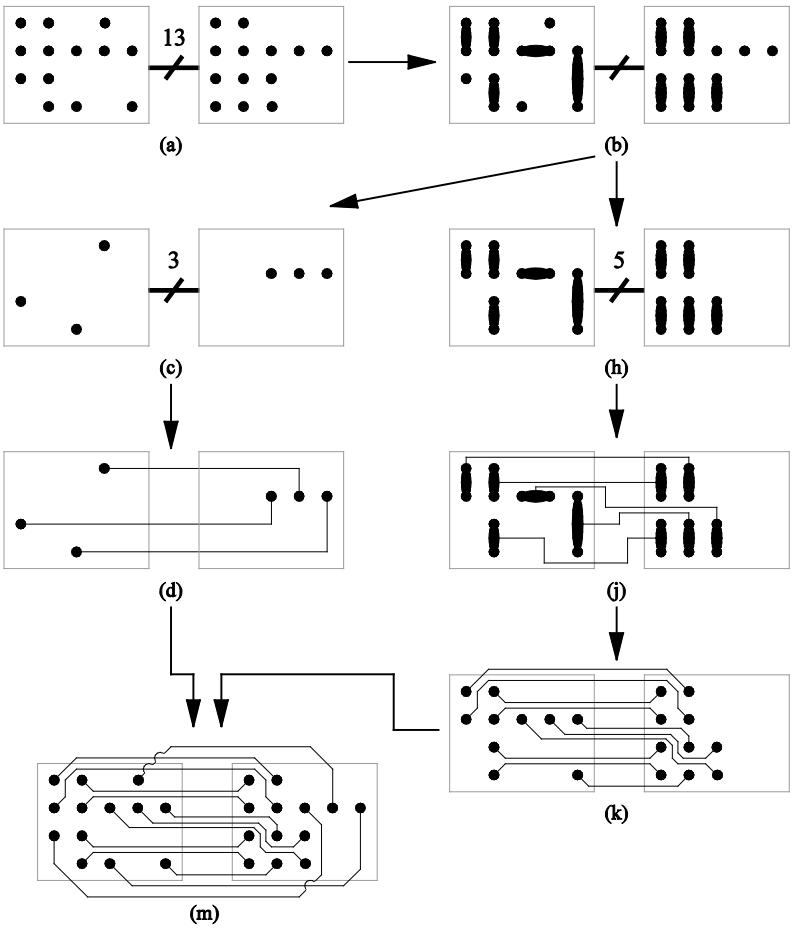
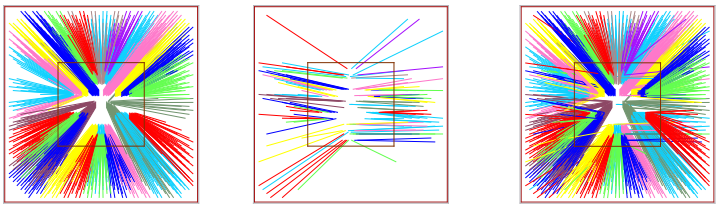


Fig. 10. Example pin assignment procedure with differential pairs using aggressive blending. (a) Pin assignment task with differential pairs for 13 nets. (b) Automatically selected fat pins. (c) Pins that were not paired to fat pins during (b). (d) Basic pin assignment for unpaired pins. (h) Automatically selected fat pins (leftover pins omitted). (j) Fat pin assignment. (k) Back-transformation of fat pin assignment to original pins. (m) The final pin assignment with differential pairs is the combination of the basic pin assignment (d) and the fat pin assignment (k).

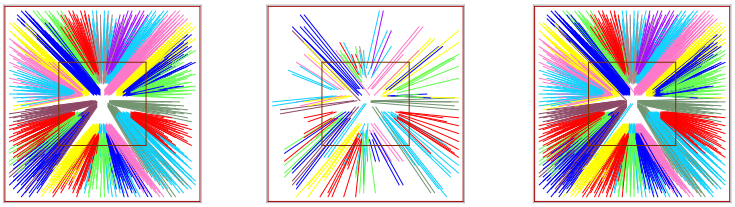
ent methods to select pin pairs (see above), all together resulting in eight different pin assignments with differential pairs.

Aggressive Blending To determine the pin assignment for all pins that did not receive a fat pin assignment (Fig. 10 c), the basic PAA is applied to these pins and nets (Fig. 10 d). The final pin assignment (Fig. 10 m) with differential pairs results from the combination of the fat pin assignment (see above and Fig. 10 k) with the pin assignment created by applying the PAA to the leftover pins and nets (Fig. 10 d). For aggressive blending, the fat pin assignment is applied to all pins that were paired,



(a) (b) (c)

Fig. 11. Pin assignment for a single chip module using aggressive blending. (a) Fat pin assignment. (b) Assignment of remaining pins and nets. (c) The final pin assignment with 468 possible differential pairs is the combination of (a) and (b).



(a) (b) (c)

Fig. 12. Pin assignment for a single chip module using defensive blending. (a) Fat pin assignment. (b) Assignment of remaining pins and nets. (c) The final pin assignment with 454 possible differential pairs is the combination of (a) and (b).

while the basic PAA is limited to the remaining pins and nets and is not aware of the fat pin assignment already created. Fig. 7 shows the flow of this algorithm. The limitation of the basic PAA to pins without a fat pin assignment is indicated by the dashed arrow pointing from step 3 to step 4. A step-by-step example of pin assignment using aggressive blending is shown in Fig. 10.

Compared to defensive blending (described in the following subsection), the resulting pin assignment is of lower quality with respect to the objective function of the PAA, because the topologies of the two interim pin assignments differ in general. However, their better topological similarity during defensive blending results in fewer possible differential pairs, as shown in Figs. 11 and 12 and described in the following subsection.

Defensive Blending Defensive blending is an iterative process to improve the integratability of the fat pin assignment by *incrementally* adapting the selection of differential pin pairs. The advantage of defensive blending, in contrast to aggressive blending, is that all pins and nets are considered during the creation of the basic pin assignment. However, fewer pins are combined to fat pins. Compared to aggressive blending, this yields a better final pin assignment with respect to the basic objective function at the cost of decreasing the number of possible differential pairs in the final pin assignment.

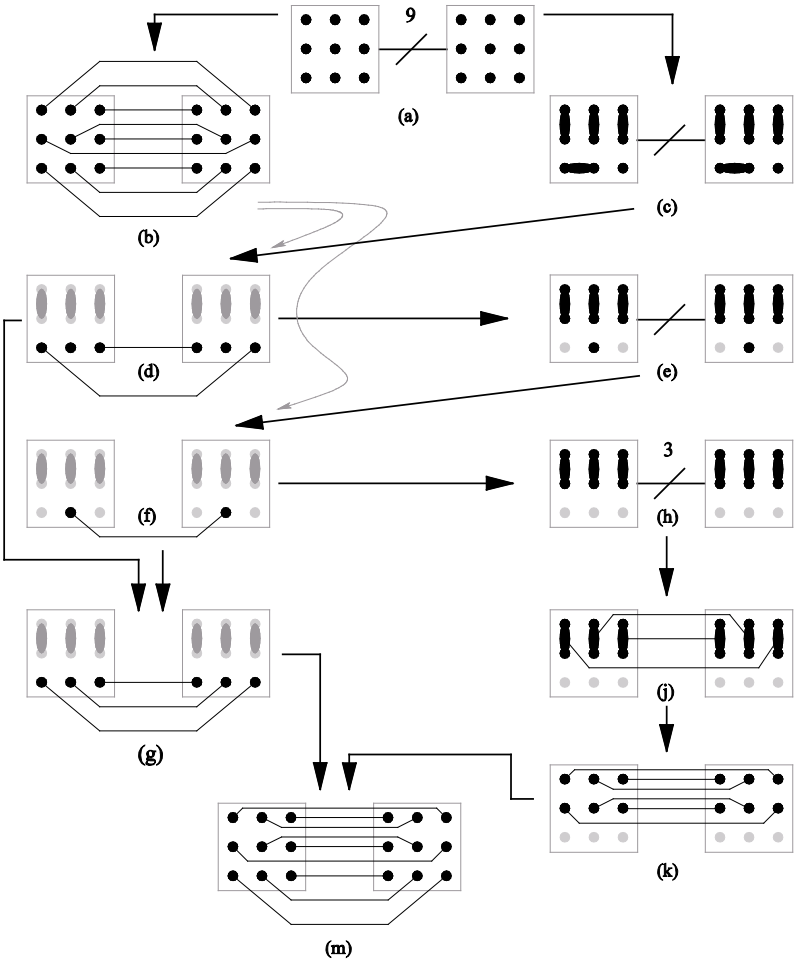


Fig. 13. Using defensive blending to obtain differential pairs of two 9x9 pin arrays. (a) Pin assignment task with differential pairs. (b) Basic pin assignment (PAA) without differential pairs. (c) Automatically selected fat pins. (d) Basic pin assignment for pins that were not paired to fat pins during (c). (e) Newly selected fat pins (pins with an assignment from (d) are ignored). (f) Basic pin assignment for pins that were not paired to fat pins during (e). (h) Newly selected fat pins (pins with an assignment from (d) or (f) are ignored). (j) Create fat pin assignment, since all remaining pins were paired to fat pins. (k) Back-transformation of fat pin assignment to original pins. (g) Pins with basic pin assignments from (d) and (f). (m) Final pin assignment with differential pairs is the combination of the basic pin assignment (g) and fat pin assignment (k). In this example the final pin assignment with differential pairs contains three pairs of nets usable for differential pairs, which were assigned during fat pin assignment (see (k)). It contains one more net pair that happens to be usable as differential pair, which was assigned during the basic PAA (see (g)).

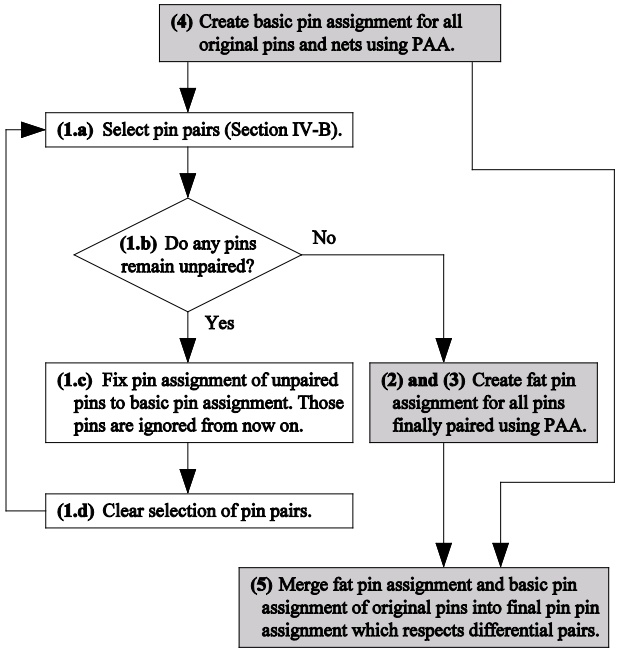


Fig. 14. Major steps of pin assignment with differential pairs using defensive blending. This figure extends Fig. 7 in which the interaction of the basic pin assignment and the process of selecting pin pairs is indicated as a dashed arrow from step 4 to step 1.

In a first step, the basic PAA is applied to the original pins and nets (Fig. 13b). This pin assignment is then used as a reference throughout the following iterations. Next, pin pairs are selected as described above (Fig. 13c). Subsequently, all pins that have not been paired receive their pin assignment from the reference pin assignment of the first step (Fig. 10d). The pin assignment of those unpaired pins is final and is never changed again. For all remaining unassigned pins, the current selection of pairs is discarded and recreated (Fig. 13e) in order to optimize the selection. This process is repeated until all pins either received their final pin assignment or are paired (Fig. 10f and 10h).

All pins that are finally paired undergo fat pin assignment, and are then transformed back to their original pins (see section on fat pin assignment above, Fig. 13j and 13k). Hence, in defensive blending, the final pin assignment results from the combination of the reference pin assignment for all finally unpaired pins and the back transformation of the fat pin assignment (Fig. 13m). Fig. 14 shows the flowchart of the defensive blending method.

Fig. 12 shows the two interim pin assignments (a) and (b) and the final pin assignment (c) created with defensive blending for the same single chip module as in Fig. 11.

Defensive blending and aggressive blending do not differ and give identical results in case all pins are paired to fat pins during the first iteration.

Summary

The eight possible combinations of methods for selecting fat pins and methods for integrating the interim pin assignments yield eight different pin assignments with differential pairs. They vary in the number of possible differential pairs and in the magnitude of changes compared to the basic pin assignment without differential pairs.

The number of possible differential pairs of each variant cannot be predicted exactly. Yet, experimental results show that the different variants can be ranked with respect to their quality and the number of possible differential pairs. In general, the quality of the pin assignment deteriorates with an increase in possible differential pairs. Therefore, the best pin assignment for a specific design is the one with just enough possible differential pairs. We find this pin assignment by sequentially applying the different variants starting with the one that creates best pin assignment results while providing the least differential pairs. Subsequently, pin assignment variants with more and more differential pairs are created, until the best pin assignment for the design is found.

Experimental Results

The effectiveness of the presented methodology is proven by comparing pin assignments with differential pairs to those without differential pairs. First, results from PAAs (without differential pairs) applied to industrial designs are reported. Next, these PAAs are extended by the fat pin methodology to include differential pairs. The pin assignments are compared by means of *SHPWL*, *HPWL MATCH*, *AVG Flylines*, *STD Dev*, and the number of signal intersections.
If (x_{ai}, y_{ai}) and (x_{bi}, y_{bi}) are the coordinates of the two pins of net i, p is the number of nets in the pin assignment task and $dx_i = |x_{ai} - x_{bi}|$, $dy_i = |y_{ai} - y_{bi}|$, then the measurement metrics are defined as follows:

- *SHPWL*: The sum of the HPWLs (half perimeter wire lengths) of all nets.

$$SHPWL = \sum_{i}^{p} dx_i + dy_i$$

- *HPWL MATCH*: The additional length necessary to match the HPWL routing length of all nets. A lower value of *HPWL MATCH* indicates less routing effort, especially for busses.

$$HPWLMATCH = p \cdot max(dx_1 + dy_1, \ldots, dx_p + dy_p) - SHPWL$$

- *AVG Flylines*: The average net length in Euclidean geometry.

$$AVG\ Flylines = \frac{1}{p} \sum_{i}^{p} \sqrt{dx_i^2 + dy_i^2}$$

- *STD Dev*: The standard deviation of the net lengths in Euclidean geometry, which similarly to *HPWL MATCH* evaluates the expected wiring effort necessary to match wiring lengths.

$$STD\ Dev = \frac{1}{p-1}\sqrt{\sum_{i}^{p}\left(AVG\ Flylines - \sqrt{dx_i^2 + dy_i^2}\right)^2}$$

- The number of signal intersections is calculated as the number of intersections within the flylines of all nets.

In the following subsection, the differential pair methodology is compared with regular PAAs using the above metrics. In the subsection after the following, an investigation of the four proposed fat pin variants and the two proposed merging strategies is presented.

Quality of Fat Pin Methodology

The results presented in Table 1 are taken from a commercially fabricated IBM single chip module (SCM) that carries one die on top and is covered with a regular array of pins on the bottom side (1058 signal pins, 1058 power/ground pins). The pin assignment algorithms are extended by our fat pin methodology and used to create an assignment with differential pairs of die signal pins to bottom signal pins.

The used PAAs have the following objectives (a detailed description of these algorithms can be found in [7]):

1. Heuristic to minimize *HPWL MATCH* and *STD Dev*
2. Heuristic to minimize signal intersections within busses for a specified direction of fanout.
3. Same as 1. with subsequent removal of signal intersections.
4. Same as 2. with subsequent removal of signal intersections.
5. Minimum *AVG Flylines*.
6. Minimum *SHPWL*.
7. Concurrent minimization of *SHPWL* and signal intersections.

All pins of design SCM are transformed to valid fat pins by the PREFERRED_PAIRS algorithm accordingly to Fig. 8 (a, b). As a result, the final pin assignment is completely defined by the fat pin assignment and no merging of interim pin assignments is necessary. In addition, the creation of fat pins by the MOST_PAIRS algorithm returns identical results. Hence, there are two relevant pin assignments with differential pairs for each PAA. Firstly, the PAA unintentionally allows for a significant number of differential pairs. Those pairs result from *parallel pins* (see section on differential pairs) with a distance smaller than d_{max} (d_{max} is equal to the diagonal pin grid in our experiments, Fig. 8 c). Secondly, the pin assignment created by fat pins allows all nets to be used as differential pairs.

For each PAA 1–7, Table 1 compares the pin assignment created by the basic PAA and its differential pair extension. Absolute values are given for the number of possible differential pairs (#Diff Pairs), intersections of flylines and the runtime. For measures *SHPWL*, *HPWL MATCH*, *AVG Flylines* and *STD Dev* the pin assignment results with differential pairs are given as a the percentaged difference (Δ) to the respective result of the basic PAA. Table 1 shows that the impact of fat pins on the objectives of the basic PAAs is marginal. One exception are signal intersections estimated as inter-

Table 1. Experimental pin assignment results of design SCM without and with differential pairs (/o | w/ DP) using the seven PAAs 1–7 with different objectives, as listed in the text. Percentage values denote the difference between the basic PAA and its differential pair extension with positive percentages indicating an increase of the respective value.

PAAs	#Diff Pairs (/o | w/ DP)	Δ*SHPWL*	Δ*HPWL MATCH*	Δ *AVG Flylines*	Δ *STD Dev*	Intersect. of Flylines (/o | w/ DP)	Runtime in s (/o | w/ DP)
1.	367 | 529	+0.17%	+1.3%	+0.17%	+0.34 %	6159 | 7000	<1 |<1
2.	160 | 529	-0.52%	-6.4%	+0.70%	-2.4 %	44686 | 46331	<1 |<1
3.	313 | 529	+0.17%	-6.7%	+0.18%	+1.0 %	0 | 1633	1 |<1
4.	294 | 529	+0.00%	-3.2%	+0.01%	-0.57 %	0 | 1551	<1 |<1
5.	283 | 529	+0.13%	+1.6%	+0.12%	-0.28 %	80 | 1619	8 | 1
6.	93 | 529	+0.28%	+5.2%	-0.02%	-0.92 %	27955 | 27212	3 |<1
7.	298 | 529	+0.13%	-6.1%	+0.09%	-0.29 %	0 | 1564	10 | 1
Absolute Average	258 | 529	0.20%	4.4%	0.18%	0.83 %	11269 | 12416	3 |<1

sections of flylines, which increased considerably. Yet, closer inspection shows that intersections are introduced in places where they can easily be resolved by the router because they are either close to the endpoints of nets or the intersecting nets are almost parallel, thereby not affecting routability.

Comparison of Fat Pin Variants

In order to compare the four different variants of selecting pin pairs (PREFFERED_PAIRS without invalid DPPs, PREFFERED_PAIRS with invalid DPPs, MOST_PAIRS without invalid DPPs, and MOST_PAIRS with invalid DPPs) and the two merging strategies (aggressive blending and defensive blending), the results of a multi chip module (MCM) of an IBM industrial design are presented. This MCM has seven dies on top, 2930 signal pins and 2112 power/ground pins (see Fig. 1). The arrangement of the pins is irregular such that 68 of the signal pins cannot be used as differential pin pair because no other signal pin is closer than d_{max}. Additional 190 signal pins are not usable for differential pin pairs because these pins are in 190 “islands of pins” (which are further apart than d_{max}) with each having an odd number of pins (see Fig. 8 c).

The PAA 5, which minimizes the overall length of the flylines, is used to create the assignment of bottom signal pins (Fig. 1 b) to die signal pins (Fig. 1 a). The eight variants of the fat pin methodology (#1–#8 in Table 2) and the basic PAA alone (#0 in Table 2) deliver nine pin assignments with differential pairs. The results (Table 2, Figs. 15 and 16) show that along with an increasing number of available differential pairs, the length of the flylines, which is the objective of the used PAA 5, slightly increases. In six out of eight cases, the increase stayed below 0.25% (with no measurable increase in routing lengths when comparing the actual routing results with and without differential pairs). For variants #7 and #8 (see # in Table 2) the increase in lengths are 5.4% and 1.9%, which resulted in a similar increase in actual final routing length (Cadence SPECCTRA autorouter).

Table 2. Results of differential pair pin assignment of the eight different fat pin variants (#1–#8) and of the PAA 5 (#0) for design MCM. The used PAA 5 minimizes the overall length of the flylines. The names of the algorithms PREFERRED_PAIRS and MOST_PAIRS are abbreviated as PREF_P and MOST_P, respectively.

#	Blending Method	Invalid DPPs	Selection of Pin Pairs	Number of Diff. Pairs	*SHPWL*	*AVG Flylines*	*STD Dev*	Intersections of Flylines	Runtime in Sec
#0	n/a	n/a	*None*	376	46321	**11.84**	8.49	209	243
#1	defensive	no	PREF_P	237	46351	11.85	8.48	2365	245
#2			MOST_P	243	46350	11.85	8.48	2437	244
#3		yes	PREF_P	1127	46452	11.87	8.46	4120	277
#4			MOST_P	1081	46430	11.86	8.52	3661	266
#5	aggressive	yes	PREF_P	1143	46410	11.86	8.48	3808	34
#6			MOST_P	1217	46440	11.86	8.51	4052	32
#7		no	PREF_P	1251	48182	12.48	9.25	17008	20
#8			MOST_P	1336	47110	12.06	8.54	9750	23

The results show that aggressive blending (#5–#8) yields more differential pairs than defensive blending (#1–#4). Furthermore, the selection of pin pairs by MOST_PAIRS generally gives more differential pairs than PREFERRED_PAIRS.

The effect of invalid DPPs depends on the method of blending. For defensive blending, the number of created differential pairs is drastically increased by using invalid DPPs (variants #3 and #4), while the quality with respect to the basic objective

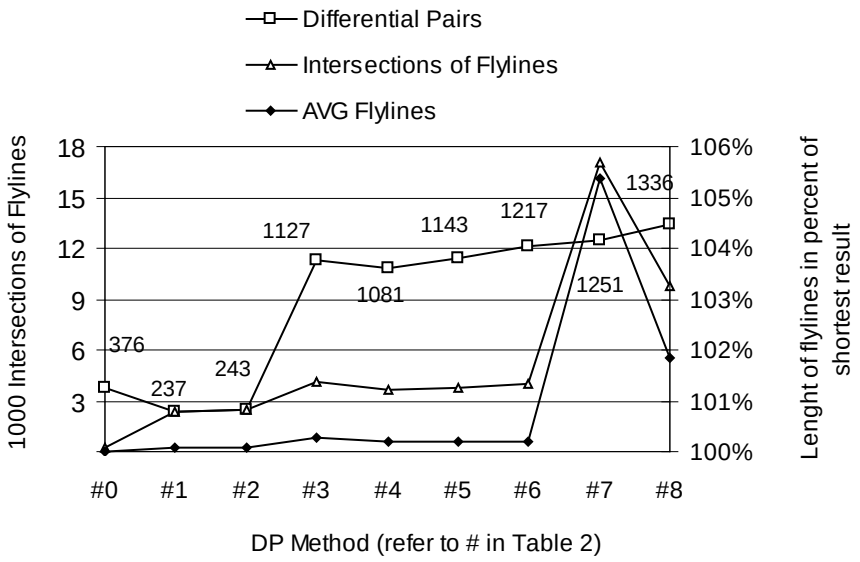
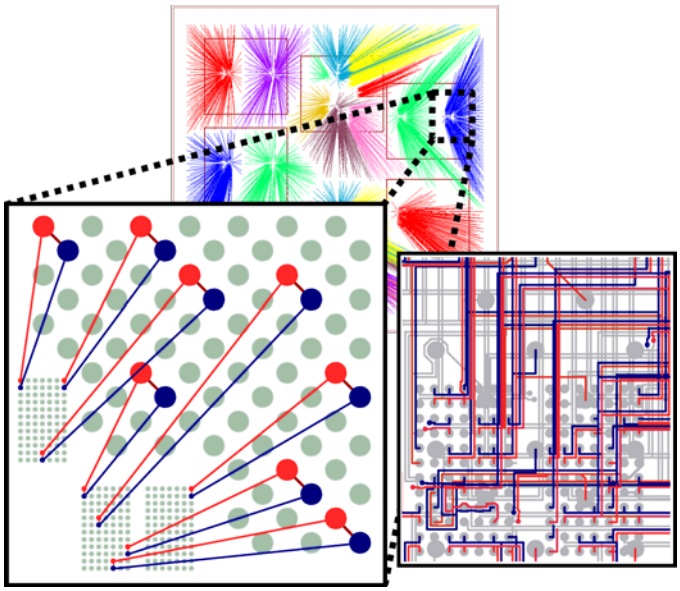


Fig. 15. The impact of the eight different fat pin variants (#1–#8) on the number of differential pairs, flyline intersections and overall flyline lengths (results of design MCM, see also Table II).



(a) (b)

Fig. 16. Details of a differential pair pin assignment. As illustrated by the shown subset of fly-lines in (a), each differential pair is assigned adjoining chip and MCM pins (smaller and larger dots) with distances of less than d_{max}. The final routing result of differential pairs is shown in (b). Note that (a) contains only a small subset of differential pairs, non-differential pairs are omitted for simplicity.

slightly decreases. (Without invalid DPPs, many pins are not transformed to fat pins and receive their basic pin assignment, while only paired pins are treated via fat pin assignment.)

For aggressive blending, invalid DPPs (#5 and #6) decrease the number of created differential pairs, while improving the quality with respect to the basic objective. This is because each pair of nets that is assigned at least one invalid DPP cannot be used for a differential pair. However, more pins are considered during fat pin assignment, hence, the overall pin assignment quality is better.

The number of created differential pairs by each variant is not predicable. Therefore, we sequentially apply variant #0 (pin assignment with the best quality and least possible differential pairs) followed by variants #3 through #8 (pin assignment with the least quality and the most possible differential pairs) until the pin assignment with enough differential pairs and the best quality achievable for the specific design is found. This methodology has been proven effective in numerous industrial examples.

Limitations and Outlook

Conventional pin assignment algorithms that minimize the overall lengths of flylines, the overall Manhattan lengths and the standard deviation of those lengths can easily be combined with the fat pin methodology without solution degradation. Pin assignment algorithms with the objective of minimum signal intersections have a limited compatibility to the fat pin methodology. This is due to the difference in coordinates of the fat pin and its two original pins that can lead to intersections near the end of the routing path. However, these additional intersections are in places where they are easily resolved by the final router and thus, do not affect routability.

The presented algorithms to select pin pairs are based on the distances of pins. Pin pairs have been specified manually if specific DPP patterns are needed (e.g., for specialized differential pair connectors). In the future, pairing algorithms must include more complex constraints than only one spacing rule. Due to the modularity of our fat pin methodology, the presented pairing algorithms can easily be replaced with any other extended method for pairing.

One example for future design challenges is the signaling method presented in [13]. It requires four nets and their pins to be handled in one group. Our methodology can easily be modified for pin assignments suitable for this signaling method by selecting groups of four pins that are to be represented by one fat pin.

Conclusions

In this chapter, a universal differential pair methodology that is applicable to all algorithms or manual processes that solve the pin assignment problem has been presented. This is the first algorithmic approach that includes differential pair constraints during pin assignment. It has been shown that it has only a minor effect on the quality of the underlying basic pin assignment algorithm (PAA). This has been verified not only during pin assignment but also by considering the actual routing results.

The fundamental principle of the presented solution is that two nets, which *can* be used for a differential pair (because their parallel pin pairs meet the spacing rules), do *not need* to be used for a differential pair. Instead, they can also be used for any two single ended nets. Based on this observation, the fat pin transformation approximately halves the number of pins that have to be considered. Thereby, the complexity of the pin assignment problem is significantly reduced, while still allowing for near optimal solutions. A pin assignment of differential pairs can be retrieved from this reduced number of pins by PAAs that originally do not respect differential pairs.

The methodology consists of different algorithms for selecting differential pin pairs (DPPs) and integrating the fat pin assignment. They can be used in different combinations to produce similar pin assignments with different numbers of possible differential pairs. The number of created possible differential pairs by the variants cannot be predicted exactly. Yet, specific variants create more differential pairs than others while in general the quality of the pin assignment decreases with an increasing number of differential pairs. Therefore, in order to find the best pin assignment with differential pairs for a specific design, the variants are executed sequentially starting with

the one producing the least differential pairs, until the first pin assignment with sufficient differential pairs is found.

Based on this add-on methodology, any present or future algorithms for the pin assignment problem can easily be extended to include differential pairs. The presented differential pair methodology is in use in the industrial design flow at IBM. Here it has shown its robustness and quality combined with a minimum of interference with the design flow that had already been established.

References

1 K. Nakagawa, M.Watanabe, et al., Giga-hertz electrical characteristics of flip-chip BGA package exceeding 2,000 pin counts. 54th Conference Proceedings, Electronic Components and Technology, 2004, 1:334–341.

2 W. Yuan, K. H. Pang, et al., Electrical analysis and design of differential pairs used in high-speed flip-chip BGA packages. 17th International Zurich Symposium on Electromagnetic Compatibility 2006, 578–581.

3 J. Westra, P. Groeneveld, Post-placement pin optimization. IEEE Computer Society Annual Symposium on VLSI. 2005, 238–243.

4 J. Westra, P. Groeneveld, Towards integration of quadratic placement and pin assignment. Proceedings. IEEE Computer Society Annual Symposium on VLSI 2005, 284–286.

5 H. Xiang, X. Tang, D.F. Wong, An algorithm for integrated pin assignment and buffer planning. Proceedings. 39th Design Automation Conference 2002, 584–589.

6 H. Xiang, X. Tang, D.F. Wong, Min-cost flow-based algorithm for simultaneous pin assignment and routing. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2003, 22(7):870–878.

7 T. Meister, J. Lienig, G. Thomke, Novel Pin Assignment Algorithms for Components with Very High Pin Counts. In Proceedings Design, Automation and Test in Europe, 2008, DATE '08, 837–842.

8 S. Chen, W. Tseng, J. Yan, S. Chen, Printed circuit board routing and package layout codesign. In APCCAS 2002 1:155–158.

9 Y. Kubo, A. Takahashi, A global routing method for 2-layer ball grid array packages. Proceedings of the 2005 ISPD, 36–43.

10 M. Yu and W.W.-M. Dai, Pin assignment and routing on a single-layer pin grid array. Proceedings of the ASP-DAC '95/CHDL '95/VLSI '95, 1995, 203–208.

11 Z. Galil. Efficient algorithms for finding maximum matching in graphs. ACM Comput. Surv., 1986, 18(1):23–38.

12 H. N. Gabow, An efficient implementation of Edmonds algorithm for maximum matching on graphs. Journal of the ACM, 1976, 23(2):221–234.

13 S. Choi, H. Lee, H. Park, A three-data differential signaling over four conductors with pre-emphasis and equalization: a CMOS current mode implementation Solid-State Circuits, IEEE Journal of Solid-State Circuits, 2006, 41:633–641.

14 N. A. Sherwani, Algorithms for VLSI Physcial Design Automation, Kluwer Academic Publishers, 1998.

15 J. Westra, P. Groeneveld, Towards integration of quadratic placement and pin assignment, VLSI, 2005. Proceedings. IEEE Computer Society Annual Symposium on, 2005, 284-286.

16 H. Xiang, X. Tang, M. Wong, Min-cost flow-based algorithm for simultaneous pin assignment and routing Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 2003, 22, 870-878.

17 J. Lienig, Layoutsynthese elektronischer Schaltungen (Algorithms in Physical Design), Springer Verlag Heidelberg, 2006.