Pair Wise Swapping Based Hypergraph Partitioning Algorithms for VLSI Design

Mitali Sinha

Abstract: A VLSI integrated circuit is the most significant part of electronic systems such as personal computer or workstation, digital camera, cell phone or a portable computing device, and automobile. So development within the field of electronic space depends on the design planning of VLSI integrated circuit. Circuit partitioning is most important step in VLSI physical design process. Many heuristic partitioning algorithms are proposed for this problem. The first heuristic algorithm for hypergraph partitioning in the domain of VLSI is FM algorithm. In this paper, I have proposed three variations of FM algorithm by utilizing pair insightful swapping strategies. I have played out a relative investigation of FM and my proposed algorithms utilizing two datasets for example ISP98 and ISPD99. Test results demonstrate that my proposed calculations outflank the FM algorithm.

Index terms: Block, Graph, Hypergraph, netcut, netlist, VLSI.

I. INTRODUCTION

VLSI circuit comprises of different components put on a single chip which are associated through wires and a gathering of associated components can be spoken to as a block. Partitioning is a most essential step in the physical structure cycle in which a given VLSI circuit is partitioned into a lot of disjoint blocks of explicit sizes. Size of the wires within the components of two unique blocks is more than the size of wires of components inside a similar block; hence we need to limit the quantity of wires between the components of two distinct blocks to decrease the expense by utilizing a parceling algorithm. Input to the partitioning algorithm is a VLSI circuit and output is number of disjoint blocks with minimum number of inter-block wires. The Significant of a partitioning algorithm in any physical framework configuration is that it breaks down the entire complex framework into a lot of subsystems. Consequently every subsystem can be planned autonomously and all the while to speed of the structure procedure. Partitioning has numerous applications, for example, tremor examination, social database design, reproduction, testing and VLSI design.

A. Hypergraph partitioning problem.

Hypergraph is a theory of diagram wherein an edge relates any number of vertices and this edge is called as hyperedge. Numerically hypergraph can be spoken to as H(V, E) where V is the arrangement of vertices and E is the arrangement of hyperedges. A circuit can be changed over to a hypergraph in which a vertex of hypergraph speaks to a component of the circuit and a hyperedge speaks to the arrangement of components which share same signal known as net. A lot of nets which speak to a circuit is known as netlist. Two components of a circuit are said to be neighbor if both are present within an at least one common net.

Let’s take an example to illustrate the partitioning problem. A circuit and its netlist portrayal are appeared in fig. 1(a), fig. 1(b) respectively. This netlist contains three nets named as L_1, L_2, and L_3. L_1 contains 4, 5 components and output of 4 is given as input to 5. Similarly L_2 and L_3 can be depicted. Netlist to hypergraph H(V,E) transformation is appeared in fig.1(c) in which V={1,2,3,4,5} and E=[L_1,L_2,L_3]. The fundamental objective of the hypergraph partitioning is to put the parts of the circuit in their ideal blocks with the end goal that netcut or hyperedge cut can be limited. A net or hyperedge is said to be cut if its vertices are present in two or more blocks.

A hypergraph as appeared in fig.1(c) is given to a partitioning algorithm and gap it into two around equivalent sized blocks. The outcome of the partitioning algorithm can be 4, 5, 3 components are placed in one block and 1, 2 are placed in another block as appeared in fig.2. The netcut of this partition is one on the basis that the components of net L_3 are placed in both the blocks.

Revised Manuscript Received on July 09, 2019.
Er Mitali Sinha, Department of Computer science & Engineering, parala maharaja Engineering college Affiliated to Biju Patnaik uni-versity of Technology, Berhampur, Odisha, India
Hypergraph partitioning is a NP-hard [1] issue. Despite the fact that partitioning has broad applications in different fields, for example, information mining, work booking, picture handling, enhancing page blame and VLSI plan, various heuristic algorithms were created with polynomial time-complexity. Fiduccia and Mattheyses (FM) algorithm [4] is a first hypergraph partitioning algorithm which time complexity is polynomial in nature.

B. Literature review
So as to take care of the parceling issue in VLSI context, the first polynomial graph bi-partitioning algorithm was proposed in the year 1970 known as KL[2] algorithm. Graph is certifiably not an appropriate portrayal of a circuit [3] in light of the fact that it can't effectively change over a net to an edge or a lot of edges. The most right portrayal of a circuit in graph theorem is hypergraph. So in the year 1982 a hypergraph bi-partitioning algorithm was proposed known as FM algorithm [4]. The main advantage of this algorithm is its polynomial time-complexity which is directly proportional to the size of the circuit. But this algorithm is not worthy for a VLSI circuit. A VLSI circuit contains more than one million of transistors on a single chip. So a various variations of FM algorithm were developed [5,6,10,12,13]. C.J. Alpert and A.B. Kahng has completed an exhaustive review on netlist partitioning [14]. Another class of partitioning algorithm was developed [7,8,9,11,20]. These algorithms comprise of two stages. In the first phase clustering algorithm is applied on original hypergraph in order to reduce the size of hypergraph. Then partitioning algorithm is applied on the reduced hypergraph. Partitioning result of the reduced hypergraph is applied on original hypergraph. This partitioning result is refined using FM or variant of FM algorithm. A predominant expansion of FM algorithm is multi level FM (MLFM) algorithms [15,16,17,18,19] which gives better outcome both as far as solution quality and run time. The multi-level algorithm consists of three stages such as coarsening stages, initial partitioning of coarsest hypergraph, un-coarsening and refinement. In coarsening stage a sequence of coarser hypergraphs are produced until the size of the hypergraph is not greater than a pre-defined size. So a number of smaller hypergraphs are produced at different level of coarsening stages. Suppose H0 is an original hypergraph, at the first level of coarsening phase H1 is produced by applying clustering algorithm on H0. The size of H1 is smaller than H0. At the next level of clustering phase H2 hypergraph is produced from H1 and its size is smaller than H1. This process is continued until a pre-defined smallest size hypergraph is produced. The next stage in multi-level algorithm is initial partitioning stage, in which a random initial partitioning of coarsest hypergraph is performed which is refined by FM algorithm or by its variant. The last stage of multi-level algorithm is un-coarsening and refinement stage. In this stage, a partitioning of the coarser hypergraph is projected on its next level finer hypergraph and refined using FM or variant of FM algorithm. This process is continued until partitioning result is projected on original hypergraph and refined using FM or variant of FM algorithm.

C. My contribution
In this paper I have proposed a clever thought of pair wise swapping of components in hypergraph partitioning. At first a circuit is divided in to two roughly equal sized blocks with haphazardly assign the components of a circuit to blocks. Then the selection of components is in pair wise manner, each from one block to reduce the netcut using swapping technique. I have proposed three different types of swapping technique. My proposed algorithms are variations of FM algorithm which I call as Pair Wise swapping based Hypergraph Partitioning (PSHP). I have made a relative execution investigation of PSHPs with FM algorithm utilizing two informational indexes, for example, ISPD98 and ISPD99 benchmark circuits. My trial results demonstrate that PSHPs beat FM algorithm.

II. A PRIMITIVE HYPERGRAPH PARTITIONING ALGORITHM - FM ALGORITHM

The first fundamental hypergraph partitioning algorithm is FM algorithm [4]. It begins with an irregular beginning allotment of the hypergraph into two balanced blocks. Toward the start of the procedure every one of the vertices of hypergraph are made allowed to move from its own block to its complementary block and gain value of every vertex is determined. The gain value of a vertex depends on number of decrease in net from netcut when it is moved from its present block to its complimentary block. Vertices of two blocks are arranged using bucket sorting method according to their gain value. A component c with highest gain value of a bigger sized block is chosen to shift to its complementary block and remains locked all through the procedure. After c is shifted, the gain values of its neighbors are refreshed for next move and the netcut is recorded for that move. This is proceeded until all cells are locked. This whole procedure is known as a pass. Toward the finish of a pass, the point where the ideal netcut was accomplished is chosen and the moves of all vertices after that point are dropped. The consequence of one pass of FM algorithm is given to next pass as input. This process is continued still no chance of improvement in netcut.

A. Pseudo-code of FM algorithm in a pass

1) An initial random partition is applied on the circuit in order to divide it into two balanced blocks. The size of a block is defined as number of components within that block.

2) The gain of each cell c is calculated on the basis of reduction of nets from netcut due to its move from its current block F to its complementary block T. The formula for gain value of each component c is FS(c) – TE(c). FS(c) is number of nets which contain only c in F block. TE(c) is number of nets which contain c in F block but no component in T block.

3) Components of two blocks are sorted using bucket sorting method.

4) The component c with highest gain value of a block with greater size is selected for shifting and it becomes locked in this pass. Gain values of c' neighboring cells are updated and reduction of netcut due to shifting is recorded.

5) step-4 is repeated until all the components are not locked.
6) After completion of a pass, the point at which reduction in netcut is optimal is selected and shifting of cells after that point is cancelled.

B. Limitation of FM algorithm

- In FM algorithm, selection of component for shifting is dependent on the balancing constraint of block’s size.
- Swapping of components provides better result than single shifting of a component [10, 21].

These two limitations are overcome in my proposed algorithms.

III. PROPOSED ALGORITHMS

A partitioning algorithm that swaps node pair gives a superior netcut enhancement than one that shifts single node at once [10, 21]. So I have applied pair wise swapping of nodes on hypergraph partitioning with three different techniques. Before going to discuss different pair wise technique I have presented here the gain or reduction in netcut due to pair wise swapping. Let’s take u_i and v_j to swap and both are present in two different blocks. The gain due to swapping of u_i and v_j is denoted as Gain(PS) and calculated using the formula i.e Gain(u_i) + Gain(v_j) = correct-term. Gain(u_i) is already discussed in section 2. The pseudo-code for correct-term is shown in fig.3. A net which is cut is said to be critical if any one of its cell is shifted from its block to another, then that net is removed from netcut. As shown in fig.4 this net is cut and it is a critical net because its cells are present in both the blocks and if cell 5 is shifted, then this net is removed from netcut.

Fig3. Pseudo-code for correct term

```
Correct-term
begin
Correct-term = 0;
for each critical net N_i(N_{ij} \in critical-net)
if(cardinality of critical-net is 2)
Correct-term = Correct-term + 2;
else
Correct-term = Correct-term + 1;
end

```

Fig4. Critical net

A. Pair Wise swapping Based Hypergraph partitioning algorithms

I am now ready to represent the complete description of pair wise swapping algorithm with in a pass:
1. If a hypergraph contains odd number of vertices, then add a dome node. Dome node will have no connection with other vertices.
2. Now a random partition is applied on the hypergraph to divide the vertices into two equal sized blocks let’s say A and B blocks. Initially gain due to pair wise swapping is zero, let’s denote it as Gain(PS)=0 and netcut is equal to number of nets which components are present in both the block.
3. The gain of each vertex is calculated and it is based on reduction in netcut when it is shifted from its current block to other as described in FM algorithm.
4. Vertices of two blocks are arranged using bucket sorting method according to their gain value. Let’s say \{u_1, u_2, u_3, ..., u_m\} vertices are sorted in block A and \{v_1, v_2, ..., v_n\} vertices are sorted in block B.
5. One vertex from each block is selected using a pair wise swapping technique. Let’s say u_i from block-A and v_j from block-B are selected for swapping. Now Gain(PS)=Gain(PS) + Gain due to pair wise swapping and netcut-netcut - Gain due to pair wise swapping.
6. Now u_i and v_j are locked. Gain values of all unlocked neighboring vertices of u_i and v_j are updated and replaced in their respective buckets.
7. Step-5 and 6 are repeated until all the vertices are locked.
8. At the end of the pass the point at which Gain(PS) is maximum selected and all swaps after that point are cancelled.

I have discussed three types of pair wise swapping techniques below.

- In the first technique after vertices are sorted in both the blocks using bucket sorting which is mentioned above in step-4, a vertex with highest gain value i.e. u_i is selected from block-A and its non-neighboring vertex with highest gain value is selected from block-B i.e. v_j. The time complexity is required for finding the non-neighboring vertex of u_i is O(k). This step is repeated n times to get n pairs if a hypergraph contains 2n vertices. So total time complexity of this step is O(n*k). Here k <= (maximum number of nets belong to a cell) (maximum size of a hyperedge).

For the future reference this technique is named as Pairwise Swapping based Hypergraph Partitioning(PHSP).

- In the second technique, a vertex with highest gain value from each block is selected for swapping i.e. u_i from block-A and v_j from block-B are selected for swapping. The time complexity is required for selecting u_i and v_j is O(1). So for selections of n pairs, time complexity is O(n). For the future reference this technique is named as Pairwise Swapping based Hypergraph Partitioning 2(PHSP_2).

- In the third technique, u_i and v_j from each block are selected so that u_i and v_j provides highest gain value than any other pair. The procedure for this technique is described below:
1) \{u_1, u_2, u_3, ..., u_m\} vertices are sorted in block-A and \{v_1, v_2, ..., v_n\} vertices are sorted in block B using gain value.
2) Suppose v_j is the non-neighboring vertex of u_i with highest gain value. Now the best cell in the block-B is considered for swapping can be any one from \{v_1, v_2, ..., v_n\}
3) Suppose u_i is the non-neighboring vertex of v_j with highest gain value. Now the best cell in the block-A is considered for swapping can be any one from \{u_1, u_2, u_3, ..., u_m\}. The cells which are nominated as best can be arranged in matrix form. So the time complexity required to select the best pair from this matrix is O(k^2).
4) This procedure is repeated n times to get n best pairs so total time complexity is O(n*k^2) and k<5<<n.
Pair Wise Swapping Based Hypergraph Partitioning Algorithms for VLSI Design

For the future reference this technique is named as Pair-wise Swapping based Hypergraph Partitioning (PSHP).

IV. EXPERIMENTAL STUDY

FM and my proposed swapping techniques are tested using two extensive datasets such as ISPD98 and ISPD99 benchmark circuits.

A. Experimental Setup

The source code of the algorithms are applied in ‘C++’ language and home windows environment. I even have used a 32-bit compiler (Dev C++ Version 4.9.2), 2 GB RAM and a processor which pace is two GHz. Input to the application is a document (dataset) and the output of the program is minimum netcut due to partition.

B. The Dataset

The ISPD98 benchmark circuit is the biggest dataset that’s kept up through the Collaborative Benchmarking Laboratory. The ISPD98 benchmark circuit consists of eighteen different styles of records which includes IBM01 to IBM18. Each document comes with three exceptional codecs together with ,Internet, .Are and .NetD. Another model of ISPD benchmark circuit is ISPD99. This benchmark circuit consists of 9 extraordinary sorts of file. These datasets are freely to be had inside the website online http://vlsicad.U.S.Edu/UCLAWeb.Html.

C. Performance evaluation on ISPD benchmark circuits

I have performed the experiments using two different datasets. For all the experiments I have defined Gain(μ) as follow

\[
\text{Gain}(\mu) = \frac{\text{Netcut}_{\text{FM}} - \text{Netcut}_{\text{PSHP}}}{\text{Netcut}_{\text{FM}}} * 100
\]

Performance of PSHP algorithms can be considered to be better if the Gain(μ) is positive. In the first experiment I have computed the netcut of FM and PSHP algorithms by considering ISPD98 as input datasets. In the second experiments I have computed the netcut of FM and PSHP algorithms by taking ISPD99 as input datasets. Gain(μ), Gain(μ2), Gain(μ3) are used for the comparison of PSHP1 and FM, PSHP2 and FM, PSHP3 and FM respectively.

**Experiment-1: ISPD98 as input dataset**

In this experiment I actually have considered eighteen exclusive documents of ISPD98 benchmark circuit. I have computed the Netcut of FM and PSHPs algorithms in addition to the gain values as proven in table 1.

**Experiment-2: ISPD99 as input dataset**

In this experiment I even have taken 9 one-of-a-kind files of ISPD99 benchmark circuit. I have computed the Netcut of FM and PSHPs algorithms in addition to the gain values as proven in table 2.

V. CONCLUSION

In this work I have proposed 3 styles of Pair-sensible Swapping Hypergraph Partitioning algorithms (PSHPs). I even have carried out an experimental look at to evaluate performance of my proposed PSHPs algorithms and FM algorithm through considering two enter datasets together with ISPD98 and ISPD99 benchmark circuits. From experimental end result it is concluded that my proposed algorithms PSHP1 and PSHP3 outperforms FM algorithm for netlists generated from above datasets. PSHP2 provides better effects in netcut for all files as examine to FM except IBM01C. Thus PSHPs give higher result than FM. If PSHP is applied in partitioning phase of multi-level algorithm, it’s going to give higher result than hmetis.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Netcut</th>
<th>Netcut</th>
<th>Gain(μ)</th>
<th>Gain(μ2)</th>
<th>Gain(μ3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISPD98</td>
<td>532</td>
<td>529</td>
<td>5.9</td>
<td>66.8</td>
<td>66.8</td>
</tr>
</tbody>
</table>

Table 1: Netcut incurred by FM and PSHPs for ISPD98

REFERENCES


DOI: 10.35940/ijrte.B3612078219/190BEIESP

Published By: Blue Eyes Intelligence Engineering & Sciences Publication

5592
Table II: Netcut incurred by FM and PSHP for ISPD99

<table>
<thead>
<tr>
<th>File Name</th>
<th>Initial Netcut</th>
<th>Net cut</th>
<th>Net cut PSHP</th>
<th>Gain (µ)</th>
<th>Net cut</th>
<th>Net cut PSHP</th>
<th>Gain (µ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM01A</td>
<td>5212</td>
<td>2148</td>
<td>599</td>
<td>72.11</td>
<td>599</td>
<td>72.11</td>
<td>364.81</td>
</tr>
<tr>
<td>IBM01B</td>
<td>4998</td>
<td>685</td>
<td>124</td>
<td>82</td>
<td>95</td>
<td>86.13</td>
<td>124</td>
</tr>
<tr>
<td>IBM01C</td>
<td>4870</td>
<td>1000</td>
<td>433</td>
<td>46.6</td>
<td>832.7</td>
<td>4.5</td>
<td>495.81</td>
</tr>
<tr>
<td>IBM01D</td>
<td>4985</td>
<td>2288</td>
<td>295</td>
<td>76.74</td>
<td>254</td>
<td>80</td>
<td>295</td>
</tr>
<tr>
<td>IBM02A</td>
<td>22088</td>
<td>5147</td>
<td>1315</td>
<td>74.45</td>
<td>1695</td>
<td>67.1</td>
<td>1338</td>
</tr>
<tr>
<td>IBM02B</td>
<td>1982</td>
<td>2565</td>
<td>948</td>
<td>63.41</td>
<td>554</td>
<td>76.94</td>
<td>948</td>
</tr>
<tr>
<td>IBM02C</td>
<td>14358</td>
<td>3044</td>
<td>977</td>
<td>69.9</td>
<td>958</td>
<td>69.2</td>
<td>963</td>
</tr>
<tr>
<td>IBM02D</td>
<td>8693</td>
<td>3360</td>
<td>871</td>
<td>67.09</td>
<td>870</td>
<td>61.1</td>
<td>859</td>
</tr>
<tr>
<td>IBM03A</td>
<td>40187</td>
<td>9966</td>
<td>2894</td>
<td>70.96</td>
<td>3347</td>
<td>68.42</td>
<td>3026</td>
</tr>
<tr>
<td>IBM03B</td>
<td>23104</td>
<td>10064</td>
<td>4040</td>
<td>50</td>
<td>3666</td>
<td>54.53</td>
<td>3820</td>
</tr>
<tr>
<td>IBM03C</td>
<td>36006</td>
<td>8022</td>
<td>4457</td>
<td>44.44</td>
<td>2311</td>
<td>71.2</td>
<td>4706</td>
</tr>
<tr>
<td>IBM03D</td>
<td>31809</td>
<td>8343</td>
<td>3890</td>
<td>53.37</td>
<td>5656</td>
<td>32.21</td>
<td>2674</td>
</tr>
<tr>
<td>IBM04A</td>
<td>58751</td>
<td>11898</td>
<td>2661</td>
<td>77.8</td>
<td>3329</td>
<td>72.73</td>
<td>2865</td>
</tr>
<tr>
<td>IBM04B</td>
<td>20021</td>
<td>4259</td>
<td>1299</td>
<td>69.65</td>
<td>1250</td>
<td>70.65</td>
<td>2531</td>
</tr>
<tr>
<td>IBM04C</td>
<td>32816</td>
<td>7819</td>
<td>2222</td>
<td>76.3</td>
<td>1159</td>
<td>83.18</td>
<td>2322</td>
</tr>
<tr>
<td>IBM04D</td>
<td>21601</td>
<td>4938</td>
<td>1394</td>
<td>71.77</td>
<td>3022</td>
<td>38.6</td>
<td>1394</td>
</tr>
<tr>
<td>IBM01A</td>
<td>53407</td>
<td>12833</td>
<td>4885</td>
<td>61.93</td>
<td>3711</td>
<td>71.73</td>
<td>4408</td>
</tr>
<tr>
<td>IBM01B</td>
<td>27539</td>
<td>5701</td>
<td>2404</td>
<td>57.83</td>
<td>3220</td>
<td>43.52</td>
<td>3606</td>
</tr>
<tr>
<td>IBM01C</td>
<td>29364</td>
<td>7013</td>
<td>2916</td>
<td>61</td>
<td>1651</td>
<td>78.27</td>
<td>2744</td>
</tr>
<tr>
<td>IBM01D</td>
<td>24600</td>
<td>5471</td>
<td>2948</td>
<td>46.12</td>
<td>2460</td>
<td>53.04</td>
<td>3219</td>
</tr>
<tr>
<td>IB3A2A</td>
<td>51221</td>
<td>13329</td>
<td>6082</td>
<td>71.49</td>
<td>4103</td>
<td>69.25</td>
<td>5671</td>
</tr>
<tr>
<td>IB3A2B</td>
<td>26068</td>
<td>6961</td>
<td>2623</td>
<td>62.43</td>
<td>2337</td>
<td>66.52</td>
<td>2755</td>
</tr>
<tr>
<td>IB3A2C</td>
<td>23309</td>
<td>5593</td>
<td>2339</td>
<td>59.53</td>
<td>1909</td>
<td>63.89</td>
<td>2018</td>
</tr>
<tr>
<td>IB3A2D</td>
<td>22923</td>
<td>6085</td>
<td>1992</td>
<td>70.73</td>
<td>1803</td>
<td>72.02</td>
<td>1011</td>
</tr>
<tr>
<td>IB3A3A</td>
<td>66251</td>
<td>15845</td>
<td>3184</td>
<td>79.65</td>
<td>4555</td>
<td>72.16</td>
<td>3197</td>
</tr>
<tr>
<td>IB3A3B</td>
<td>25850</td>
<td>6813</td>
<td>501</td>
<td>97.64</td>
<td>498</td>
<td>92.69</td>
<td>499</td>
</tr>
<tr>
<td>IB3A3C</td>
<td>53728</td>
<td>14661</td>
<td>1220</td>
<td>72.65</td>
<td>1282</td>
<td>71.36</td>
<td>1427</td>
</tr>
<tr>
<td>IB3A3D</td>
<td>31089</td>
<td>6766</td>
<td>1622</td>
<td>70.03</td>
<td>2076</td>
<td>69.32</td>
<td>2019</td>
</tr>
<tr>
<td>IB3A5A</td>
<td>129950</td>
<td>36089</td>
<td>7633</td>
<td>79.3</td>
<td>6626</td>
<td>82.03</td>
<td>6814</td>
</tr>
<tr>
<td>IB3A5B</td>
<td>52220</td>
<td>19093</td>
<td>3129</td>
<td>83.6</td>
<td>4172</td>
<td>78.14</td>
<td>2003</td>
</tr>
<tr>
<td>IB3A4C</td>
<td>61793</td>
<td>19269</td>
<td>2047</td>
<td>83.22</td>
<td>4008</td>
<td>70.06</td>
<td>3047</td>
</tr>
<tr>
<td>IB3A4D</td>
<td>48717</td>
<td>14541</td>
<td>4574</td>
<td>88.1</td>
<td>2216</td>
<td>84.54</td>
<td>3084</td>
</tr>
<tr>
<td>IB3A7A</td>
<td>131753</td>
<td>43454</td>
<td>9609</td>
<td>77.93</td>
<td>5608</td>
<td>86.9</td>
<td>9069</td>
</tr>
<tr>
<td>IB3A7B</td>
<td>74374</td>
<td>27412</td>
<td>3349</td>
<td>86.48</td>
<td>4468</td>
<td>82</td>
<td>2116</td>
</tr>
<tr>
<td>IB3A7C</td>
<td>62567</td>
<td>10845</td>
<td>6172</td>
<td>67.45</td>
<td>8815</td>
<td>52.89</td>
<td>4127</td>
</tr>
<tr>
<td>IB3A7D</td>
<td>41472</td>
<td>11707</td>
<td>2593</td>
<td>69.31</td>
<td>5334</td>
<td>54.44</td>
<td>3802</td>
</tr>
</tbody>
</table>