

Fast Evaluation of Symmetric-Feasible Sequence-Pairs for Analog Topological Placement

Karthik Krishnamoorthy Sarat C. Maruvada Florin Balasa

Dept. of Computer Science, University of Illinois at Chicago, 851 S. Morgan St., Chicago, IL 60607, U.S.A.

Abstract – Using nonslicing topological representations in device-level analog placement has recently gained interest [1, 2, 8] since only feasible placements are analyzed, and since the number of devices in analog blocks is typically no more than 100 [3] and, therefore, no scalability problem may occur. In this context, Balasa and Lampaert introduced the concept of *symmetric-feasible* sequence-pair [1] in order to take into account during the exploration of the sequence-pair solution space the typical presence of symmetry groups of devices in analog designs. This paper presents a fast evaluation algorithm – of $O(n \log \log n)$ complexity – for *symmetric-feasible* sequence-pairs, superior to the algorithm in [1] having a quadratic complexity. Similar to Tang and Wong’s sequence-pair evaluation [9], this algorithm is based on the longest common subsequence computation, using an efficient *priority queue* data structure introduced by Johnson [4]. A prototype analog placement tool based on this algorithm has been implemented. The novel approach exhibits significantly better computation times than other traditional analog placement techniques.

1 Introduction

In CAD systems for analog layout, the traditional way of approaching device-level placement problems is to explore a huge search space using a combinatorial optimization technique (like, for instance, simulated annealing), the cell positions being specified in terms of absolute coordinates. This exploration strategy allowing illegal overlaps during the cell moves (translations, changes in orientation) is used by several software systems for analog layout like, for instance, KOAN/ANAGRAM II [3], PUPPY-A [6], and LAYLA [5]. Since this strategy often exhibits a slow convergence, the alternative strategy is to employ special nonslicing topological representations – like (symmetric-feasible) sequence-pairs [1], ordered (O)-trees [8], or binary (B*)-trees [2] – in order to explore only feasible placement solutions. While the placement techniques using the absolute representation trade-off a larger number of moves (in a combinatorial optimization framework) for easier and quicker-to-build layout configurations (not always physically realizable), the techniques adopting topological representations trade-off a smaller number of moves for more complex-to-build feasible layouts. Note that employing topological representations in device-level analog placement does not raise any scalability problem since analog circuits and the analog portions of mixed-signal systems-on-a-chip are significantly smaller than digital circuits – usually up to 100 devices in an analog block (and less than 20,000 devices in a complete subsystem) [3].

2 Symmetric-feasible sequence-pairs

In high performance analog circuits it is often required that groups of devices are placed symmetrically with respect to one or several axes. The main reason for symmetric placement and routing is to match the layout induced parasitics in the two halves of a group of devices. Failure to match these parasitics in, for instance, differential analog circuits can lead to higher offset voltages and degraded power-supply rejection ratio [3]. Placement symmetry can also be used to reduce the circuit sensitivity to thermal gradients. In order to combat potential mismatches, the thermally-sensitive device couples should be placed symmetrically relative to the thermally-radiating devices. This is why many analog designs contain along an asymmetric component an arbitrary number of symmetry groups of devices (that is, groups having distinct symmetry axes), each group containing an arbitrary number of pairs of symmetric devices, as well as self-symmetric devices – presenting a geometrical symmetry and sharing the same axis with its group.

Dealing efficiently with symmetry constraints in the framework of topological representations implies two difficult problems: a) how to recognize the encodings complying with the given symmetry constraints, without building the corresponding layout; and b) how to restrict the exploration of the solution space of the representation only to the subspace of those “symmetric-feasible” (S-F) codes. These problems have already been solved [1] for the sequence-pair representation [7].

Let (α, β) be the sequence-pair of a placement configuration containing a number of symmetry groups (each group composed of pairs of symmetric blocks, and self-symmetric blocks relative to a common vertical axis). Denoting by α_A^{-1} the position of block A in sequence α (as α can be viewed as a one-to-one mapping, α^{-1} is well-defined) and defining similarly β_A^{-1} , and also denoting by $sym(x)$ the block symmetric to x , the sequence-pair (α, β) is called *symmetric-feasible* (S-F) if for any distinct blocks x, y in any of the symmetry groups

$$(S) \quad \alpha_x^{-1} < \alpha_y^{-1} \iff \beta_{sym(y)}^{-1} < \beta_{sym(x)}^{-1}$$

Taking $y = sym(x)$, condition (S) shows that any symmetric pair of cells appears in the same order in both sequences α and β . In addition, two cells x, y belonging to distinct symmetric pairs and, respectively, their symmetric cells appear in reversed order in the two sequences of the encoding, as well as any two self-symmetric cells in the symmetry group.

Example: Fig. 1 displays a placement corresponding to the sequence-pair $(EBAFCDG, EBCDFAG)$. Assuming a symmetry group $(C, D), (B, G), A, F)$ composed of

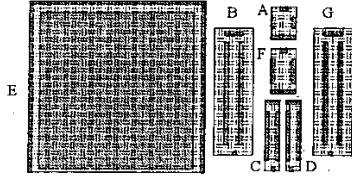


Figure 1: Placement with symmetry group ((C,D), (B,G), A, F) corresponding to the symmetric-feasible sequence-pair (EBAFCDG, EBCDFAG)

two symmetric pairs and two self-symmetric cells A and F, the sequence-pair above is symmetric-feasible. Taking, for instance, the self-symmetric cell A and comparing its positions $\alpha_A^{-1} = 3$ and $\beta_A^{-1} = 6$ in the sequences α and β to the corresponding positions of the other cells in the group, it follows that condition (S) is satisfied whenever cell A is involved. Similar comparisons involving the positions of the other cells in the symmetry group will conclude the verification.

This paper will present an evaluation algorithm of $O(n \log \log n)$ complexity for *symmetric-feasible* sequence-pairs, superior to the algorithm of quadratic complexity in [1]. The rest of the paper is organized as follows: Section 3 will present the novel evaluation algorithm for S-F sequence-pairs based on the longest common subsequence (LCS) approach [9]; an example will illustrate the algorithm flow. Afterwards, Section 4 will present an overview of the experimental results and Section 5 will summarize the conclusions of this research.

3 Fast evaluation of S-F sequence-pairs

Given a sequence-pair (α, β) , the maximal (i.e., which cannot be enlarged) subsequences common to α and β represent paths in the horizontal constraint graph of (α, β) . For instance, EBCDG, EBAG, and EBBG are common subsequences of the sequence-pair (EBAFCDG, EBCDFAG) (see Fig. 1). If the cells are weighted with their widths, the longest common subsequence (e.g., the total weight of EBCDG) represents the width of the block placement [9]. Similarly, the maximal subsequences common to α^R (sequence α reversed) and β represent paths in the vertical constraint graph of (α, β) – as, for instance, CFA (see Fig. 1). If the cells are weighted with their heights, the longest common subsequence represents the height of the whole placement. Based on these concepts, an evaluation algorithm of complexity $O(n \log \log n)$ building the block placement from a given sequence-pair was presented in [9].

This section will present an algorithm of the same complexity building a placement *subject to symmetry constraints* from a given *symmetric-feasible* sequence-pair. The algorithm described below is executed in each inner-loop iteration of the simulated annealing, evaluating the layout cost after each move. The analog devices to be placed on the chip area are represented by n rectangular blocks B_1, \dots, B_n , each block B_i having the width w_i and the height h_i , and having (x_i, y_i) as coordinates of its left-bottom corner. For sake of readability, the algorithm was simplified assuming that all the devices subject to symmetry

constraints belong to a single symmetry group. The implementation takes into consideration an arbitrary number of symmetry groups, though (see Section 4).

The subtle part of the algorithm is the use of a *priority queue* introduced by Johnson [4]. This data structure is implemented as a complete binary tree which leaves correspond to a doubly linked list of queue elements (see Fig. 2 (a-h)) called *buckets* [4]. Each bucket in the priority queue has associated two keys (*index*, *length*), where the *index* key is the cell position in sequence β and *length* is the length of the LCS until that cell in the sequence-pair. Since the number of buckets is at most $n+1$, the insertion and deletion operations take $O(\log \log n)$ time [4].

The property (S) of “symmetric-feasibility” ensures that the computation of the device abscissae x_j (initially, zero) can be done with three traversals of sequence α . The first traversal is similar to the computation of block abscissae in [9]:

```

insert bucket (0,0) in the priority queue;
for each index  $i$  in  $\alpha$  ( $i=1$  to  $n$ )
  let  $l$  be the index in  $\beta$  of cell  $B_{j=\alpha_i}$ ;
  find bucket (say,  $pred_l$ ) which index is
    the largest lesser than  $l$ ;
   $x_j = \text{length of } pred_l$ ;
  insert bucket ( $l, x_j + w_j$ ) into the queue;
  remove buckets with greater index than  $l$ 
    and lesser length than  $x_j + w_j$ ;
end.for
 $x_{symAxis} = \max \{ \frac{x_j + (x_k + width_k)}{2} \}, \forall \text{ sym. pairs } (B_j, B_k)$ ;
remove all buckets from priority queue;

```

The second traversal is similar to the first, but before inserting the bucket $(l, x_j + w_j)$ into the queue, if (B_j, B_k) is a symmetric pair and $d = x_{symAxis} - \frac{x_j + (x_k + width_k)}{2} > 0$, then x_j is increased by d . The width of the chip W is the *length* of the bucket $pred_n$. At the end of this traversal, some of the devices may have been pushed further to the right. The third traversal of α will fix all the symmetry constraints:

```

insert bucket ( $n+1, W$ ) in the queue;
for each index  $i$  in  $\alpha$  ( $i=n$  to  $1$ )
  let  $l$  be the index in  $\beta$  of cell  $B_{j=\alpha_i}$ ;
  find bucket (say,  $succ_l$ ) which index is
    the smallest greater than  $l$ ;
   $x_j = \text{length of } succ_l$ ;
  if  $(B_j, B_k)$  is a symmetric pair and
     $d = x_j + (x_k + width_k) - 2x_{symAxis} > 0$ 
  then  $x_j = x_j - d$ ;
  insert bucket ( $l, x_j + w_j$ ) into the queue;
  remove buckets with lesser index than  $l$ 
    and greater length than  $x_j + w_j$ ;
end.for
remove all buckets from priority queue;

```

The algorithm deriving the y -coordinates of the devices performs the LCS computation with the sequence α in reverse order. In the absence of symmetry constraints one traversal of the sequence-pair is sufficient; however, a second traversal will be necessary in order to satisfy the symmetry constraints of the type $y_j = y_k$ for two symmetric devices (B_j, B_k) .

```

insert bucket (0,0) in the priority queue;
for each index  $i$  in  $\alpha$  ( $i=n$  to  $1$ )

```

let l be the index in β of cell $B_{j=\alpha_i}$;
 find bucket (say, $pred_l$) which index is
 the largest lesser than l ;
 $y_j = \max\{y_j, \text{length of } pred_l\}$;
 if B_j has a symmetric B_k then $y_k = y_j$;
 insert bucket $(l, y_j + h_j)$ into the queue;
 remove buckets with greater index than l
 and lesser length than $y_j + h_j$;

endfor

remove all buckets from priority queue;

Example: Consider the symmetric-feasible sequence-pair (BFCDFGEA, ABCDFGE) relative to a symmetry group consisting of two pairs of symmetric devices (B,E) and (C,D), and a self-symmetric device A. The seven cells have the widths and heights as indicated: A(8x2), B(2x4), C(1x2), D(1x2), E(2x4), F(2x2), and G(2x3). The successive modifications of the priority queue during the first traversal of the sequence-pair are displayed in Fig. 2(a-h). The current width of the placement (Fig. 2(i)) is 8, the length field in the last bucket in Fig. 2(h). In the second traversal, the rightmost devices in the symmetry pairs are positioned attempting to meet the x -symmetry constraints ($x_i + w_i$) + $x_j = 2x_{symAxis}$. However, some symmetry constraints will still not be satisfied: for instance, the cells (C,D) are not symmetric yet since $((x_C + w_C) + x_D = 7) < (2x_{symAxis} = 8)$ (see Fig. 2(j)). In the subsequent traversal, cell B is pushed further to the left (see Fig. 2(k)) and all the x -positioning and symmetry constraints will be satisfied. The y -coordinates of the devices in Fig. 2(i-k) are arbitrary since they were not computed yet. They are subsequently determined using the LCS approach, with only one traversal for this illustrative example. The final placement is shown in Fig. 2(k).

The priority queue has at most $n + 1$ buckets. The insert and remove operations in this data structure can be performed in $O(\log \log n)$ time [4]. Although in some iterations $O(n)$ buckets may be discarded, the amortized complexity per traversal is $O(n \log \log n)$. Since the number of traversals is at most 5 due to the *symmetric-feasibility* property of the sequence-pair [1] (see Section 2), the overall complexity is $O(n \log \log n)$.

4 Experimental results

A placement tool for analog layout using selectable exploration algorithms has been implemented in C++ on a SUN Blade 100 workstation. This prototype tool can operate both with different topological representations (sequence-pairs, O-trees, and B*-trees) and different code evaluation algorithms. In addition, a complementary placement algorithm based on the traditional absolute representation has been embedded in the tool as well. The tool uses a simulated annealing optimizer with a complex cost function comprising, along with the chip area and estimated wire length, different penalty terms like, e.g., modeling device separation constraints. Besides symmetry constraints, the tool handles systematically-induced device mismatches, alignment constraints, and also performs shape optimizations. Table 1 displays part of the results of our experiments. The test benchmarks are analog blocks components of a spread spectrum transceiver. Figure 3 displays the placement

Design	Nr. cell	Sym. grp.	Abs. repr.	Seq. pair[1]	Crt. alg.
			Time/Area	Time/Area	Time/Area
lpf2_b25b	52	1	19.5 / 45.7	5.2 / 38.0	1.7 / 37.4
biasynth_2p4g	65	3	29.3 / 6.8	12.6 / 5.4	2.5 / 5.2
dcervo_cmfb	66	2	24.8 / 72.0	16.9 / 61.2	4.8 / 59.4
modbias_2p4g	87	5	45.0 / 65.9	28.8 / 56.9	7.8 / 55.8
div_by_2or4	116	5	61.7 / 59.6	41.2 / 51.4	11.4 / 50.6

Table 1: Placement results (Time [min], Area [$10^3 \times \mu m^2$])

solution for the first benchmark in Table 1.

The experiments show that using topological representations is significantly better both in terms of computational effort and placement quality than using the more traditional absolute representation [3, 5, 6]. The running times obtained when using our novel approach are much better than the computation times obtained when using an exploration based on sequence-pairs [7] with the an algorithm of quadratic complexity for the evaluation of the symmetric-feasible sequence-pairs [1].

5 Conclusions

This paper has presented a novel analog topological placement algorithm operating with symmetric-feasible sequence-pairs. The algorithm employs the *longest common subsequence* approach, its efficiency being due to the use of a special data structure – the Johnson's priority queue.

References

- [1] F. Balasa, K. Lampaert, "Symmetry within the sequence-pair representation in the context of placement for analog design," *IEEE Trans. CAD*, vol. 17, no. 7, pp. 721-731, July, 2000.
- [2] F. Balasa, S.C. Maruvada, K. Krishnamoorthy, "Efficient solution space exploration based on segment trees in analog placement with symmetry constraints," *Proc. IEEE/ACM Int. Conf. on Comp.-Aided Design*, pp. 497-502, San Jose CA, Nov. 2002.
- [3] J. Cohn, D. Garrod, R. Rutenbar, L. Carley, *Analog Device-Level Automation*, Kluwer Acad. Publ., 1994.
- [4] D.B. Johnson, "A priority queue in which initialization and queue operation take $O(\log \log D)$ time," *Math System Theory*, vol. 15, no. 4, pp. 295-309, Dec. 1982.
- [5] K. Lampaert, G. Gielen, W. Sansen, "A performance-driven placement tool for analog integrated circuits," *IEEE J. Solid-State Circ.*, SC-30, no. 7, pp. 773-780, 1995.
- [6] E. Malavasi, E. Charbon, E. Felt, A. Sangiovanni-Vincentelli, "Automation of IC layout with analog constraints," *IEEE Trans. CAD*, vol. 15, no. 8, pp. 923-942, Aug. 1996.
- [7] H. Murata, K. Fujiyoshi, S. Nakatake, Y. Kajitani, "VLSI module placement based on rectangle-packing by the sequence-pair," *IEEE Trans. CAD*, vol. 15, no. 12, pp. 1518-1524, Dec. 1996.
- [8] Y. Pang, F. Balasa, K. Lampaert, C.-K. Cheng, "Block placement with symmetry constraints based on the O-tree non-slicing representation," *Proc. 37th Design Aut. Conf.*, pp. 464-467, 2000.
- [9] X. Tang, D.F. Wong, "FAST-SP: A fast algorithm for block placement based on sequence pair," *Proc. Asia-S. Pacific Design Automation Conf.*, pp. 521-526, Yokohama, Japan, Jan. 2001.

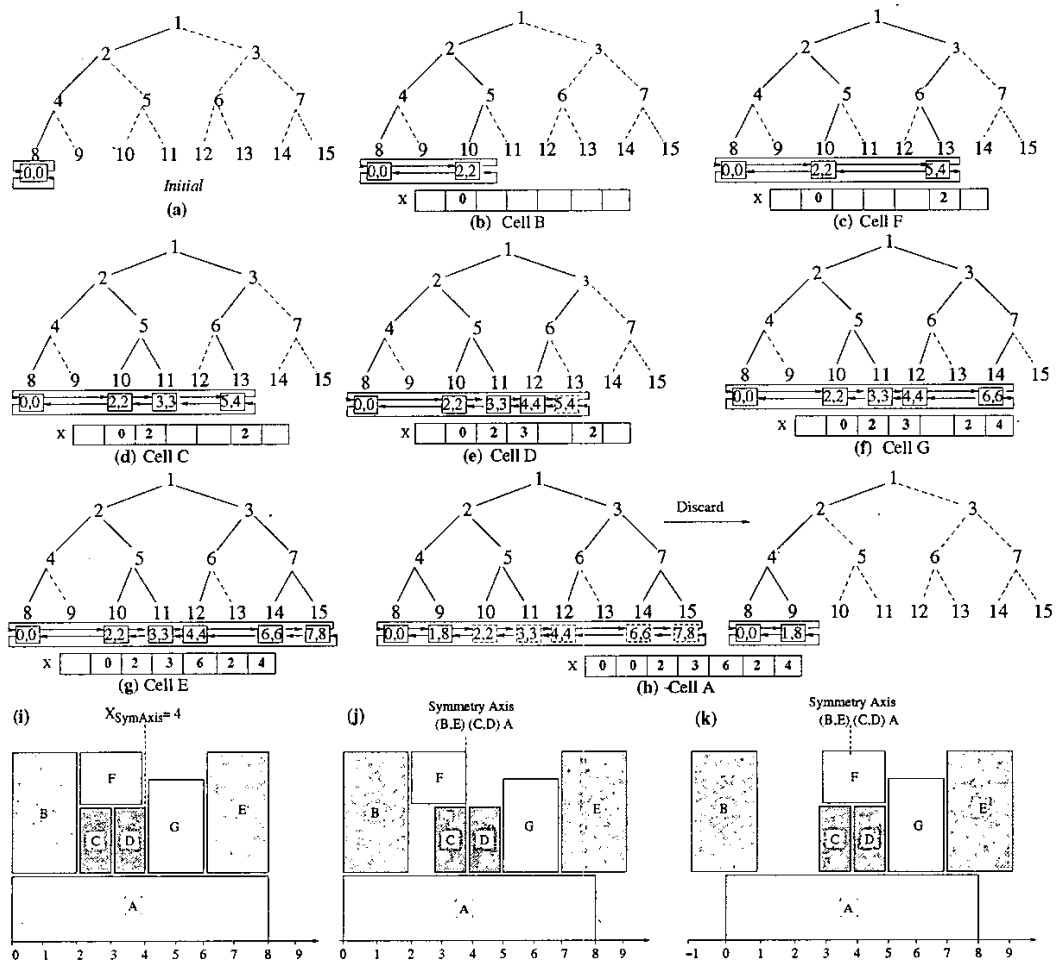


Figure 2: Computation of the device abscissae: (a-h) the evolution of the priority queue during the first traversal of the S-F sequence-pair $(BFCDGEA, ABCDFGE)$, and (i-k) the placement after each traversal

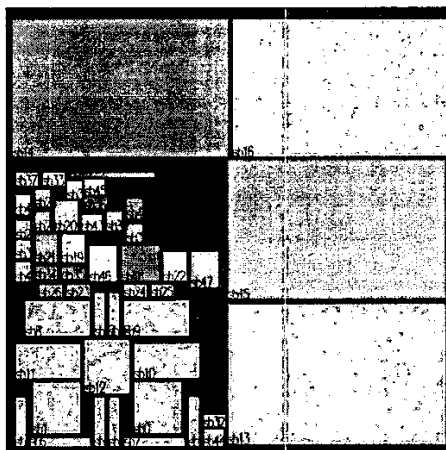


Figure 3: Placement of an analog block with symmetry group