A Method for the Evolution of Transistor Placement in CMOS Cells

Jesse Craig
B-14 Stonehedge Dr.
South Burlington, VT 05403
jecraig@yahoo.com

Steven G. Lovejoy
47 Butler Drive
South Burlington, VT 05403
stevelj@rocketmail.com

Abstract

This paper considers a method for evolving an ideal placement of transistors in a CMOS cell using genetic algorithm techniques. The problem of transistor placement is an optimization problem constrained by the wireability of the design, the total wire length needed to fulfill all the interconnections, and the total size taken by the transistors. The method is ignorant of the actual function of the cell, whether it be an AND, OR, MUX, or ADDER. The method takes as its input the schematic of a functional cell detailing the transistors needed, a combination of PFETs and NFETs, and the interconnections between these transistors. The method outputs an ordering of the transistors which can be fed to a routing tool for wiring and finally yield a valid recipe for building the cell in silicon. This method, applied to several standard cells, has generated a near optimal solution when compared to manually placed versions under the same constraints.

Keywords: CMOS transistor placement, genetic algorithm, row-based transistor placement

1 Introduction

The VLSI industry is fast evolving new technologies to increase performance and decrease cost. With each new technology comes constraints on how transistors must interact. These constraints cause nearly every standard cell in the library to be redesigned. As the speed in which new technologies are developed increases, and the time-to-market demands of customers grow, automated techniques for redesigning the libraries becomes a necessity. An optimal standard cell design will reduce the total size of the cell and the total wire length needed to wire interconnections. The smaller a cell is, the smaller the resulting chip will be, and the larger profit margin the microelectronics fabricator will receive. The lower the total wire length the higher performance the cell can attain. This method's fitness function takes these goals into account when deciding the optimal placement .

Common design practice divides a standard cell into many P and N wells, or channels, which can respectively hold PFETs and NFETs. This style is called row-based transistor placement. When two transistors in a well are interconnected and adjacent they can be overlapped in silicon, reducing the size of that cell. Otherwise a wire must be run which reduces performance and can cause unwireable conditions if too much wire congestion occurs. Conforming to Schemata theory (Holland, 1992) the building blocks of this method are transistors which are all interconnected to each other with few connections to any other transistors. Transistors which have the ability to overlap and reduce the number of gaps would also be building blocks because of the higher precedence given to gap reduction. The example in section 3 illustrates these building blocks.

Under the constraints of cell size and wire length the total number of possible solutions is an incredibly large number, making a random walk method impractical. The method described in this paper touches on roughly 0.005% of the possible solution space, as defined below. By exploring so little of the possible solutions the algorithm remains fast while still producing a near optimal solution.

Let n be the number of transistor wells

Let w be the maximum transistors per well

Let s be the size of the problem space

$$s = \left(2^{w}\left(w!\right)\right)^{n}$$

2 Internal Workings

This method, like most genetic algorithms, is made up of a selection operator, crossover operator, and mutation operator, which work on an internal representation of the solution called the genotype. Selection insures only the fittest survive while crossover and mutation introduce new solutions. One run of each of the operators on the population is referred to as a generation. The population for each generation is initialized to be the children of the prior generation’s operators. The population is said to have chosen a best solution, or converged, when the best individual represents the maximum predicted stability of the population. This number is the average number of individuals in a heterogeneous population that can survive mutation and crossover.

Let p be the population size

Let Pc be the probability of crossover

Let Pm be the probability of mutation

Let t be the maximum prediced stability

$$t = \left[1 - \left(Pc + Pm\right)\right]p$$

This act of convergence is driven by the selection operator. The population will never converge to complete uniformity because the mutation and crossover operators are always introducing new individuals into the population.

An implementation of the method was developed for experimentation in C++. To support this effort several of IBM’s internal Electronic Design Automation (EDA) tools were used. These included tools for routing the wires, viewing schematics, and viewing the final transistors after they were wired and placed. Perl scripts were used for collecting the data and filtering it into a usable form.

2.1 Fitness function and Constraints

For each individual in the population, the fitness is calculated. The best individual is determined by the lowest fitness function value. There are three pieces of the fitness function: area, gate alignment, and total wire length. The area of the cell is critical so that more cells can be placed on a chip. The gate alignment is a simple wireability check. If NFETs and PFETs with common gate connections are aligned, then the wiring for the FETs is a simple wire. If the FETs are misaligned, then there are extra jogs or layer changes to connect the FET's gate. The total wire length is a measurement to create the cell with the least amount of wiring. In general, the simpler the cell wiring, the smaller the cell design becomes.

To get a minimized area, the number of spaces between devices must be minimized. If two adjacent devices are connected to different diffusion nodes, then the manufacturing rules require a space between these two devices. For every time a common diffusion node is found between two adjacent devices, the space between the two devices is removed and they are overlapped to ensure connectivity. If the two nodes are connected nowhere else, then the space required for contacts is also removed, forming a very tight layout. The fitness function calculates the number of gaps in the individual's device ordering.

For maximizing the gate alignment, the number of gate misalignments is calculated. To start, the misalignment cost is equal to the maximum number of NFETs or PFETs. Then for each pair of aligned gates, the misalignment cost is decremented. The objective is to get all gates aligned and therefore yield a misalignment cost of 0.

The total wire length calculation is similar to the calculation documented in S. Saika's paper (Saika, et al. 1997). For each net, the connections to the FETs are searched in the horizontal direction looking for the first and last connection. The wire length of the nets is the difference between the last and first connection. The vertical wire length was not calculated because it is the same for all nets that cross between the NFETs and PFETs. The schematic connections determine the number of nets that cross from the N well and P well and are not changed by the transistor placement. Since our placement is an ordering only, the distances are in grid units and not very accurate.

After the fitness components are calculated, they are added together in a weighted sum.

Let G be the number of gaps
Let M be the number of misaligned FETs
Let L be the wire length
Let F be the fitness function
Let $W1,\ W2,\ W3$ be constant weights

$$F = G * W1 + M * W2 + L * W3$$

The order of importance of the fitness components is, reducing gaps, followed by misaligned FETs, and then wire length. Our values for W1, W2, and W3 are 10000, 1000, and 1 respectively.

2.2 Genotype

The individual solutions are represented internally as a set of arrays, one per transistor row. The size of these arrays is taken to be the greatest of the number of PFETs and NFETs. Each element of an array is filled with a signed random value. The ordering of these values decides the placement of each transistor inside the solution's transistor row. This is based on the "Random Key" method defined by Jim Bean (Bean, 1992). Simply stated, if the absolute value of the first element in the array is the sixth largest absolute value in that array then transistor number six will occupy the first position in the transistor well. The sign of the value decides the orientation of the transistor, a positive value meaning normal orientation, and a negative meaning the transistor has been flipped along the axis perpendicular to the transistor row. This flipping is needed to reduce the number of gaps between the transistors as described above. The random value method allows crossover to occur without introducing duplicates of one transistor in the same well.

Previous methods for solving this problem used a tree representation of the genotype (Schnecke, et al., 1997). The representation takes advantage of our proposed building block structure by allowing groups of cells to be moved together during crossover. One difficulty with a tree structure is that crossover can introduce duplicate transistors into the same well which later have to be removed during a post-crossover phase in each generation.

2.3 Selection

The selection operator uses tournament selection in which the best individual, as defined by the fitness function, is chosen from a pool of randomly chosen individuals in the current population. The size of this pool is a parameter to this operator which allows the selection pressure to be controlled. When a problem nears convergence the fitness values of the competing individuals are often close together. Using other forms of selection, such as proportional selection, causes the population to converge significantly slower. With tournament selection the difference between competing fitnesses is not a concern, only that one fitness is better than the others. The selection pressure for the best individual can be described using the following equation.

Let p be the population size
Let s be the selection pool size
Let $m(i, t)$ be the number of individuals i at time t

$$m(i, t+1) = s(m(i, t))$$

Where $m(i, t)$ has an upper - bound of p

2.4 Crossover

Crossover is a means of combining the traits, or building blocks, from two individuals in hopes that the best traits from each will combine to form superior offspring. This method's crossover uses selection to choose two individuals which will be the parents. Next a random location inside the transistor well arrays of one parent is chosen as the cross-point. For each array in that parent the contents from the beginning of that array up to the cross-point are swapped with the contents of the other parent's corresponding array. This destroys the parents and leaves two children individuals with traits from each parent. With building blocks often being transistors that are adjacent to each other this form of crossover works well as it tries to keep these relationships.

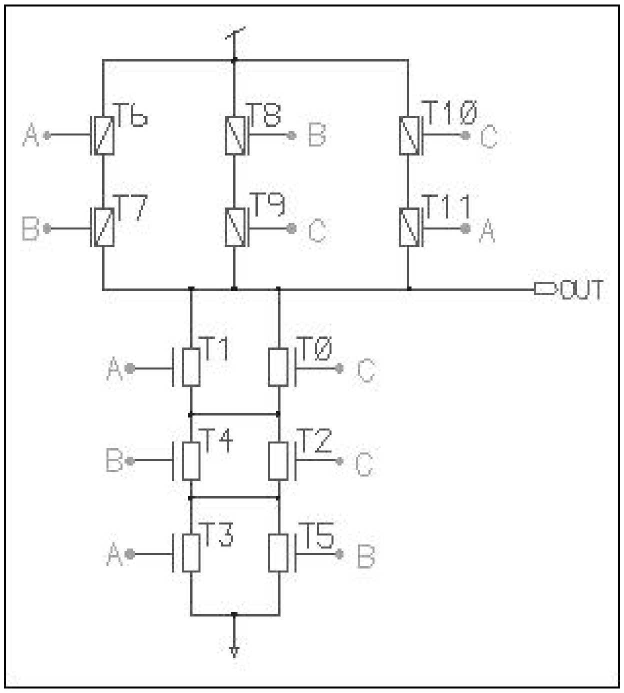
2.5 Mutation

The representation and crossover this method uses prevents any trait from being permanently removed from the population. This makes it so mutation is not required. However mutation is an excellent way to introduce noise into the system and allow greater amounts of the search space to be visited. Two mutation operators were used in this method. The first took a randomly chosen value in each of an individuals transistor well arrays and replaced it with a new random value. If this new value caused a different sort order in the array then nearly every transistor would be shifted to a new position. If the new value was close enough to the old value that the sort order remained the same no mutation would take place. This mutation operator causes a good deal of noise to be introduced. The second operator was designed to cause more, smaller changes. It chooses two values at random in each transistor well array and swaps them. This causes the transistors they represent to swap as well. At most only two transistors are effected, at worst a transistor is swapped with itself causing no change.

The second operator does a better job of introducing noise related to which transistors are adjacent to each other while the first merely shifts adjacent transistors to a new location in the transistor well. With building blocks often being made of adjacent transistors the second operator allows for better creation and destruction of building blocks. Gaps can be eliminated by getting the correct adjacencies between transistors and with the precedence put on removing gaps the second mutation seems logically better. In practice both mutations work well.

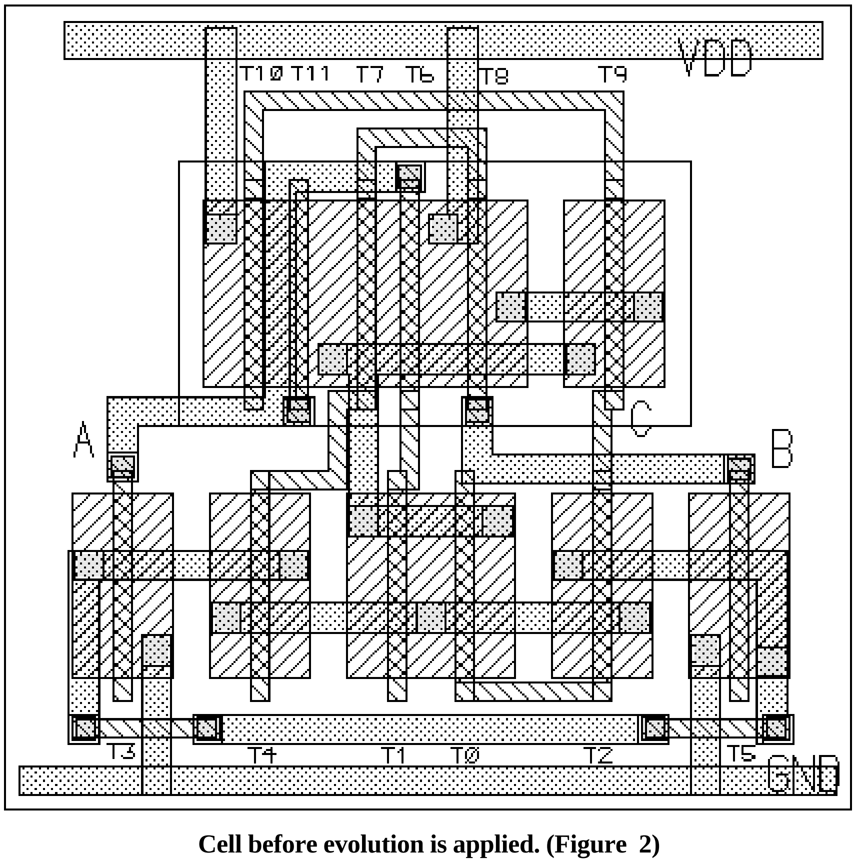
3 Experimental Results

An example run was done on a simple carry cell (Figure 1). This cell consisted of 6 NFETs and 6 PFETs with a realistic amount of interconnections. The letters A, B, and C denote inputs to the cell. The transistors are labeled T0 through T11. A probability for crossover of 10% and for mutation of 5% was chosen for the run. The population size of 4000 individuals was used. These numbers are based on previous trials and are chosen to balance the speed to convergence against searching enough of the problem space. A goal of searching at least 0.005% of the problem space has shown good results and converged within 1000 generations.



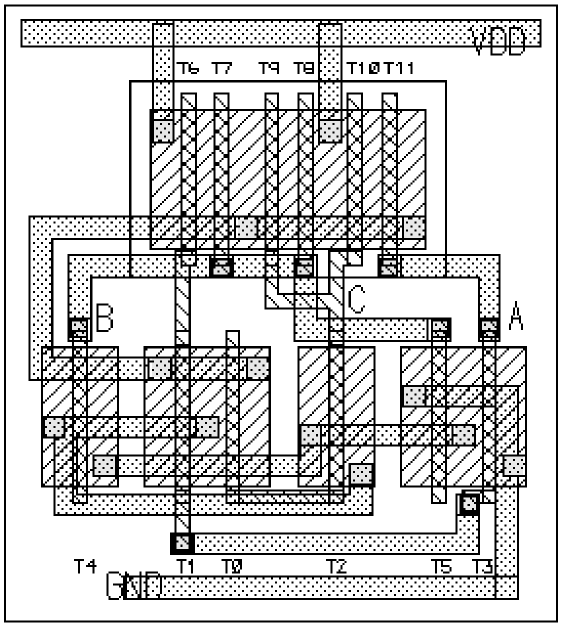
Schematic for a carry standard cell (Figure 1)

The run was initialized with a population of randomly generated individuals. The best of these is shown in Figure 2. As one can see the diagram has wires running throughout the design, large numbers of gaps, and few transistors aligned. Figure 3 shows the best individual at generation 80. The number of gaps has been reduced by two, and the wires have settled on shorter routes through the design. The final solution, taken at generation 200, is shown in Figure 4. This individual has every cell aligned and no gaps. The wires have chosen a uniform and short means of interconnecting the transistors. A graph of the fitness over time is shown in Figure 5. The way that transistors 9 and 8, 10 and 11, and 3 and 5 stay adjacent between Figures 3 and 4 hints that the proposed building block structure above holds true. It is interesting to note that at generations 59, 90, and 132 a best individual is discovered but killed in the next generation. This is most

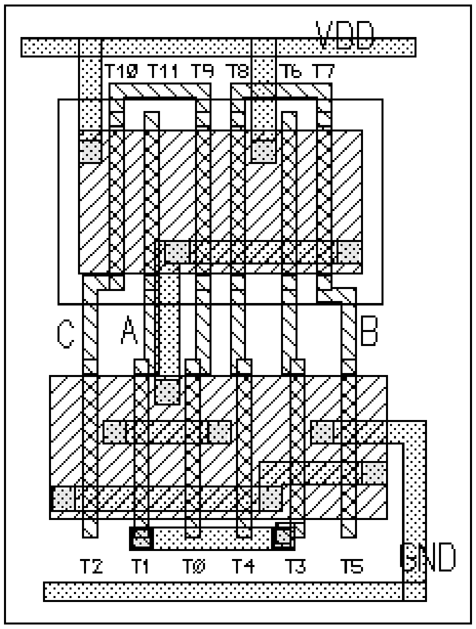


Cell before evolution is applied. (Figure 2)

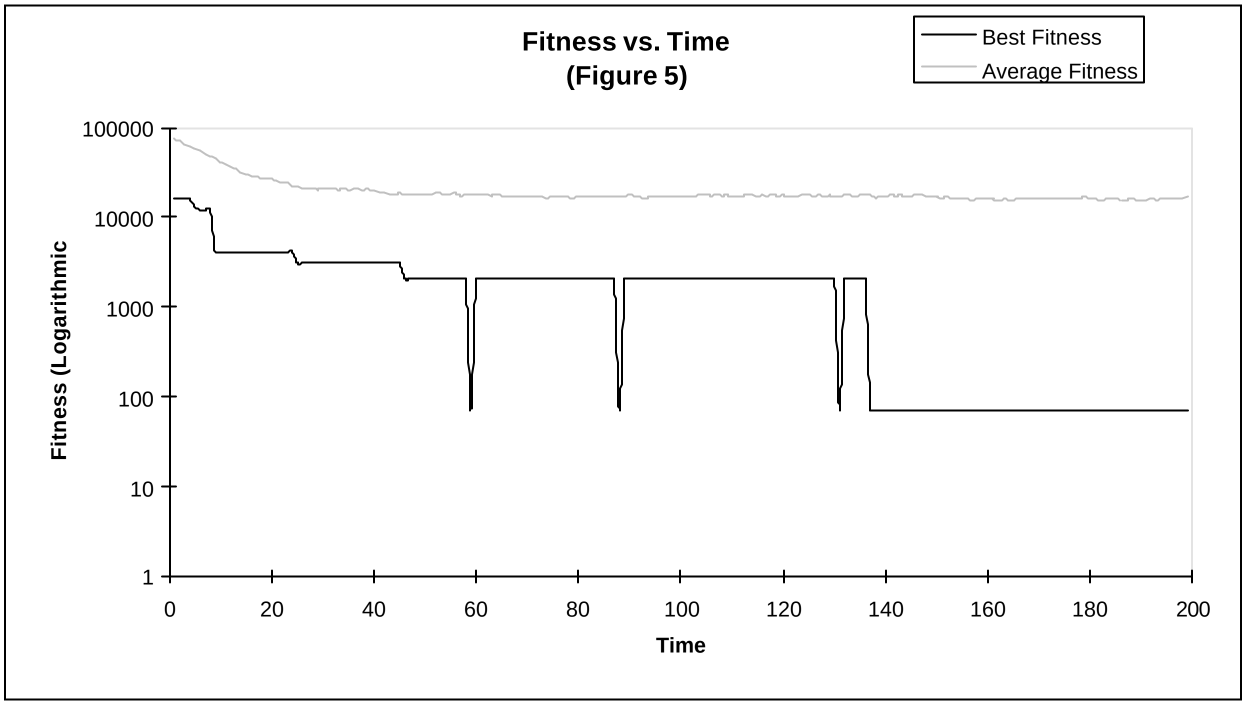
likely caused by that individual being chosen for crossover, which is a destructive operator as described above. The difference between this best solution and the next best appears to be two misalignments which causes a 1000 unit difference in the fitness and explains the sharp spike in the graph. The run visited 92231 unique solutions to the problem which is 0.0043% of the total problem space. The best individual was visited 23715 times, two orders of magnitude more than any one other individual. In general this run is representative of the results that can be achieved with this method but is better than the average result achieved.



Cell at intermediate generation. (Figure 3)



Final solution. (Figure 4)



4 Conclusion and Extensions

The method does yield a near optimal solution for most standard cells but not consistently. It often takes several runs of the program for a solution to arise which rivals a manually designed solution. We believe the reason for this inconsistency comes from the fact we are searching such a small part of the problem space. Two solutions to this problem are to introduce more noise, or increase the population size. Both methods yielded less than remarkable improvements or greatly increase run-time. It is important to note that even when running this method several times one only searches at most 1% of the problem space which shows that solutions are not being found at random.

The method also takes a simplistic approach to dealing with the multi-constraint nature of transistor placement. The current fitness function takes each of the constraints and then using weights discovered from empirical data, sums them into a final fitness function. These constant weights are only ideal for a mythical average problem and don't take into account each schematics unique concerns. A system for intelligently combining the constraints into one fitness function based on the schematic could improve the method.

References

Saika, S.; Fukui, M.; Shinomiya, N.; Akino, T., "A Two-dimensional Transistor Placement for Cell Synthesis", Design Automation Conference, 1997. Proceedings of the ASP-DAC '97 Asia and South Pacific, Page(s): 557 -562, 1997

Bean J. C., "Genetics and Random Keys for Sequencing and Optimization", University of Michigan Press, Ann Arbor Mich., 1992

Holland J.H., "Adaptation in Natural and Artificial System: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence", MIT Press, Cambridge Mass., 1992

Schnecke, V.; Vornberger, O, "Hybrid Genetic Algorithms for Constrained Placement Problems", Evolutionary Computation, IEEE Transactions on , Volume: 1 Issue: 4, Nov. 1997, Page(s): 266 -277, 1997