Performance Driven Multiway Partitioning*

Jason Cong and Sung Kyu Lim
UCLA Department of Computer Science, Los Angeles, CA 90095
{cong,limsk}@cs.ucla.edu

Abstract— Under the interconnect-centric design paradigm, partitioning is seen as the crucial step that defines the interconnect [1]. To meet the performance requirement of today’s complex design, performance driven partitioners must consider the amount of interconnect induced by partitioning as well as its impact on performance. In this paper, we provide new performance driven formulation for cell move based top-down multiway partitioning algorithms with consideration of the local and global interconnect delay. In our “constrained acyclic partitioning” formulation, cell moves are restricted to maintain acyclicity in partitioning solution to prevent cyclic dependency among cells in different partitions. In our “relaxed acyclic partitioning” formulation, acyclic constraints are relaxed to give partitioners capability of minimizing cutsize and delay. Our new acyclic constraint based performance driven multiway partitioning algorithm FLARE obtains (i) 4% to 13% better delay compared to the state-of-the-art cutsize minimization based hMetis [10] at almost no increase in cutsize, and (ii) 84% better cutsize compared to the state-of-the-art delay minimization based PRIME [2] at an expense of 16% increase in delay.

I. INTRODUCTION

Many sources including NTRS (National Technology Roadmap for Semiconductor) predict that 80% or more of the critical path delay will be directly linked to interconnect in deep submicron geometries. Thus, addressing interconnect issues in all steps involved in VLSI design process has become an essential goal. Under the interconnect-centric design paradigm, partitioning is seen as the crucial step that defines the interconnect [1]. To meet the performance requirement of today’s complex design, performance driven partitioners must consider the amount as well as performance-related quality of the interconnect induced by partitioning. Cutsizes minimization helps to lower the possibility of critical paths crossing partition boundary multiple times, thus improving performance. In addition, a proper model of delay estimation for partitioning has direct impact on delay minimization. Many proposed cutsize oriented partitioners do not consider delay, while many proposed delay oriented partitioners do not consider cutsize. As a result, there is a strong need for a performance driven partitioner that considers both cutsize and delay and provides smooth cutsize/delay tradeoff.

The performance driven circuit decomposition algorithms can be grouped into two categories: bottom-up clustering and top-down partitioning algorithms. Performance driven bottom-up circuit clustering problem is to group components in a circuit into clusters subject to an upper bound on the total area and/or total I/O pins in each cluster. The objective is to minimize the delay of the circuit. In [11], the authors proposed an efficient labeling based clustering algorithm to achieve the minimum delay for combinational circuits under simplistic delay model. [15, 13, 18] extend this work to consider more general delay model. Pan et al. [16] proposed a polynomial-time clustering algorithm for sequential circuits with retiming that achieves quasi-optimal delay under general delay model. The current state-of-the-art is established by PRIME [2] that provides significant space and time complexity improvement of [16] while maintaining quasi-optimal delay solutions. However, these methods face one or both of the following limitations: (i) they produce much worse cutsize compared to conventional top-down partitioning, which in turn translates into more routing area and congestion problem, (ii) it is hard to control area balance among partitions and sometimes fail to obtain exact number of partitions.

Performance driven top-down circuit partitioning problem has been studied actively especially during recent years. The problem is to divide a circuit into predetermined number of partitions while maintaining the area of each partition within user specified range. Unlike the conventional top-down partitioning algorithms [8, 10] that minimize cutsize only, the primary objective of performance driven algorithms is to minimize the delay of the circuit. Shih et al. [17] proposed an algorithm to guarantee that the delay between registers satisfies the timing constraint. Hwang and Gamal [9] showed that logic replication from one partition to another can improve cutsize and delay. Liu et al. [14] proposed an efficient algorithm to combine logic replication and retiming for bipartitioning. However, these algorithms may suffer a long runtime for large circuits and do not guarantee any optimality.

In this paper, we provide new performance driven for-
mulation for cell move based top-down multiway partitioning algorithms with consideration of the local and global interconnect delay induced by the partitioning. In our constrained acyclic partitioning formulation, cell moves are restricted to maintain acyclicity in partitioning solution to prevent cyclic dependency among cells in different partitions. In our relaxed acyclic partitioning formulation, acyclic constraints are relaxed to give partitioners capability of minimizing cutsize and delay. Our new acyclic constraint based performance driven multiway partitioning algorithm FLARE obtains (i) 4% to 13% better delay compared to the state-of-the-art cutsize minimization based Metis [10] at almost no increase in cutsize, and (ii) 84% better cutsize compared to the state-of-the-art delay minimization based PRIME [2] at the expense of 16% increase in delay.

The remainder of the paper is organized as follows. Section II presents problem formulation. Section III presents partitioning with acyclic constraints. Section IV provides experimental results. Section V concludes the paper with our ongoing research.

II. PROBLEM FORMULATION

A sequential gate-level circuit netlist NL can be represented as a directed retiming graph G(V, E) where V is the set of nodes representing gates in the circuit, and E is the set of edges representing the connections between gates. A directed edge e(u, v) denotes the connection from gate u to gate v. A fan-in set of vertex v is defined as FI(v) = {u | u ∈ V and e = (u, v) ∈ E}, and fan-out set of vertex v is similarly defined as FO(v) = {u | u ∈ V and e = (v, u) ∈ E}. A set of primary inputs is defined as PI = {v | v ∈ V and FI(v) = ∅}, and a set of primary outputs is defined as PO = {v | v ∈ V and FO(v) = ∅}.

A balanced duplication free K-way partitioning B = {B₁, B₂, ···, B_K} of given G(V, E, W) satisfies the following conditions:

- Bᵢ ∩ Bⱼ = ∅ for i ≠ j and B₁ ∪ B₂ ∪ ··· ∪ B_K = V
- αᵢ ≤ |Bᵢ| ≤ βᵢ for given αᵢ and βᵢ, 1 ≤ i ≤ K

We build dependency graph D(G, B) [4] as follows; each vertex in D(G, B) represents a partition in B, and a directed edge (Bᵢ, Bⱼ) exists if there exists an edge e = (x, y) ∈ E such that x ∈ Bᵢ and y ∈ Bⱼ. We call partitioning solution B acyclic if its dependency graph D(G, B) is a directed acyclic graph.

We measure delay for given partitioning solution B of a sequential circuit, denoted \( \phi(B) \), for performance evaluation of B. In general delay model [15, 16, 2], each node v has a delay of \( d(v) \), and each edge e(u, v) has a delay of \( d(e) \) defined as follows:

\[
d(e) = \begin{cases} 
D & \text{if } e = (u, v) \in E, u \in Bᵢ, v \in Bⱼ, i \neq j \\
0 & \text{if } e = (u, v) \in E \text{ and } u, v \in Bᵢ 
\end{cases}
\]

The delay of a path \( p = (u \rightarrow v) \) from \( u \in V \) to \( v \in V \), denoted \( d(p) \), is defined to be the sum of \( d(e) \) and \( d(g) \) along p. \( \phi(B) \) is the longest path delay among all combinational paths of one of the following types; \( PI \rightarrow PO, PI \rightarrow FF, FF \rightarrow PO, \) or \( FF \rightarrow FF \). The delay ratio corresponds to \( d(e)/d(g) \), which is equivalent to D in case we assume (i) \( d(g) = 1 \), and (ii) e connects vertices in different partitions. The delay ratio serves as a first-order approximation of how big global interconnect delay is compared to local interconnect delay.

III. PARTITIONING WITH ACYCLIC CONSTRAINTS

We present our multiway partitioning algorithm FLARE that simultaneously considers cutsize and delay minimizing under new acyclic constraint based formulation. In the constrained acyclic partitioning formulation, cell moves are restricted to maintain acyclicity in partitioning solution to prevent cyclic dependency among cells in different partitions. In the relaxed acyclic partitioning formulation, acyclic constraints are relaxed to give partitioners capability of optimizing delay. FLARE performs relaxed acyclic formulation based zLB bipartitioning algorithm on top of two-level cutsize oriented ESC [6] cluster hierarchy. This is then used as the bipartitioning engine for pairwise movement based multiway partitioning framework PM [5].

A. Constrained Acyclic Partitioning

Assuming topological ordering of V in G(V, E) is from partition B_f to B_i, we define the backward edge set V = \{e | e = (x, y) ∈ E, x ∈ Bᵢ, y ∈ B_f\}. Then B = (B_f, B_i) is acyclic iff V = ∅. We define A-counter for each vertex \( x \in V \), denoted \( a(x) \), as follows:

\[
a(x) = \begin{cases} 
\{\{y | y \in FO(x) \text{ and } y \in B_f\} \} & \text{if } x \in B_f \\
\{\{y | y \in FI(x) \text{ and } y \in B_f\} \} & \text{if } x \in B_i
\end{cases}
\]

The acyclic constraint will be violated, i.e. \( V \neq \emptyset \), if \( x \) is moved to the other partition when \( a(x) > 0 \). In other words, we can only move a cell \( x \) with \( a(x) = 0 \) under acyclic constraint. This is an additional constraint imposed on cell moves other than the conventional area balance constraint. An illustration of A-counters is shown in Figure 1-(a). If \( \delta \) denotes the maximum degree among vertices in \( V \), the computation of \( a(x) \) takes \( O(\delta) \) since it requires to examine all its neighbors. In addition, A-counters range from 0 to \( \delta \).

A naive way to incorporate A-counter based acyclic constraints into PM would be to find next legal cell move by searching down the bucket to obtain the first maximum gain cell \( x \) that has \( a(x) = 0 \). However, this is a very inefficient approach since \( O(\delta) \) computation of A-counter is required for all cells examined before we find the first

Local interconnect delay can be estimated and its average can be lumped into the gate delay \( d(v) \) for simplicity.
legal cell move. In order to maintain the linear complexity of single pass of FM under acyclic constraints, we provide two schemes. First, we precompute A-counters for all vertices before the cell move begins and incrementally update them upon each cell move. Second, in order to avoid searching down the bucket to obtain unconstrained cells, we use hybrid bucket that arranges unconstrained cells in LIFO order by inserting at the head of the list, while arranging constrained cells in FIFO order by inserting at the tail. The hybrid bucket structure also maintains counters at the header to keep track of how many unconstrained cells remain at each bucket.

B. Relaxed Acyclic Partitioning

One major drawback of partitioning under acyclic constraints is the restriction of cell moves due to the existence of acyclic constraints imposed on cells. We sacrifice cutsize by ignoring many beneficial cell moves in order to maintain acyclicity in the partitioning solution. As it will become evident from related experiments, cutsize degradation has negative effect on delay minimization. However, if we treat acyclicity to be an object to be optimized instead of a constraint to be satisfied at all times, we can optimize both cutsize and delay. Assuming topological ordering of \( V \) in \( G(V,E) \) is from partition \( B_f \) to \( B_t \), we use the size of backward edge set \( |V| = \{|e|e = (x,y) \in E, x \in B_t, y \in B_f\| \) to represent the degree of acyclic constraint violation. Then, we try to minimize \( |V| \) instead of requiring \( V \) to be \( \emptyset \).

We extend the definition of A-counter in previous section to formulate the reduction in \( |V| \). Assuming \( B_f \) to \( B_t \) topological ordering of \( V \), we define R-counter, denoted \( r(x) \), for each vertex \( x \in V \) as follows:

\[
r(x) = \begin{cases} 
|\{y|y \in FI(x) and y \in B_t\}| - a(x) & \text{if } x \in B_f \\
|\{y|y \in FO(x) and y \in B_f\}| - a(x) & \text{if } x \in B_t 
\end{cases}
\]

\( r(x) \) represents the reduction in \( |V| \) if \( x \) is moved to the other partition. An illustration of R-counters is shown in Figure 1-(b). Since acyclic constraint is relaxed, every cell move is legal if it does not violate the conventional area balance constraint. In addition, we can use the conventional LIFO bucket structure to manage cell gains. The computation and incremental update of R-counters can be done in a similar way as A-counters.

Note that R-counters do not represent real hyperedge cutsize reduction that FM [8] tries to minimize. Therefore, we incorporate R-counters into FM based cutsize gain formulation in the following way; if \( g(x) \) denotes conventional FM cutsize reduction gain, our new hybrid gain becomes:

\[
h(x) = \alpha \cdot g(x) + \beta \cdot r(x)
\]

where \( \alpha \) and \( \beta \) serve as weighting constants. \( h(x) \) represents real reduction in both hyperedge cutset and backward edge set if \( \alpha = \beta = 1 \), which is our empirical choice. We perform cell moves based on \( h(x) \) and update \( g(x) \) and \( r(x) \) together while visiting neighboring cells of \( x \). Note that the partitioner performs the conventional cutsize oriented moves if \( r(x) = 0 \), which in turn indicates that \( r(x) \) component has an effect of altering the conventional move sequence towards better delay result. It is possible to obtain cyclic partitioning result, i.e., \( |V| > 0 \), at the termination of partitioning. However, relaxed acyclic partitioning enables the partitioner to optimize cutsize without any restriction and at the same time prevent critical paths in the given circuit from being cut multiple times if not once in \( (B_f, B_t) \).

C. Summary of FLARE Algorithm

Our performance driven multiway partitioning algorithm FLARE first builds two-level cluster hierarchy with ESC [6] bottom-up clustering algorithm. Then, we adopt two-phase top-down partitioning scheme to integrate clustering into partitioning; we perform our performance driven xLR partitioning first on the clustered circuit for global optimization and then on the declustered circuit for local refinement. We develop xLR algorithm by incorporating two types of enhancement to FM [8] algorithm, (i) LR [3] for cutsize minimization, (ii) relaxed acyclic partitioning based R-counter \( r(x) \) discussed in Section III-B for simultaneous cutsize and delay minimization. We provide comprehensive experimental justification for each enhancement in xLR in the following Section IV-B. Lastly, the two-phase partitioning algorithm based on the combination of xLR and ESC is used as the bipartitioning engine for pairwise movement based multiway partitioning framework FM [5]. Interested readers are referred to [5] for more details on FM framework.

An overview of FLARE algorithm is shown in Figure 2. FLARE maintains (i) undirected graph \( U \) for executing ESC clustering algorithm, (ii) directed graph \( G_0 \) and \( G_1 \) from which relaxed acyclic constraint based \( r(x) \) is computed, and (iii) hypergraph \( H_0 \) and \( H_1 \) from which conventional cutsize gain \( g(x) \) is computed (line 1 to 3). The cluster-
FLARE(NL.K)

Input: sequential circuit netlist NL, # of block K
Output: partition B, cutsizes c(B), and delay φ(B)

1. obtain undirected graph U from NL;
2. obtain directed graph G0 from NL;
3. obtain bipartition H0 from NL;
4. $C = \text{ESC}(U)$;
5. obtain $G_1$ and $H_1$ from $C$;
6. remove $FF$ from $G_0$;
7. $T = \text{topological sorting of } (G_1)$;
8. $B_1 = \text{split } T \text{ into } K \text{ parts}$;
9. while (gain > 0)
   10. obtain block pairing with PM;
   11. compute $g(x)$ from $H_1$;
   12. compute $r(x)$ from $G_1$;
   13. while (exists legal cell move)
      14. perform cell move;
      15. update $g(x)$ and $r(x)$;
      16. update $B_1$;
      17. project $B_1$ to $B$;
      18. compute $g(x)$ from $H_0$;
      19. compute $r(x)$ from $G_0$;
      20. while (exists legal cell move)
         21. perform cell move;
         22. update $g(x)$ and $r(x)$;
         23. update $B$;
   24. compute $c(B)$ from $H_0$;
   25. compute $φ(B)$ from $G_0$;
26. return $B$, c(B), and φ(B)

Fig. 2. Overview of performance-driven partitioning algorithm FLARE. FLARE performs relaxed acyclic formulation based xLR bipartitioning algorithm on top of two-level cutsize oriented ESC [5] cluster hierarchy. This is then used as the bipartitioning engine for pairwise movement based multiway partitioning framework PM [5].

ing solution $C$ from ESC algorithm is used to contract $G_0$ and $H_0$ to obtain $G_1$ and $H_1$ (line 4 and 5). Under the assumption that all cycles in $G$ involve flip-flops, we convert $G_0$ into directed acyclic graph by removing flip-flop set $FF$ from $G_0$ (line 6). We obtain the initial acyclic $K$-way partition $B_1$ from topological sorting of vertices in $G_1$ (line 7 to 8). Our relaxed acyclic partitioning algorithm xLR applied on the clustered netlist (line 11 to 16) and then on the original netlist (line 18 to 23) according to two-phase partitioning framework.

IV. Experimental Result

A. Experimental Setting

We implemented our algorithms in C++/STL and tested on SUN ULTRA SPARC60 at 360Mhz. We obtained the latest binary executable of hMetis [10] (v1.5.3) from the website for the evaluation. The benchmark set consists of 7 ISCAS circuits and 4 large scale industrial designs provided by our industrial sponsor. Table I shows the characteristics of these circuits. We report csize, delay, and runtime from 16-way partitioning results. All algorithms mentioned in this section obtain 16 partitions by recursively applying bipartitioning, except for LR/ESC-PM [6] and FLARE that obtain 16 partitions simultaneously using PM [5] framework. The bipartitioning area balance skew is set to [0.45, 0.55], which is equivalent to $0.45^3 = 0.041, 0.55^3 = 0.092$ for the 16-way partitioning. Runtimes are measured in seconds, and cizes are based on Cost-1 metric that counts the number of hyperedges that span more than single partition. We assume that all gates have unit area and unit delay, while primary inputs, primary outputs and flip-flops have no area and no delay. We apply retiming [12] on all algorithms used in our experiments as a post delay refinement process. We use $D = 5$ for the current 0.18μm technology throughout the entire experiments unless specified otherwise.

<table>
<thead>
<tr>
<th>name</th>
<th>#GA</th>
<th>#PI</th>
<th>#PO</th>
<th>#FF</th>
<th>#net</th>
</tr>
</thead>
<tbody>
<tr>
<td>s9234</td>
<td>1290</td>
<td>28</td>
<td>39</td>
<td>135</td>
<td>1492</td>
</tr>
<tr>
<td>s5738</td>
<td>1443</td>
<td>35</td>
<td>49</td>
<td>163</td>
<td>1690</td>
</tr>
<tr>
<td>s13207</td>
<td>3146</td>
<td>59</td>
<td>152</td>
<td>486</td>
<td>3843</td>
</tr>
<tr>
<td>s15850</td>
<td>3784</td>
<td>76</td>
<td>150</td>
<td>515</td>
<td>4525</td>
</tr>
<tr>
<td>bigkey</td>
<td>8599</td>
<td>228</td>
<td>197</td>
<td>224</td>
<td>9248</td>
</tr>
<tr>
<td>s38584</td>
<td>13209</td>
<td>38</td>
<td>304</td>
<td>1423</td>
<td>14974</td>
</tr>
<tr>
<td>cluna</td>
<td>30552</td>
<td>61</td>
<td>82</td>
<td>33</td>
<td>30728</td>
</tr>
<tr>
<td>ind1</td>
<td>28780</td>
<td>2630</td>
<td>3242</td>
<td>603</td>
<td>36255</td>
</tr>
<tr>
<td>ind2</td>
<td>26060</td>
<td>2772</td>
<td>6242</td>
<td>1755</td>
<td>36829</td>
</tr>
<tr>
<td>ind3</td>
<td>52197</td>
<td>2801</td>
<td>3070</td>
<td>2001</td>
<td>60669</td>
</tr>
<tr>
<td>ind4</td>
<td>101531</td>
<td>4155</td>
<td>4547</td>
<td>8333</td>
<td>118566</td>
</tr>
</tbody>
</table>

B. Overall Comparison

We use two existing cutsize oriented cell move based partitioning algorithms PM [8] and LR [3] to evaluate our constrained and relaxed acyclic partitioning formulation. PM is the standard cell move based algorithm, whereas LR is an enhancement of PM based on modified gain function. The naming convention of algorithms we test is of [c|x][PM|LR] form. The prefix [c|x] respectively denotes constrained and relaxed acyclic constraint partitioning. The objective of adding acyclic constraints is to improve delay at the expense of csize increase, while relaxing acyclic constraint is to release restriction on cell moves to minimize both csize and delay. Indeed, we observe the corresponding trends from Table II. We note (i) constrained acyclic partitioners cPM and cLR improve delay while degrading csize, (ii) relaxed acyclic partitioners xPM and xLR improve bad csize results by cPM and cLR while maintaining delay results.

Table III reveals the overall csize and delay comparison among (i) csize oriented PM, hMetis, and
TABLE II
Cutsize, delay, and runtime reduction trends on 16-way partitioning result of constrained and relaxed acyclic partitioning algorithms. The cutsize result is based on Cost-1 metric, and the delay ratio $D$ is set to 5. TIME denotes total runtime including partitioning and retiming in seconds.

<table>
<thead>
<tr>
<th>ctk</th>
<th>FM based Partitioning</th>
<th>LR based Partitioning</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FM</td>
<td>cFM</td>
</tr>
<tr>
<td></td>
<td>cut</td>
<td>dly</td>
</tr>
<tr>
<td>s9234</td>
<td>148</td>
<td>35</td>
</tr>
<tr>
<td>s5378</td>
<td>201</td>
<td>33</td>
</tr>
<tr>
<td>s3207</td>
<td>238</td>
<td>65</td>
</tr>
<tr>
<td>s55850</td>
<td>305</td>
<td>67</td>
</tr>
<tr>
<td>bigkey</td>
<td>78</td>
<td>22</td>
</tr>
<tr>
<td>s38584</td>
<td>475</td>
<td>52</td>
</tr>
<tr>
<td>clma</td>
<td>1031</td>
<td>89</td>
</tr>
<tr>
<td>ind1</td>
<td>1842</td>
<td>474</td>
</tr>
<tr>
<td>ind2</td>
<td>1355</td>
<td>55</td>
</tr>
<tr>
<td>ind3</td>
<td>4415</td>
<td>785</td>
</tr>
<tr>
<td>ind4</td>
<td>6887</td>
<td>124</td>
</tr>
<tr>
<td>TOTAL</td>
<td>17155</td>
<td>1801</td>
</tr>
<tr>
<td>TIME</td>
<td>3372</td>
<td>1234</td>
</tr>
</tbody>
</table>

LR/ESC-$\text{FM}$, (ii) delay oriented PRIME [2] and our FLARE algorithm. FLARE performs relaxed acyclic formulation based $\text{xLR}$ partitioning algorithm on top of two-level cutsize oriented ESC [6] cluster hierarchy. This is then used as the bipartitioning engine for pairwise movement based multiway partitioning framework $\text{FM}$ [5]. First of all, FLARE obtains (i) better delay compared to the state-of-the-art cutsize oriented $\text{hMetis}$ [10] at almost no increase in cutsize, and (ii) significantly better cutsize compared to the state-of-the-art quasi-optimal delay oriented PRIME at the expense of 16% increase in delay. Secondly, the conventional cutsize minimization partitioning $\text{hMetis}$ improves both the cutsize and delay of $\text{FM}$. This illustrates the side-effect of cutsize minimization objective on delay minimization. However, we show that the delay of $\text{hMetis}$ can still be further improved by FLARE while maintaining comparable cutsize quality.

The global vs local interconnect delay ratio $D$ is expected to increase as the technology advances into deeper sub-micron. Then, the delay advantage of FLARE over $\text{hMetis}$ is expected to increase from 4% to 13% when $D$ increases from 5 (estimated for 0.18$\mu$m technology) to 16 (estimated for 0.07$\mu$m technology). Interested readers are referred to our technical report for more details [7].

V. CONCLUSION AND ONGOING WORK

In this paper, we provided two new performance driven formulations for cell move based top-down multiway partitioning algorithms for sequential circuits; constrained and relaxed acyclic partitioning. The objective of adding acyclic constraints was to improve delay at the expense of cutsize increase, while relaxing acyclic constraint was to release restriction on cell moves to minimize cutsize and delay. We develop an efficient multiway partitioning algorithm FLARE that simultaneously considers cutsize and delay minimization under new acyclic constraint based formulation. Our ongoing study includes performance driven mincut placement based on our new performance driven formulation.

REFERENCES


<table>
<thead>
<tr>
<th></th>
<th>Cutsize Minimization</th>
<th>Delay Minimization</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FM</td>
<td>hMetis</td>
</tr>
<tr>
<td></td>
<td>ckt</td>
<td>cut</td>
</tr>
<tr>
<td>s9234</td>
<td>148</td>
<td>35</td>
</tr>
<tr>
<td>s3378</td>
<td>201</td>
<td>33</td>
</tr>
<tr>
<td>s13207</td>
<td>238</td>
<td>65</td>
</tr>
<tr>
<td>s15850</td>
<td>305</td>
<td>67</td>
</tr>
<tr>
<td>big key</td>
<td>78</td>
<td>22</td>
</tr>
<tr>
<td>s38584</td>
<td>475</td>
<td>52</td>
</tr>
<tr>
<td>clma</td>
<td>1031</td>
<td>89</td>
</tr>
<tr>
<td>ind1</td>
<td>1842</td>
<td>474</td>
</tr>
<tr>
<td>ind2</td>
<td>1535</td>
<td>55</td>
</tr>
<tr>
<td>ind3</td>
<td>4415</td>
<td>785</td>
</tr>
<tr>
<td>ind4</td>
<td>6887</td>
<td>124</td>
</tr>
<tr>
<td>TOTAL</td>
<td>17155</td>
<td>1801</td>
</tr>
<tr>
<td>RATIO</td>
<td>2.33</td>
<td>1.26</td>
</tr>
<tr>
<td>TIME</td>
<td>3372</td>
<td>3694</td>
</tr>
</tbody>
</table>


