# Two-Dimensional Layout Migration by Soft Constraint Satisfaction

Qianying Tang, Jianwen Zhu
Electrical and Computer Engineering
University of Toronto, Ontario M5S 3G4, Canada
{tangq, jzhu}@eecg.toronto.edu

## ABSTRACT

Layout migration has re-emerged as an important task due to the increasing use of library hard intellectual properties. While recent advances of migration tools have accommodated new metrics, the underlying engine is based on the one-dimensional (1-D) layout compaction algorithm, largely due to its efficiency compared to its two-dimensional (2-D) counterpart. In this paper, we propose a new method that can overcome the artificial constraints introduced by the 1-D compaction algorithm, thereby effectively achieving the quality of 2-D compaction, yet keeping the computational cost almost as low as 1-D compaction. Our method is based on the application of soft constraints, or artificial constraints that are initially relaxed, and gradually tightened to be satisfied. We demonstrate the effectiveness of our approach by successfully solving the difficult 1-D compaction instances we found in the migration of Berkeley low power library, originally developed for 1.2um MOSIS process, into TSMC 0.25um and 0.18um technology.

## 1. INTRODUCTION

It is generally felt that the complexity involved in systems-on-chip (SOC) design can only be tamed by intellectual-property (IP) based design, where the reuse of existing components is advocated to boost design productivity. Two forms of IP are generally used: *soft IPs*, delivered in the form of synthesizable RTL, and *hard IPs*, delivered in the form of layout. Since soft IPs will be re-synthesized by the SOC integrator for their chosen technology, they are favored for their portability. On the other hand, soft IPs are only limited to digital logic and SOC is by no means only a sea of gates. In fact, 70% of the silicon area will be occupied by hard IPs such as SRAMs, FIFOs, CAMs, datapaths and analog front-ends, which are typically designed by abutting a *library* of hand crafted cells. With custom layout design, hard IPs can be delivered with higher performance and better predictability than the soft IPs, and therefore less effort in the SOC integration.

The major bottleneck that prevents the wide adoption of hard IPs is, as its name suggests, the dependency of layout on process. The cost of initial custom layout design is already very high. This applies even to IPs such as standard cells, which are used to be considered simple compared to other IP libraries. Today's standard cell libraries contain hundreds of cells with different functions and driving strengths. Most fabless companies choose to use libraries offered by hard IP companies precisely because of the high cost associated with library development. To make things worse, manufacturing processes are updated every 18 month, each with a different set of design rules. This makes the development cost of hard IPs too high even for hard IP vendors, since they have to offer different versions for different foundries as well. Automatic layout migration technology, which can amortize the high development cost associated with custom design across different foundries and processes, is therefore crucial for the sustained growth of hard IPs for the small, and the viability of IP-based design for the large.

Layout migration has been studied decades ago as the layout compaction problem. There is, however, a re-emerging interest for new migration algorithms in order to accommodate the need of new metrics [1, 2], the adherence to library layout architecture [3], the special treatment of analog IPs [4], and the new challenges of complex design rules and resolution enhancement technologies [5]. In this paper, we revisit the subject of migration engine, which is crucial for the runtime efficiency of any new migration framework. The migration engines, traditional or new, are formulated either as two successive one-dimensional (1-D) compaction problems (X direction followed by Y direction), or one two-dimensional (2-D) compaction problem. The former is known for its speed and wide adoptance, but also poor layout quality for some cases. The latter is known for its better quality, but extremely long run time.

In this paper, we present a new technique such that the 2-D compaction quality can be achieved by iterative 1-D compaction. The key idea is to "soften" those constraints generated as artifacts of 1-D compaction – instead of hard constraints that must be satisfied, they can be violated. During the iteration, these constraints are gradually tightened, eventually leading to a feasible solution. Our experience indicates that for those hard instances we encounter with the traditional 1-D compaction strategy, the proposed method converge in only a few iterations.

The rest of the paper is organized as follows. We first review the related work in Section 2. We then describe our migration framework in Section 3. In Section 4, we describe in detail how artificial constraints are treated. We give experimental results in Section 5 before we draw conclusions.

## 2. RELATED WORK

Automatic layout migration was among the oldest CAD problems investigated and a large body of research was carried out under the layout compaction problem. A good survey of layout compaction can be found in [6] and [7]. The compaction methodologies include: shear-line approach, virtual grid approach and constraint graph approach. Our migration tool extends the constraint graph approach, which represents constraints only as distance inequalities, into general linear formula, which are needed in some occasions.

The two-dimensional compaction problems were examined in [8, 9, 10]. Proposed methods include zone-refining and simulated annealing. However, the zone-refining approach requires a reasonably optimal layout to start with, while the simulated annealing method may requires a significant running time at each cooling stage. Our migration method works directly on the original layout to be migrated, and has a runtime comparable to 1-D compaction.

The *geometric closeness objective function* for migration was proposed in [2]. It is an extension to the *minimum perturbation objective function* proposed by [1]. The 1-D migration tool for datapath layout with layout architecture considerations such as routine grid, power line, and pitch matching was explored in [3].

## 3. MIGRATION FRAMEWORK

In this section, we describe our integer linear programming (ILP) based migration framework. The migration engine of our tool assumes that the leaf cell layout is Manhattan. The migration engine can therefore generate the migrated layout by determining new positions of horizontal and vertical edges of all geometries.

Our migration framework is iterative. Each iteration resembles a classical 1-D migration framework: it first migrates along the X direction, or determining positions of vertical edges, and then the Y direction, or determining positions of horizontal edges. Without loss of generality, in the discussion that follows, we assume migration in the X direction only.

A *constraint-based migration* of a leaf cell can be formulated as an integer linear programming (ILP) problem:

$$\boxed{\begin{array}{rl} \textit{minimize} & o^T x \\ \textit{subject to} & Ax \geq b \\ & x \geq 0 \end{array}}$$

Here $x$ is a vector of variables to be determined, and in the simplest case would just be coordinates for all vertical edges. Vectors $o$ represent the coefficients of $x$ in the objective function of optimization. $A$ is the constraints, each row of which represents coefficients of $x$ in an inequality. Typically, an equality will be in the form $x_i - x_j \geq b_k$, where the constant $b_k$ represents the minimum distance between two edges.

The constraint generation is a key component of the migration process. It discovers the relative position requirements between layout edges coming from different sources. One source of constraints is imposed by the *design rules*. The second source of constraints is imposed by *high-level architectural requirements*, which have to do with the global structure of the entire library, rather than individual cells. This includes the routing track constraints, the power line constraints, the transistor size constraints and the port matching constraints. The third source of constraints, unfortunately, are *artificial*. This is the primary reason that layout generated by 1-D compaction is inferior to 2-D compaction. We examine the sources of such constraints and elaborate our treatment in the next section.

The migrated layout is largely influenced by the choice of the objective function. The traditional minimum area objective function is defined by

$$x_r - x_l \tag{1}$$

where $x_r$ is the X coordinate of the right most vertical edge in the layout, and $X_l$ is the X coordinate of left most vertical edge, as illustrated in Figure 1 (a).

To preserve advanced design considerations in the original layout, as argued earlier, we define a new objective function, called the *geometric closeness* as follows:

$$\sum \left| (x_{ri} - x_{li}) - (x_{ri}^{old} - x_{li}^{old}) \right| \tag{2}$$

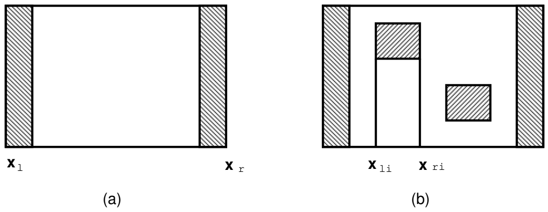



**Figure 1: (a) Minimum area layout objective function. (b) Geometric closeness objective function.**

Here, $x_{ri}$ and $x_{li}$ are the X coordinates of the right and left edges of each rectangle in the layout. The constants $x_{ri}^{old}$ and $x_{li}^{old}$ are the X coordinates of the right and left edges of the corresponding rectangle in the original layout, as shown in Figure 1 (b). Note that space is explicitly represented as rectangles and are counted in the geometric closeness calculation. This is most conveniently achieved by using the corner-stitched layout representation [11].

To linearize, or to remove the absolute value computation in (2), we use a method similar to [1] by introducing two variables $R_i$ and $L_i$ for each rectangle, such that (3) are introduced as constraints,

$$\begin{aligned} R_i &\geq x_{ri} - x_{li} \\ R_i &\geq x_{ri}^{old} - x_{li}^{old} \\ L_i &\leq x_{ri} - x_{li} \\ L_i &\leq x_{ri}^{old} - x_{li}^{old} \end{aligned} \tag{3}$$

and the objective function is replaced by (4):

$$\sum (R_i - L_i) \tag{4}$$

## 4. SOFT CONSTRAINT BASED MIGRATION

In the 1-D migration strategy, X direction and Y direction are migrated separately, and artificial constraints are introduced to ensure that there is no arbitrary movement of layout rectangles in one direction when performing migration on the other direction. However, these artificial constraints compromise the optimality of the migrated layout. An improvement is to relax these constraints by introducing additional variables, called *elastic* variables. With the elastic variables, the artificial constraints can be violated under a controlled way.

In a corner-stitched representation, layout rectangles, called tiles, are organized in different planes. Accordingly, design rules are imposed between tiles within the same plane, called *intraplane* rules, and across different planes, called *interplane* rules. In the 1-D migration framework, artificial constraints are introduced for both intraplane rules, and interplane rules.

When migration is performed in the X direction, constraint generation algorithm such as the depth-K shadowing closure [2] is used. In such algorithm, constraints are generated according to the relative position of tiles according to their Y coordinates. When we proceed to migrate in the Y direction, these assumptions have to be preserved. Otherwise the solution along the X direction will no longer be valid, or DRC-clean. For example, in Figure 2, the following artificial constraints have to be generated during the Y compaction.

$$
\begin{aligned}
y_{2t} - y_{1t} &\geq 0 \\
y_{1b} - y_{3b} &\geq 0 \\
y_{1b} - y_{4b} &\geq 0.
\end{aligned}
$$

Here for the ith tile, $y_{it}$ denotes the Y coordinate of the top edge, while $y_{ib}$ denotes the Y coordinate of the bottom edge.

We relax these constraints by introducing three additional variables $s_1, s_2, s_3$ and rewrite the above constraints as follows

$$
\begin{aligned}
y_{2t} - y_{1t} &\geq 0 - s_1 \\
y_{1b} - y_{3b} &\geq 0 - s_2 \\
y_{1b} - y_{4b} &\geq 0 - s_3 \\
s_1, \ldots, s_3 &\geq 0
\end{aligned}
$$

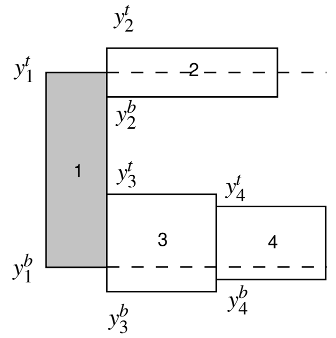



**Figure 2: Intra plane artificial constraints.**

Artificial constraints associated with interplane rules are similar, except that in this case, the source edge and the constraint area are on separate planes, which are migrated independently. We therefore need two additional artificial constraints on the X direction to ensure that the relative position of tiles in the constraint area to the source edge are kept. For example, in Figure 3, the two additional artificial constraints are $x_{edge} - x_{1l} \geq 0, x_{1r} - x_{edge} \geq 0$, where $x_{edge}$ denotes the position of the source edge, $x_{1l}$ and $x_{ir}$ denotes the left and right edge of tile 1 respectively. Similarly, these constraints can be formulated as soft constraints as follows,

$$
\begin{aligned}
x_{edge} - x_{1l} &\geq -s_1 \\
x_{1r} - x_{edge} &\geq -s_2 \\
s_1 &\geq 0 \\
s_2 &\geq 0
\end{aligned}
$$

We now consider how we can control the elastic variables so that ultimately we can obtain DRC-clean layout. The idea is to use the objective function. Our objective function is formulated as

$$o^T x + C \sum_i s_i,$$

Here the term $o^T x$ is the same as the objective function of the original one in 1-D migration. The term $C \sum_i s_i$ accounts for the contributions of soft constraints. Variables $s_i$ denotes the elastic variables introduced in generating soft constraints. The scalar $C$ is used as the weighting factor.

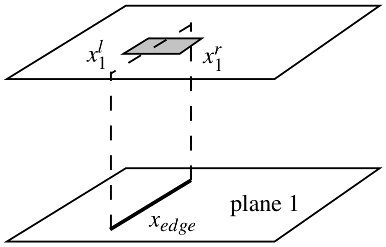



**Figure 3: Interplane artificial constraints.**

It is easy to see that if the value of $C$ is small, we practically relax all the artificial constraints introduced and approximating a 2-D migration. When the value of $C$ approaches infinity, requiring all the elastic variables $s_i$ to approach zero, the problem becomes the same as the original 1-D migration.

To migration the cell with soft constraints, we adopted an iterative optimization procedure in which the value of the weight factor $C$ is adjusted at each iteration. Starting with small value of $C$, the objective function described above and both the soft and hard constraints are fed into a ILP solver and solutions for each edge position are found. We migrate the cell based on the solution. Next we check to see if the cell is free of design rule violations. If so, we stop at this point, and return the migrated cell. Otherwise, a new set of soft and hard constraints for the migrated cell are derived in a similar manner for the original cell. The value of $C$ in the objective function is updated with an appropriate step size. The new objective functions and constraints are again fed to ILP solvers to find solutions for next migration. The entire procedure is repeated until the migrated layout is free of design rule violations. The scheme for updating $C$ can affect the optimality of the solution obtained. In our experiments, we used the following scheme:

| | |
|---|---|
| initial value | $C = 0.001$ |
| 1st iteration | $C = 1$ |
| succeeding iterations | $C_{new} = 10 * C_{old}$ |

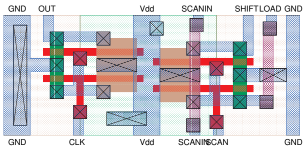
# 5. EXPERIMENTAL RESULTS

To test the effectiveness of our tool, we apply our tool on the low-power standard cell library and datapath library developed by Burd [12] at University of California, Berkeley. This library was based on SCMOS 1.2$\mu$m technology. The standard cell library contains 94 cells, including gates performing common logic functions with different drive strength. The datapath library contains 285 cells, including those to construct adders, shifters, multiplexors and register files. Both libraries are silicon proven and have been used at Berkeley for at least a dozen chips (`www.faqs.org/faqs/lsi-cad-faq/part4`). Our targeted processes are TSMC 0.25 $\mu$m and TSMC 0.18$\mu$m technologies. All cells are migrated successfully: they are all DRC clean and do not require manual editing. All experiments are carried out on a SUNBlade 100 system running at 500 MHZ.

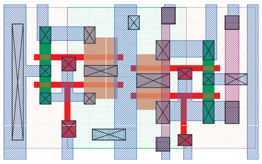
## 5.1 Sample Layout

In this section we show sample migration results to demonstrate the effectiveness of the proposed method.

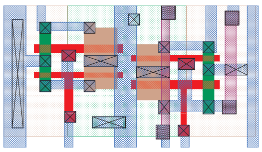
We compare results generated by the regular 1-D framework and our soft constraint based 2-D framework. Figure 4 (a) shows the layout of the *scantspcr_cs1* cell in the library. Figure 4 (b) and (c) show the migrated layout by the traditional 1-D framework and by our soft constraint based 2-D migration. The widths of *metal*1 that are noticeably stretched in the 1-D migration layout. This is not the case for 2-D migration. In addition, several contact stretches observed in the 1-D migration are also avoided in our method.




(a) Original layout



(b) 1-D migration result



(c) 2-D migration result

**Figure 4: Migration of scantspcr cell.**

Similarly, Figure 5 shows the migration result for *shcs* cell. The width of the vertical metal 1 in the middle of the cell in (c) is not as stretched as in (b). In addition, the aspect ratios of the contacts between metal1 and p-diff is better preserved in (c).

Figure 6 shows the migration result for cell *csamsbodd*. The stretch in the y-direction is significantly reduced in the new migration method. It is apparent that the cell migrated in the new method not only resembles the original cell much better, but also has significantly smaller area.

## 5.2 Quantitative Results

In this section, we provide a more comprehensive, quantitative results.

We demonstrate the effectiveness of soft constraint based 2-D migration in Table 1. The value of the geometric closeness (GC) objective function are evaluated for (a) traditional 1-D migration, (b) each iteration in the soft constraint based migration, and (c) overall value for the soft constraint based migration. The overall value of the objective function for soft constraint based migration are calculated by adding that of each iterations. Comparing the results in column (a) and column (c) suggests that in general, our method can find a better solution to the minimization problem than traditional 1-D migration. Furthermore, we note that at most three iterations are required to obtain the final solution, which means that the run time for our method is at most four times longer than the regular 1-D migration.

Finally in column 9-13 of Table 1 list the detailed area result: Column 9 (orig) corresponds to the layout area after linear scaling, which is not DRC-clean. Column 10 (MA) lists the area obtained by minimum area layout compaction. Column 11 (MP) lists the area obtained by minimum perturbation metric proposed in [1]. Column 12 (GC) lists the area obtained by 1-D geometric closeness metric. The last column (2D) lists the area obtained by geometric closeness metric under the 2-D iterative framework. It can be observed that our solution is approximately 11% better than minimum perturbation method in area, and only 7.5% worse than minimum area compaction, even though area is not our primary optimization goal.

# 6. CONCLUSION

In conclusion, we have argued that the migration technology is essential for the success of hard IPs in general, and library IPs in particular. We identified the inherent problem of artificial constraints that arise in the popular 1-D migration framework and proposed a solution based on iterative soft constraint satisfaction. Based on our study, we conclude that the proposed strategy is effective, as the difficult instances we encounter using the classical 1-D framework can be solved, and efficient, as our framework only needs a few iterations to converge, therefore has comparable runtime to the 1-D framework.

# 7. REFERENCES

[1] Fook-Luen Heng, Zhan Chen, and Gustavo E.Tellez, "A VLSI artwork legalization technique based on a new criterion of minimum layout perturbation," in *International Symposium on Physical Design (ISPD)*, 1997, pp. 116–121.

[2] Fang Fang and Jianwen Zhu, "Calligrapher: A new layout migration engine based on geometric closeness," in *International Symposium on Quality Electronic Design (ISQED)*, 2004, pp. 25–30.

[3] Fang Fang and Jianwen Zhu, "Automatic migration of datapath IP libraries," in *Proceeding of Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2004.

[4] Nuttorn Jangkrajarng, Sambuddha Bhattacharya, Roy Hartono, and C.-J. Richard Shi, "Automatic analog layout retargeting for new processes and device sizes," in *International Symposium in Circuit and Systems (ISCAS)*, 2003.

[5] Fook-Luen Heng, Lars Liebmann, and Jennifer Lund, "Application of automated design migration to alternating phase shift mask design," in *International Symposium on Physical Design (ISPD)*, 2001, pp. 116–121.

[6] Y. Eric Cho, "A subjective review of compaction," in *Proceeding of the Design Automation Conference (DAC)*, 1985, pp. 396–404.

[7] David G. Boyer, "Symbolic layout compaction review," in *Proceeding of the Design Automation Conference (DAC)*, 1988, pp. 383–389.

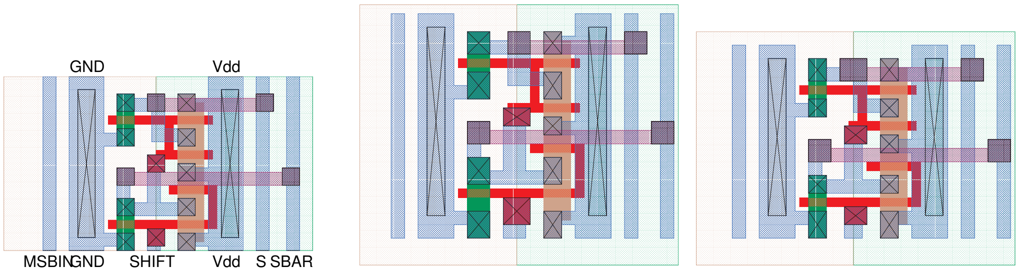
[8] Hyunchul Shin, Alberto L. Sangiovanni-Vincentelli, and Carlo H. Sequin, "Two-dimensional compaction by zone refining," in *Proceeding of the Design Automation Conference (DAC)*, 1986, pp. 115–122.

[9] H. Shim and C.-Y. Lo, "An efficient two-dimensional layout compaction algorithm," in *Proceeding of the Design Automation Conference (DAC)*, 1989, pp. 290–295.

[10] R.C. Mosteller, "Monte carlo methods for 2-d compaction," Tech. Rep., California Institute of Technology, 1986.

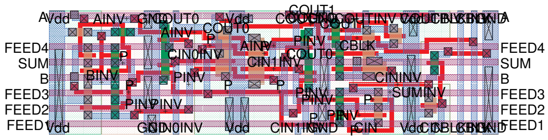
[11] John K.Ousterhout, "Corner stitching: A data-structuring technique for VLSI layout tools," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 87–100, 1984.

[12] Tom Burd, "Very low power cell library," Tech. Rep., University of California, Berkeley, 1995.

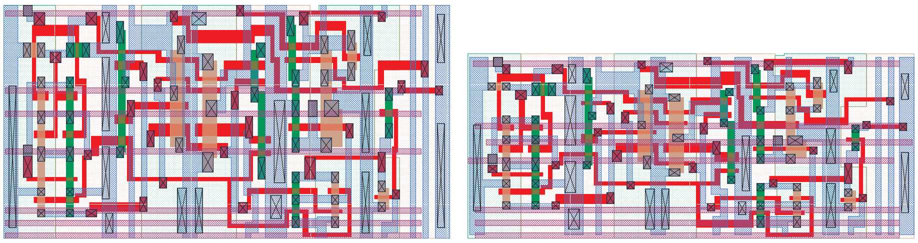



(a) Original layout (b) 1-D migration result (c) 2-D migration result

**Figure 5: Migration of shcs1 cell.**




(a) Original layout



(b) 1-D migration result (c) 2-D migration result

**Figure 6: Migration of csamsbodd cell.**

**Table 1: Area results and 2-D/1-D comparison.**

| Cell Name | No. of tiles | (a) GC change X/Y | (b) GC layout change for soft constraint(X/Y) | | | | (c) total GC change X/Y | (d) area | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | iter 1 | iter 2 | iter 3 | iter 4 | | orig | MA | MP | GC | 2D |
| cnt_top | 117 | 10/48 | 10/21 | 0/7 | | | 10/28 | 3040 | 2400 | 4000 | 4025 | 3864 |
| csainbuf | 192 | 57/237 | 38/110 | 23/23 | 0/93 | | 61/226 | 6264 | 3744 | 8388 | 8424 | 7956 |
| csalsbeven | 1053 | 759/3289 | 425/1709 | 34/68 | 167/464 | 50/285 | 676/2526 | 14848 | 27963 | 27963 | 28548 | 25920 |
| csamsbodd | 1180 | 920/4893 | 379/1708 | 84/35 | 166/310 | 53/392 | 682/2445 | 14848 | 32370 | 32370 | 32370 | 24644 |
| invs | 166 | 227/57 | 36/53 | 154/2 | 8/0 | | 198/55 | 2560 | 2142 | 2688 | 2814 | 2814 |
| mux2 | 343 | 224/347 | 21/216 | 10/121 | 40/0 | | 71/337 | 3584 | 4644 | 4816 | 4988 | 4930 |
| mux2_cs1 | 165 | 62/56 | 42/49 | 28/8 | 1/6 | | 43/54 | 1680 | 1890 | 2030 | 2100 | 2065 |
| mux2_cs2 | 167 | 61/66 | 41/54 | 28/13 | 2/6 | | 43/60 | 1680 | 1995 | 2030 | 2100 | 2100 |
| mux2_cs3 | 252 | 178/181 | 65/92 | 110/31 | | | 175/123 | 3080 | 3762 | 3828 | 3960 | 3599 |
| mux2_cs4 | 354 | 197/197 | 86/109 | 5/22 | 4/42 | | 100/197 | 4368 | 5369 | 5369 | 5551 | 5349 |
| nand2lp | 226 | 171/161 | 46/110 | 101/24 | | | 147/134 | 2944 | 3196 | 3243 | 3196 | 3012 |
| noror2lp | 265 | 265/185 | 131/60 | 22/38 | 22/24 | | 175/122 | 3392 | 4032 | 4032 | 4032 | 3920 |
| or2blp | 259 | 226/185 | 38/99 | 153/34 | 15/60 | | 206/193 | 3136 | 3876 | 3876 | 3876 | 3876 |
| rf0 | 194 | 240/61 | 25/61 | 200/0 | 12/0 | | 237/61 | 2688 | 2928 | 3072 | 3216 | 3016 |
| scantspcr_cs1 | 280 | 127/442 | 92/222 | 23/93 | | | 115/315 | 3204 | 5040 | 5040 | 5152 | 4692 |
| shcs1 | 219 | 113/286 | 64/108 | | | | 64/108 | 2840 | 3538 | 4118 | 4118 | 3692 |
| shl1bit | 239 | 415/14 | 56/12 | 328/5 | 0/2 | | 384/14 | 4544 | 3224 | 4615 | 4736 | 4736 |
| shl1end | 221 | 410/38 | 55/13 | 328/15 | 0/16 | | 383/29 | 4544 | 3286 | 4615 | 4736 | 4736 |
| shl1lsb | 326 | 446/144 | 76/99 | 328/43 | 0/35 | | 404/134 | 4544 | 4884 | 5254 | 5476 | 5024 |
| tspcr | 345 | 393/286 | 53/218 | 20/7 | 63/14 | | 136/239 | 3392 | 4930 | 4930 | 4930 | 4731 |
| Average | | | | | | | | | 0.925 | 1.093 | 1.109 | 1.000 |