

The Effect of Placement on Yield for Standard Cell Designs *

Rajnish K. Prasad and Israel Koren

Department of Electrical and Computer Engineering
University of Massachusetts, Amherst, MA 01003, USA

Abstract

The ability to improve the yield of integrated circuits through layout modification has been recognized and several techniques for yield enhanced routing and compaction have been developed. Yield improvement during routing is however, limited by the predetermined placement. It is conceivable therefore, that different placements of the modules (e.g., standard or custom cells) may lead to very different yield enhanced routings with different projected yields. This is conceptually similar to the effect that the floorplanning of the entire chip has on the yield [2], but while chip floorplanning deals with the major building blocks, placement deals with the modules within an individual block. Yield enhanced placement of modules has not been attempted before mainly due to the difficulty of estimating the yield of the block before the routing is done. Recently, a technique for estimating the yield prior to the routing has been developed [1] making it possible to modify the placement in order to achieve higher yield. The goals of this paper are to investigate the effect that placement has on the projected yield and to modify a standard cell placement algorithm so that yield becomes a design objective.

1: Introduction

The general placement problem is the problem of placing a set of circuit modules within a block such that a certain objective function is minimized. The ultimate goal is to minimize the total chip area occupied by the circuit modules and minimize the length of the interconnections between the modules. To make the placement problem computationally feasible, various simpler to calculate objective functions such as the area of the bounding rectangles, total interconnection wire length, or some other routing area estimates are commonly used. The yield of the circuit is normally not considered during placement.

Recently, it has been shown [2] that floorplanning may considerably affect the yield of the chip. We believe that the placement of modules within a block will have a similar impact. Yield enhancement has so far been attempted only during the detailed routing and compaction steps (e.g., [4, 5, 6, 7]).

*Supported in part by NSF under contract MIP-9710130.

However, significant changes in wiring congestion cannot be performed during these steps as the circuits have already been placed. Since there is a direct relationship between the density of the routing and the yield, it is conceivable that by incorporating the expected yield into the objective function of the placement algorithm, improvements in yield can be achieved. This has not been attempted before, since until now the expected yield was calculated only after the layout (including routing) was completed.

It has recently been demonstrated in [1] that reasonably accurate estimates for the yield can be obtained prior to routing. Thus, we can use such estimates within the placement stage to enhance the yield of the final layout. In section 2 we describe the modified placement algorithm which incorporates the yield as a design objective. In section 3 we present some of our numerical results. Section 4 presents conclusions and future work.

2: The Modified Placement Algorithm

The placement problem can be classified according to the different types of design methodologies such as gate array, standard cell and macro/custom cell placement. We focus in this paper on the placement of standard cells with yield as a design objective and we use the standard cell placement algorithm TimberWolf [3] to illustrate our approach. This algorithm employs simulated annealing for minimizing the total wire length. The simulated annealing procedure randomizes the iterative improvement technique and also allows occasional "uphill moves" in an attempt to reduce the probability of getting stuck at a local optimal solution. These uphill moves are controlled probabilistically by the temperature T , and become less and less likely toward the end of the process, as the value of T decreases. TimberWolf allows placements with overlapping modules as intermediate solutions, to achieve fast update of the cost function. After each move to a neighboring solution, the overhead in displacing modules to remove overlap is not incurred. TimberWolf also allows modules to move to a new location without any swapping or width requirement, increasing this way the number of different placements examined. The cost function in TimberWolf consists of total wire length and a measure of the overlap between modules. The cost due to module overlap converges to zero, as the temperature T approaches zero guaranteeing in this way a feasible final placement.

We have modified the TimberWolf placement algorithm to include yield as a design objective. The pseudo-code for the modified simulated annealing placement algorithm is shown below.

```
SimulatedAnnealing(x, T) {
/* Given an initial solution x and initial parameter T */
while("stopping criterion" is not satisfied ){
    generate T' < T;
    T = T';
    while("inner loop criterion" is not satisfied){
```

```

        generate a new solution x';
        /* estimate yield of new solution */
        YieldEstimate(x');
        /* c(x') is cost of the new solution */
        /* c(x) is cost of the current solution */
        if(accept(c(x'),c(x)){
            x = x';
        }
    }
}
}

```

To incorporate the yield objective into the cost function, we have to estimate the yield of intermediate placements. The routine “YieldEstimate” (shown below), estimates the yield of the new placement using *ybound* [1]. As reported in [1], fairly accurate yield estimates (with differences of 1.0 to 4.0% for short-circuit failures and 0.4 to 4.0% for open-circuit failures) can be obtained by the *ybound* algorithm in a fraction of the time required for actual yield estimation. This algorithm uses an approximation of the average length for the conductors in each wiring channel for estimating the short-circuit yield. If the current intermediate placement has overlapping cells, the overlap is removed temporarily before the yield is estimated. This is essential, since substantial overlap between adjacent modules will cause false net segment overlap, which in turn will result in a large number of track requirements and consequently, wrong yield estimation. Once overlaps are removed, a minimum spanning tree is constructed for each net. Then the left edge algorithm is used to assign tracks to all net segments. Finally, the channel information is decompiled for yield estimation. The pseudo-code for yield estimation is

```

YieldEstimate(original) {
    /* original placement has module overlaps */
    /* make a copy of it before modifying it */
    newCopy = copyCurrentState(original);

    /* remove overlap from placement */
    removeOverlap(newCopy);

    /* build a Minimum Spanning Tree for each net */
    /* assign tracks using the Left Edge algorithm */
    globroute();
    /* obtain yield estimate using ybound */
    yield = ybound();
}

```

The routine “accept” in the SimulatedAnnealing procedure, takes in the new cost $c(x')$ and current cost $c(x)$ and decides if the new solution should be accepted or rejected. The new solution is definitely accepted if the new cost is better than the current one, and is accepted with a probability determined by the annealing schedule if it is worse than the current one.

The new cost function for the modified algorithm is

$$Cost(x) = WireLength(x) + Overlap(x) - Yield(x) * ScaleFactor(x) * \beta \quad (1)$$

where x denotes the index of the current iteration of the simulated annealing process. The parameters $ScaleFactor(x)$ and β are explained below. Since the wire length and overlap costs are large integers and the yield is a fraction less than one, we introduced a scaling factor function so that changes in yield are not ignored completely. The scale factor is determined dynamically for each iteration, and is computed as shown below. We first define

$$Scale(x) = |(WireLength(x) - WireLength(x - 1)) / (Yield(x) - Yield(x - 1))| \quad (2)$$

where $x - 1$ denotes the index of the previous iteration. We then compute $ScaleFactor(x)$ as

$$ScaleFactor(x) = ScaleFactor(x - 1) + Scale(x - 1) / h - Scale(x - 1 - h) / h \quad (3)$$

where h is the depth of history for variation in wire length with respect to yield. This $ScaleFactor(x)$ captures the information about the average variation in wire length with respect to the change in yield in the last few iterations. From the experiments we carried out, we found that a depth of history equal to 3 worked well. The parameter β serves to assign a weight to the yield relative to the wire length and can be any real number greater than zero. As will become evident in the next section, the placement algorithm should be run for several values of β , and then a placement with acceptable wire length and yield should be selected.

3: Numerical Results

Ten benchmark circuits were selected from the iscas and lgsynth91 test suites. Table 1 shows the variation in yield for different placements of the ten circuits. It shows the possible range of yield for the different designs, when starting with any possible initial placement. A key observation is that for larger circuits (e.g., C5315 and C6288), the effect of placement on the yield is larger than for smaller circuits (e.g., C432, C499 and C1355). This is mainly due to the fact that a larger number of placements can be generated for a bigger circuit than for smaller circuits.

In practice however, such a choice of starting with any random initial placement (as shown in Table 1) is not available and various techniques to obtain an initial placement are used. Placement algorithms like those based on simulated annealing normally use an especially generated initial solution to obtain a near optimal placement. Table 2 compares the yield achieved by incorporating

Design	Number of Shapes	Min Yield		Max Yield		Yield Range%
		Wire Length	Yield	Wire Length	Yield	
C432	8573	95881	0.917992	88111	0.943516	2.6%
C499	9114	91797	0.921549	88554	0.945984	2.4%
C1355	9481	108052	0.926048	104271	0.946846	2.1%
C880	11058	129326	0.887666	120814	0.920834	3.3%
C1908	11232	129407	0.903055	117042	0.932235	2.9%
C2670	30602	445010	0.598460	432647	0.652999	5.5%
dalu	32313	371932	0.636525	345657	0.701436	6.4%
i8	33654	480713	0.536601	467960	0.585129	4.8%
C5315	69241	846123	0.377067	787151	0.463422	8.6%
C6288	123884	908425	0.335353	800092	0.428692	9.4%

Table 1. Range of Yield and Wire Length for the ten benchmarks

the yield objective into the cost function, to the yield of the placement generated by the original algorithm without a yield objective. Both final placements were obtained from the same initial placement. The wire length and yield of the placement generated when incorporating the yield objective into the cost function are relative to those obtained without considering yield. As all the solutions with a yield objective have a wire length which is very close to the near optimal wire length of the original solution, any of them is equally probable to be accepted and if the yield of the circuit is not considered, the algorithm may select a placement with a yield lower than that achievable. For the largest circuit (C6288), the gain in yield is 4.7%. In several cases, yield enhancement occurred along with a reduction in total wire length. For example, for C6288 the reduction in wire length is 3.8%. This reduction is due to the random nature of the simulated annealing process and can not be guaranteed. In such cases, we pay no penalty for increasing the yield. We do have to expect in some cases a possible increase in wire length for a placement with a higher yield.

Design	Without Yield Objective		With Yield Objective	
	Wire Length	Yield	Wire Length	Yield
C432	87229	0.937738	0.991	1.006
C499	88615	0.936475	0.982	1.006
C1355	106150	0.935358	1.003	1.006
C880	123322	0.910646	0.967	1.004
C1908	120123	0.915330	1.013	1.006
C2670	435714	0.621824	1.0002	1.036
dalu	351890	0.679434	0.998	1.017
i8	487279	0.555882	0.995	1.025
C5315	756475	0.449924	0.994	1.026
C6288	843885	0.368968	0.962	1.047

Table 2. The changes in Yield and Wire Length with the new cost function

Figure 1 shows the different solutions obtained for the i8 benchmark by varying the parameter β . The baseline solution is for $\beta = 0$ (i.e., the original placement algorithm with no yield con-

siderations). The yield and wire length of all the other solutions are normalized with respect to the baseline solution. As β increases, we assign a larger weight to the yield compared to the wire length. As seen in the figure, giving more weight to yield does not guarantee higher yield or vice versa. Thus, several runs with varying β are required to obtain a solution with acceptable wire length and yield. Also, as expected, some designs (like i8) give better improvement in yield for very little or no wire length increase whereas others (like C880 shown in Figure 2) do not provide such gain. Figures 1 and 2 demonstrate that the wire length and yield attributes are not correlated in the general case. As a result, we may be able to reduce both in some cases while we may have to trade-off one for the other in other cases.

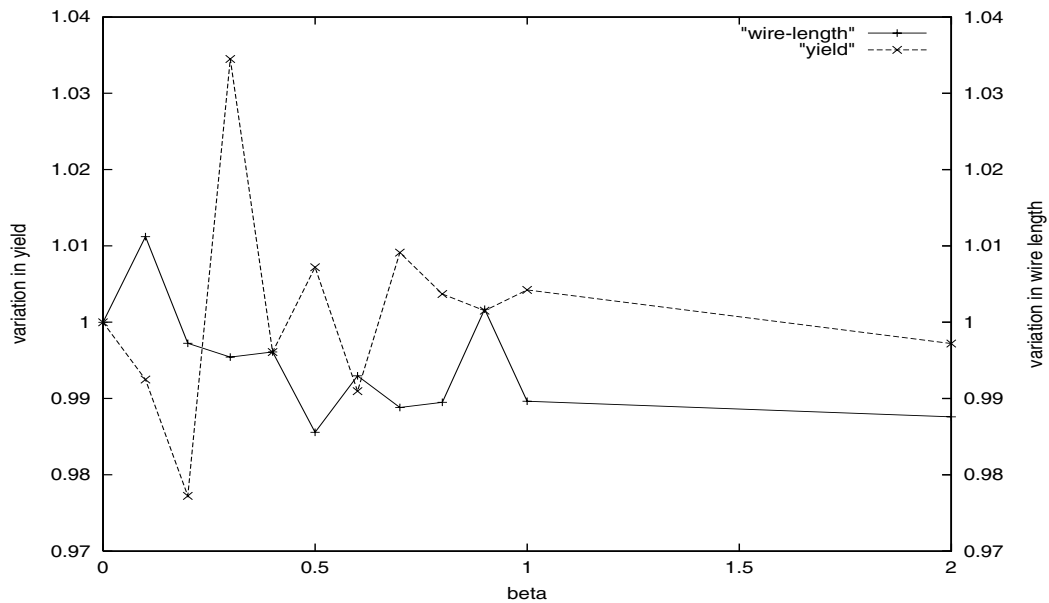


Figure 1. Variation in wire length and yield as a function of β for i8 benchmark.

Curves like those in Figure 1 and 2 do not convey in a clear way the possibility of a trade-off between yield and wire length. A figure showing the Pareto optimal solutions would better assist the designer when deciding on the final placement. Such a set of Pareto optimal solutions is shown in Figure 3 for the C2670 benchmark. This figure shows that the yield of this circuit can be increased by 3.4% for just 0.8% increase in wire length.

Figure 4 shows the estimated yield as a function of the defect density for two placements of the C5315 circuit, one obtained with the original cost function and the other obtained with the modified cost function. As seen, we get an approximately 3 to 4% improvement in yield when the defect density increases.

To illustrate the differences between placements obtained by the original and modified placement

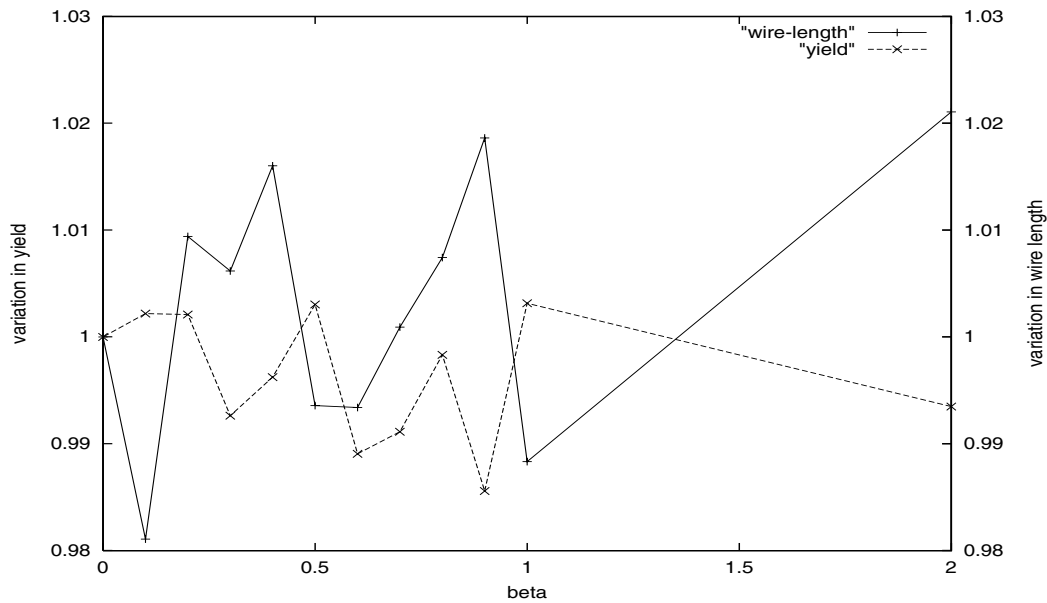


Figure 2. Variation in wire length and yield as a function of β for C880 benchmark.

algorithms we have applied the two algorithms to a simple nine standard cell design. Figure 5 shows the two placements where the placement with the higher yield has a more uniform distribution of net segments across the channels. For such a very small circuit the difference in yield is negligible (0.000380) but this difference will become noticeable when a logic block which is roughly 25 times larger is designed.

4: Conclusions and Future Work

The numerical results presented in the previous section reinforced our belief that the yield of circuits can be enhanced by incorporating the yield objective into the cost function of a placement algorithm. Also, as illustrated by the results, greater yield enhancement can be achieved for bigger circuits at lesser or no penalty in terms of wire length. We expect to be able to obtain better results by designing a new simulated annealing algorithm with all its parameters (like range of temperature and the rate of temperature reduction) fine-tuned for yield maximization. We also expect (based on the results in Table 1) that incorporating the yield objective in the generation of an initial placement would provide better opportunity for a yield enhanced layout. In addition to the above, we would also like to extend this approach to the gate array and macro/custom cell placement procedures.

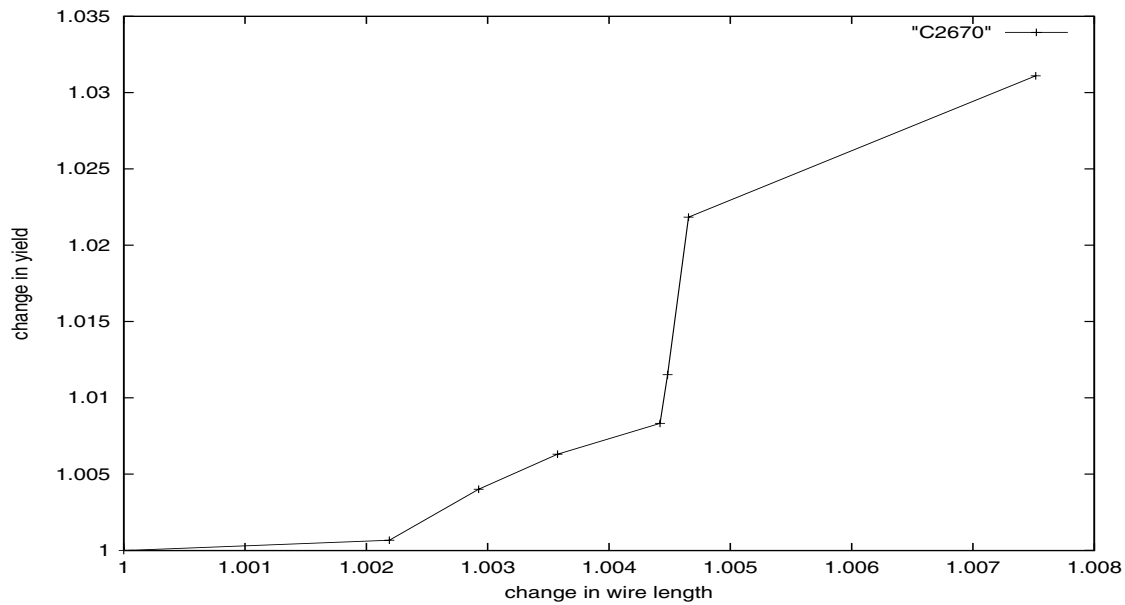


Figure 3. Pareto Optimal solutions for the C2670 benchmark.

References

- [1] A. Venkataraman and I. Koren, "Determination of yield bounds prior to routing," *Proceedings of the IEEE Symposium on Defect and Fault Tolerance in VLSI Systems*, pp 4-13, 1999.
- [2] I. Koren and Z. Koren, "On the Effect of Floorplanning on the Yield of Large Area Integrated Circuits," *IEEE Trans. on VLSI Systems*, pp. 3-14, March 1997.
- [3] University of California, "The TimberWolf Placement and Routing Package," 1984.
- [4] E.P. Huijbregts, H. Xue and J.A.G. Jess, "Routing for Reliable Manufacturing," *IEEE Trans. on Semiconductor Manufacturing*, vol. 8, pp. 188-194, May 1995.
- [5] S. Y. Kuo, "YOR: A Yield-Optimizing Routing Algorithm by Minimizing Critical Areas and vias," *IEEE Trans. Computer-Aided Design*, vol. 12, no. 9, pp. 1303-1311, Sept. 1993.
- [6] A. Venkataraman, H. Chen and I. Koren, "Yield Enhanced Routing for High-Performance VLSI Designs," *Proc. of the Microelectronics Manufacturing Yield, Reliability and Failure Analysis, SPIE'97*, pp. 50-60, Oct. 1997.
- [7] V.K.R. Chiluvuri, I. Koren and J. L. Burns, "The Effect of Wire Length Minimization on Yield," *Proc. of the 1994 IEEE Internl. Workshop on Defect and Fault Tolerance in VLSI Systems*, pp. 97-105, Oct. 1994.
- [8] S.M. Sait and H. Youssef, "VLSI Physical Design Automation: Theory and Practice," *World Scientific*, 1999
- [9] T. Lengauer, "Combinatorial Algorithms for IC Layout," *Wiley-Teubner Series*, 1990

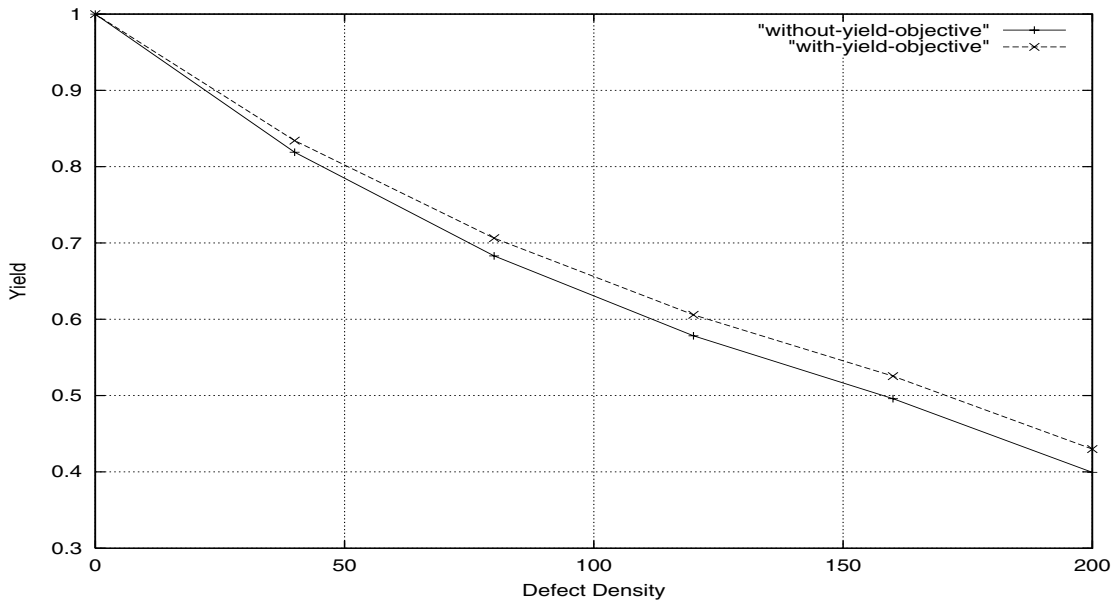


Figure 4. Yield as a function of the defect density for the C5315 benchmark.

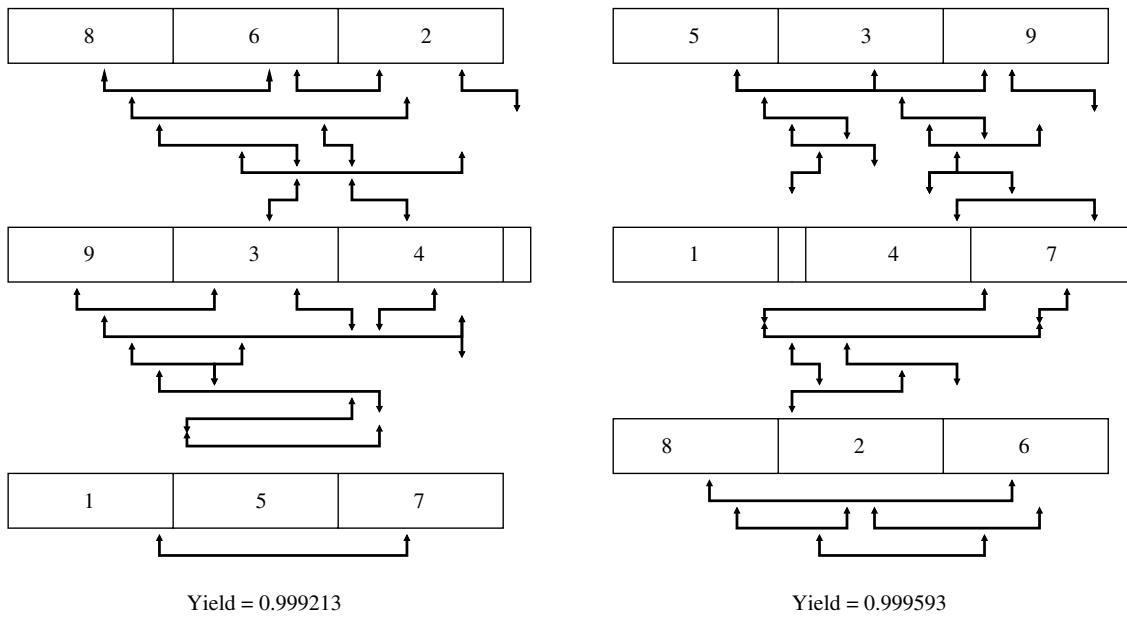


Figure 5. Two placements of a nine standard cell design.