Priority Driven Channel Pin Assignment

Ines Peters
Institut für Informatik
Lehrstuhl Prof. Dr. P. Molitor
Universität Halle
D–06099 Halle (Saale), Germany

Abstract

We present a polynomial time improvement of the linear channel pin assignment LCPA algorithms presented by Cai and Wong in 1990. We solve the LCPA problem according to minimum channel density under a special priority schedule subject to vertical constraints and flux. The priority driven linear channel pin assignment algorithm (PDCPA) reduces the channel height by an average of 17% without increasing the running time\(^1\).

1 Introduction

The channel routing problem plays an important role in the physical design of VLSI circuits. It has been extensively studied in the past ([15], [5], [19], [7], [10], [12]). Advanced (over-the-cell) channel routers consider the terminals (pins) on the two sides of the channel not completely fixed at the beginning of the routing phase ([17], [2], [6], [11], [13], [18]). This approach enables further channel density reduction. The channel pin assignment (CPA) problem subject to position, order and separation constraints has been shown to be NP-hard in general ([2]). Polynomial time algorithms for an important special case in which the relative order of the terminals on the top and on the bottom borders of the channel are completely fixed (linear channel pin assignment (LCPA) problem), have been presented in [3], [4] and [2].

We present a fast and easy schedule for the optimal linear CPA which reduces the channel height by an average of 17% as compared to the existing LCPA algorithm presented in [2]. The main idea of the presented algorithm is a priority driven computation of the minimum channel density. The priorities depend on the local densities ([16]), the properties of the vertical constraint graph ([5], [19]) and the flux\(^2\) ([11]). For vertical constraints see [5], [19], [9] and for another lower bound on the channel height called Flux see [1], [9].

\(^1\)This work was supported by the BMFT under 01 IS 102.

\(^2\)The flux \(f(C)\) of a channel \(C\) is defined as \(f(C) = \max\{f_{\text{local}}(c) \mid c \text{ is a flux cut of } C\}\), where a flux cut is a contiguous interval (of columns) on one of the sides of the channel and the local flux \(f_{\text{local}}(c)\) of a flux cut \(c\) which spans \(n\) nontrivial columns and splits exactly \(n\) nontrivial nets is defined as the smallest integer \(t\) such that the inequality \((l-n)t + (t+1) \geq n\) holds.

The Figures 1 and 2 show a comparison of the LCPA algorithm presented in [2] (Figure 1) with our new approach (PDCPA) (Figure 2). The channel length is fixed (to 20 in the example) like in most practical applications. So the channel area depends only on the channel height. The minimum channel density equals five in both cases. Both LCPA and PDCPA compute a minimum channel density solution, but the channel heights differ due to the different vertical constraints graphs (shown in the lower part of the Figures).

Figure 1: Rotate1Inverse with LCPA

2 Preliminaries

Given a channel with length \(L\). Let \(\text{TOP} = \{t_1, t_2, \ldots, t_p\}\) and \(\text{BOTTOM} = \{b_1, b_2, \ldots, b_q\}\) be the set of terminals on the top and bottom border of the channel, respectively.

The set of nets \(N = \{N_1, N_2, \ldots, N_n\}\) connecting these terminals is a partition of \(\text{TOP} \cup \text{BOTTOM}\), such that \(N_i\) contains the set of terminals of the \(i\)th
net, \(1 \leq i \leq n\). Let \(\text{net}: \text{TOP} \cup \text{BOTTOM} \rightarrow N\) assign a given terminal to the corresponding net.

The order constraints of the Linear CPA problem are given by a fixed order of the terminals. We assume that \(t_1 < t_2 < \ldots < t_p\) and \(b_1 < b_2 < \ldots < b_q\), in the sense that \(t_i\) has to be assigned to the left of \(t_{i+1}\).

There are also position constraints, i.e., valid terminal positions, given by \(T = \{T_k \subseteq \{1, 2, \ldots, L\}; 1 \leq k \leq p\}\) and \(B = \{B_k \subseteq \{1, 2, \ldots, L\}; 1 \leq k \leq q\}\).

In order to assign all terminals \(t_i, b_j\) to a position, i.e., a certain column \(k\), we have to distinguish between four types of decisions. They are shown in Figure 3. Let \(R_1(i, j)\) (\(R_2(i, j)\)) be the set of nets with one terminal in \(\{t_1, t_2, \ldots, t_{i-1}, b_1, b_2, \ldots, b_j\}\) \(\{t_1, t_2, \ldots, t_i, b_1, b_2, \ldots, b_j\}_{-1}\) and one terminal in \(\text{TOP} \cup \text{BOTTOM}\) \(\{t_1, t_2, \ldots, t_i, b_1, b_2, \ldots, b_j\}\).

The net \(\text{net}(t_i)\) \(\text{net}(b_j)\) also belongs to \(R_1(i, j)\) \(R_2(i, j)\) if it is not trivial. In the set \(R_3(i, j)\) all nets are not trivial in \(\{t_1, t_2, \ldots, t_{i-1}, b_1, b_2, \ldots, b_{j-1}\}\) and one terminal in \(\text{TOP} \cup \text{BOTTOM}\) \(\{t_1, t_2, \ldots, t_i, b_1, b_2, \ldots, b_j\}\) and the nets \(\text{net}(t_i), \text{net}(b_j)\) if they are not trivial and not \(\text{net}(t_i) = \text{net}(b_j) = \{t_i, b_j\}\).

![Figure 3: Four types of \((i, j, k)\)-decisions](image)

We define the crossing numbers\(^3\) at \((i, j, k)\) as:

\[
x(i, j, k) = \begin{cases} +\infty & \text{if } k \notin T_i \\ |R_1(i, j)| & \text{otherwise;}
\end{cases}
\]

\[
y(i, j, k) = \begin{cases} +\infty & \text{if } k \notin B_j \\ |R_2(i, j)| & \text{otherwise;}
\end{cases}
\]

\[
z(i, j, k) = \begin{cases} +\infty & \text{if } k \notin T_i \cap B_j \\ |R_3(i, j)| & \text{otherwise;}
\end{cases}
\]

Let \(d = (d_1, d_2): \{1, \ldots, p\} \times \{1, \ldots, q\} \times \{1, \ldots, L\} \rightarrow \mathbb{N} \times \{0, 1, 2, 3\}\) be a function which assigns the local density and the \((i, j, k)\)-decision type to the terminals \(t_i\) and \(b_j\) and the column \(k\).\(^4\)

### 3 The algorithm

For every column of the channel the algorithm has to decide for one of the type 0, type 1, type 2 and type 3 solutions. The decision is made according to the minimum channel density. Here we show the main part of the original CPA algorithm (see also [2]):

1. For \(k = 1\) to \(L\) do
2. For \(i = 0\) to \(p\) do
3. For \(j = 0\) to \(q\) do
4. \(D_0 := d_1(i, j, k - 1)\)
5. \(D_1 := \max\{d_1(i - 1, j, k - 1), z(i, j, k)\}\)
6. \(D_2 := \max\{d_1(i, j - 1, k - 1), y(i, j, k)\}\)
7. \(D_3 := \max\{d_1(i - 1, j - 1, k - 1), y(i, j, k)\}\)
8. \(d(i, j, k) := \text{Min}(D_0, D_1, D_2, D_3)\)

In [2] the function Min returns the minimum density and the solution type. But there may be one solution type satisfying this supposition. At this point we introduce a priority schedule to enable the algorithm to prefer one of these. We introduce priorities \(P_0, \ldots, P_3\) preferring decisions of type 0, type 1, type 2, and type 3, respectively. The priorities are increased or decreased according to the current situation before the new minimum function PriorityMin is called.

- In order to decrease channel congestion we perform the following steps according to the local densities:
  9. If \(D_0 = D_1 = D_2 = D_3\) then
  10. if \(x(i, j, k) < y(i, j, k), z(i, j, k)\)
  11. then Increase \(P_1\)
  12. if \(y(i, j, k) < x(i, j, k), z(i, j, k)\)
  13. then Increase \(P_2\)
  14. if \(z(i, j, k) < x(i, j, k), y(i, j, k)\)
  15. then Increase \(P_3\).

- Type 3 decisions affect vertical constraints. We prefer type 3 decisions having the same net on the top and on the bottom of the current column. Columns with different nets on the top and on the bottom cause edges in the vertical constraint graph:
  16. if \(\text{net}(t_i) = \text{net}(b_j)\) then Increase \(P_3\)

\(\text{Note that } d_1(p, q, L)\text{ is the channel density and } d_2(p, q, L)\text{ is the decision type for the last column: } d_2(p, q, L) = 3\text{ means that both } t_p \text{ and } b_q \text{ have to be placed to position } L. d_2(p, q, L) = 2\text{ (= 1)}\text{ means that just } b_q \text{ (} t_p \text{) has to be placed to position } L.\text{ And } d_2(p, q, L) = 0\text{ means that neither } t_p \text{ nor } b_q \text{ are assigned to } L.\)
• Free or partly free columns are important to reduce flux. This relationship has been explained in [1]. In Figure 4 we show type 3 decisions in different contexts. We have to decide whether the terminals 3 and 4 should be put together in one column (say k). Therefore in the left most case the priority is set very high because of the free column k – 1. In the right most case the priority is very low (k – 1 is not free). In the other cases the priority is set to a medium value. Decisions of other types are handled in a similar way.

2 4 2 4 2 4 2 4
1 3 1 3 1 3 1 3

Figure 4: Type 3 decisions in different contexts

In the new algorithm PDCPA we replaced the function Min by a function PriorityMin, which returns the minimum density and the decision type according to the priorities set before. It is implemented as follows: PriorityMin:

(18) Val := \text{min}\{D_0, D_1, D_2, D_3\}
(19) MinSet := \{a \mid a \in \{0, 1, 2, 3\}, D_a = \text{Val}\}
(20) if \text{MinSet} = \{\text{a}\} then \text{return} (\text{Val, a})
(21) else \text{MinSet} := \text{MinSet} \setminus \{\text{0}\}
(22) if \text{MinSet} = \{\text{a}\} then \text{return} (\text{Val, a})
(23) else if \text{MinSet} = \{1, 2\} then
(24) if \text{P}_1 > \text{P}_2 then \text{return} (\text{Val, 1})
(25) else if \text{P}_2 > \text{P}_1 then \text{return} (\text{Val, 2})
(26) else return (\text{Val, 1}), (\text{Val, 2}) alternating
(27) else \text{MinSet} = \{1, 3\}
(28) if \text{P}_1 > \text{P}_3 then \text{return} (\text{Val, 1})
(29) else return (\text{Val, 3})
(30) else if \text{MinSet} = \{2, 3\}
(31) if \text{P}_2 > \text{P}_3 then \text{return} (\text{Val, 2})
(32) else return (\text{Val, 3})
(33) else if \text{MinSet} = \{1, 2, 3\} then
(34) if \text{P}_3 = \max\{\text{P}_1, \text{P}_2, \text{P}_3\} then
(35) then return (\text{Val, 3})
(36) else
(37) if \text{P}_1 > \text{P}_2 then \text{return} (\text{Val, 1})
(38) else if \text{P}_2 > \text{P}_1 then \text{return} (\text{Val, 2})
(39) else return (\text{Val, 1}), (\text{Val, 2}) alternating

Of course the establishing of the function d = (d_1, d_2) is followed by the assignment of the terminals to the positions using backtracking. Here we use the analogous algorithm as in [2], p.13.

Obviously the proposed priority driven algorithm PDCPA runs the same time as compared to the original algorithm.

4 Experimental results

We implemented the proposed algorithm PDCPA in Modula-2 language on PC. All data are ISCAS benchmarks. For them we generated layout data and cell placements with HULDA ([9], [8]). To speed up the algorithm we partitioned the channels into several subchannels as proposed in [3]. In the last four examples we generated two-row cell placements (to get longer channels) and applied the heuristics.

The experimental results are summarized in Table 1. The L means the channel length, D the channel density, H1 the channel height achieved by LPCA and H2 the channel height achieved by PDCPA. The ratio between both heights is given in %.

Table 1: Experimental results

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Ch</th>
<th>L</th>
<th>D</th>
<th>H1</th>
<th>H2</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC</td>
<td>ch1</td>
<td>75</td>
<td>8</td>
<td>14</td>
<td>8</td>
<td>57.1</td>
</tr>
<tr>
<td>ch2</td>
<td>75</td>
<td>13</td>
<td>15</td>
<td>14</td>
<td>93.3</td>
<td></td>
</tr>
<tr>
<td>CM138</td>
<td>ch1</td>
<td>38</td>
<td>10</td>
<td>14</td>
<td>10</td>
<td>71.4</td>
</tr>
<tr>
<td>CM150</td>
<td>ch1</td>
<td>83</td>
<td>10</td>
<td>12</td>
<td>10</td>
<td>83.3</td>
</tr>
<tr>
<td>ch2</td>
<td>83</td>
<td>7</td>
<td>9</td>
<td>7</td>
<td>77.8</td>
<td></td>
</tr>
<tr>
<td>CM151</td>
<td>ch1</td>
<td>83</td>
<td>11</td>
<td>13</td>
<td>12</td>
<td>92.3</td>
</tr>
<tr>
<td>CM162</td>
<td>ch1</td>
<td>65</td>
<td>15</td>
<td>16</td>
<td>15</td>
<td>93.7</td>
</tr>
<tr>
<td>CM163</td>
<td>ch1</td>
<td>65</td>
<td>13</td>
<td>14</td>
<td>13</td>
<td>82.8</td>
</tr>
<tr>
<td>CM42</td>
<td>ch1</td>
<td>65</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>70.6</td>
</tr>
<tr>
<td>CM82</td>
<td>ch1</td>
<td>65</td>
<td>8</td>
<td>9</td>
<td>9</td>
<td>100.0</td>
</tr>
<tr>
<td>CM85</td>
<td>ch1</td>
<td>73</td>
<td>10</td>
<td>14</td>
<td>14</td>
<td>100.0</td>
</tr>
<tr>
<td>CMB</td>
<td>ch1</td>
<td>82</td>
<td>17</td>
<td>20</td>
<td>18</td>
<td>90.0</td>
</tr>
<tr>
<td>CORDIC</td>
<td>ch1</td>
<td>100</td>
<td>12</td>
<td>16</td>
<td>12</td>
<td>75.0</td>
</tr>
<tr>
<td>ch2</td>
<td>100</td>
<td>9</td>
<td>11</td>
<td>9</td>
<td>81.3</td>
<td></td>
</tr>
<tr>
<td>CU</td>
<td>ch1</td>
<td>100</td>
<td>11</td>
<td>14</td>
<td>12</td>
<td>85.7</td>
</tr>
<tr>
<td>ch2</td>
<td>100</td>
<td>10</td>
<td>13</td>
<td>10</td>
<td>76.9</td>
<td></td>
</tr>
<tr>
<td>DECOD</td>
<td>ch1</td>
<td>100</td>
<td>11</td>
<td>23</td>
<td>13</td>
<td>56.5</td>
</tr>
<tr>
<td>I1</td>
<td>ch1</td>
<td>100</td>
<td>14</td>
<td>16</td>
<td>14</td>
<td>87.5</td>
</tr>
<tr>
<td>MUX</td>
<td>ch1</td>
<td>107</td>
<td>12</td>
<td>13</td>
<td>13</td>
<td>100.0</td>
</tr>
<tr>
<td>ch2</td>
<td>107</td>
<td>11</td>
<td>12</td>
<td>11</td>
<td>91.6</td>
<td></td>
</tr>
<tr>
<td>PARITY</td>
<td>ch1</td>
<td>67</td>
<td>11</td>
<td>13</td>
<td>11</td>
<td>84.6</td>
</tr>
<tr>
<td>PCLER8</td>
<td>ch1</td>
<td>102</td>
<td>15</td>
<td>11</td>
<td>73.3</td>
<td></td>
</tr>
<tr>
<td>ch2</td>
<td>102</td>
<td>14</td>
<td>16</td>
<td>14</td>
<td>87.5</td>
<td></td>
</tr>
<tr>
<td>PM1</td>
<td>ch1</td>
<td>81</td>
<td>18</td>
<td>31</td>
<td>20</td>
<td>64.5</td>
</tr>
<tr>
<td>X2</td>
<td>ch1</td>
<td>90</td>
<td>17</td>
<td>20</td>
<td>19</td>
<td>95.0</td>
</tr>
<tr>
<td>Z4ML</td>
<td>ch1</td>
<td>90</td>
<td>11</td>
<td>15</td>
<td>11</td>
<td>73.3</td>
</tr>
<tr>
<td>ch2</td>
<td>90</td>
<td>12</td>
<td>14</td>
<td>12</td>
<td>85.7</td>
<td></td>
</tr>
<tr>
<td>CC</td>
<td>ch1</td>
<td>110</td>
<td>19</td>
<td>25</td>
<td>22</td>
<td>88.0</td>
</tr>
<tr>
<td>CU</td>
<td>ch1</td>
<td>102</td>
<td>18</td>
<td>23</td>
<td>22</td>
<td>95.7</td>
</tr>
<tr>
<td>CM150</td>
<td>ch1</td>
<td>122</td>
<td>25</td>
<td>26</td>
<td>25</td>
<td>96.2</td>
</tr>
<tr>
<td>PCLER8</td>
<td>ch1</td>
<td>144</td>
<td>18</td>
<td>29</td>
<td>26</td>
<td>89.7</td>
</tr>
</tbody>
</table>

References


