A State-of-the-Art Review of Placement in FPGA

Jyoti* and Pawan Kumar Dahiya**

Any digital circuit can easily be implemented using Field Programmable Gate Array (FPGA) with accuracy and fast implementation rate. The quality of digital circuit depends on the placement technique. The placement technique determines the physical location of logic block on the FPGA. In this paper, a number of placement techniques are reviewed like min-cut, simulated annealing, analytical placer, evolutionary placer and hybrid approach. Each optimization technique is evaluated and the significant aspect of each technique is explained. An overview of the tools used in FPGA placement is also given.

***Keywords*:** FPGA, Placement placer, Simulated Annealing (SA), Genetic Algorithm (GA), Hybrid approach

Introduction

Nowadays, all the modern electronic circuits are implemented using digital technique. As compared to Application Specific Integration Circuit (ASIC), Field Programmable Gate Array (FPGA) provides fast realization of digital circuits. FPGA has programmable components such as Configurable Logic Blocks (CLB), Switch Blocks (SB) and Input/ Output Blocks (IOB). These programmable blocks can easily be programmed by a designer. Typical architecture of FPGA is shown in Figure 1.

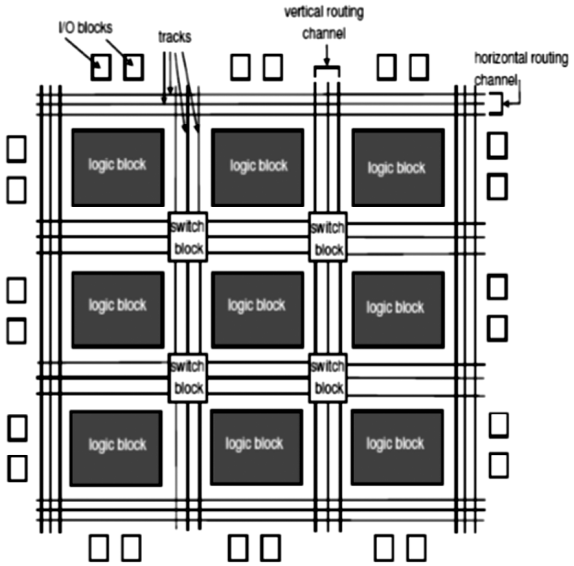
A typical FPGA design process consists of logic synthesis, technology mapping, placement and routing. The logic synthesis is a process of minimizing the logic of circuit so that more and more logic can be loaded into limited area of FPGA. Technology mapping is the process of partitioning the optimized circuit such that each partitioned block is mapped onto the available logic elements in the FPGA. In the placement phase, these partitioned blocks are assigned to the specific cells of the FPGA layout. In the routing phase, horizontal and vertical channel are used to connect the placed logic blocks. The main object of the placement problem is to minimize the total placed area such that more and more logic can be added in the available FPGA area.

The rest of this paper is organized as follows. Section 2 defines the placement problem. Some of the existing placement algorithms are briefly discussed in Section 3. Section 4 discusses the tools used in FPGA placement. Finally, the paper ends with conclusion.

* M. Tech. Student, ECE Department, DCRUST, Murthal, Haryana, India; and is the corresponding author. E-mail: geminijyoti596@gmail.com

** Professor, ECE Department, DCRUST, Murthal, Haryana, India. E-mail: pawan.dahiya@gmail.com

Figure 1: Basic Architecture of FPGA



Source: Venugopal and Manimegalai (2015)

Placement Problem in FPGA

The FPGA placement problem can be defined as follows: The inputs of the problem are modules description, consisting of shapes, size and terminal locations, and netlist describing the interconnections between terminals of modules, and output is the list of x- and y-coordinates for all modules. According to Venugopal and Manimegalai (2015), the FPGA placement problem can be formally defined as follows: At the input, a set of n modules $M = \{m_1, m_2, ..., m_n\}$ and a set of r signals = $\{s_1, s_2, ..., s_r\}$ are given, each module $m_i \in M$ can be associated with a set of signals $s_{mi} \subseteq S$. The modules may be either CLBs or IOBs, and the total number of nets is typically the sum of number of CLBs and inputs. With each signal $s_i \in S$, a set of modules M_{si} is associated, where $M_{si} = \{m_j | s_i \in S_{mi}\}$. In other words, M_{si} represents the signal net s_i. A set of slots $L = \{l_1, l_2, ... l_p\}$ are given, where $p \geq |M|$ are ranged in a two-dimensional array on the FPGA chip. Each slot $l_i \in L$ is represented by a pair of unique integer indices (x_i, y_j) of the array. The peripheral location on the two-dimensional array is reserved for IOBs, whereas the CLBs are placed inside the region bounded by periphery. The FPGA placement problem is to assign a unique location $l_i \in L$ to each module $m_i \in M$ such that the circuit can be routed with the available resources and signal delay meet timing constraints.

Techniques for Placement in FPGA

The traditional placement algorithms (Shahookar and Pinaki, 1991) are mainly divided into two major classes:

- Constructive placement; and
- Iterative improvement

Constructive Placement Algorithms: They are generally based on primitive connectivity rules. In this, first seed module is selected and placed on FPGA. Then other modules are selected, one at a time, in order of their connectivity (most densely connected first) and placed at the vacant location close to the placed module. These algorithms are very fast and provide initial placement for iterative improvement algorithms. Most resent constructive placement algorithms are quadratic placement algorithms and partitioning-based placement algorithms.

Iterative Improvement Algorithm: On the other hand, iterative and improvement algorithms produce good placement but require a large amount of computation time. In these types of algorithms, the pairs of modules are randomly selected and interchanged if cost function is reduced. These algorithms will terminate when improvement in cost function stops after a large number of trials. The current popular iterative algorithms are Simulated Annealing (SA), etc.

Other possible classifications for placement algorithms are deterministic and probabilistic algorithms (Shahookar and Pinaki, 1991). Deterministic algorithms are based on fixed connectivity rules or formulas and they will always produce the same result for particular placement problem. Probabilistic algorithms are based on randomly examining configuration and they may produce different results each time they are run. The constrictive algorithms are deterministic, whereas iterative improvement algorithms are probabilistic.

Semiperimeter method is used to estimate the wirelength. For finding the wirelength of circuit, enclose all the pints (modules) in smallest rectangle and then find half of the perimeter of rectangle which is equal to wirelength.

The popular placement optimization techniques (Venugopal and Manimegalai, 2015) are:

Min-Cut-Based Placer: It is also called placement by partitioning algorithm. In this algorithm, a given circuit is repeatedly divided into densely connected subcircuits such that the number of nets cut by the partition is minimized. Also, with each partitioning of the circuit, the available chip area is partitioned alternatively in the horizontal and vertical directions. Simple objective function is equal to the total number of nets cut by all the cut lines:

$$N_c(\sigma) = \sum v(c) \qquad ...(1)$$

where $N_c(\sigma)$ is an objective function and $\Sigma v(c)$ is the sum of all the nets cut by all vertical and horizontal cut lines. This objective function directly provides wirelength driven because minimizing the total number of nets cut using this objective function is equivalent to minimizing the semiperimeter wire length. In min-cut placement algorithm, partitioning is done by two methods (Shahookar and Pinaki, 1991):

- Cut-oriented min-cut; and
- Block-oriented min-cut.

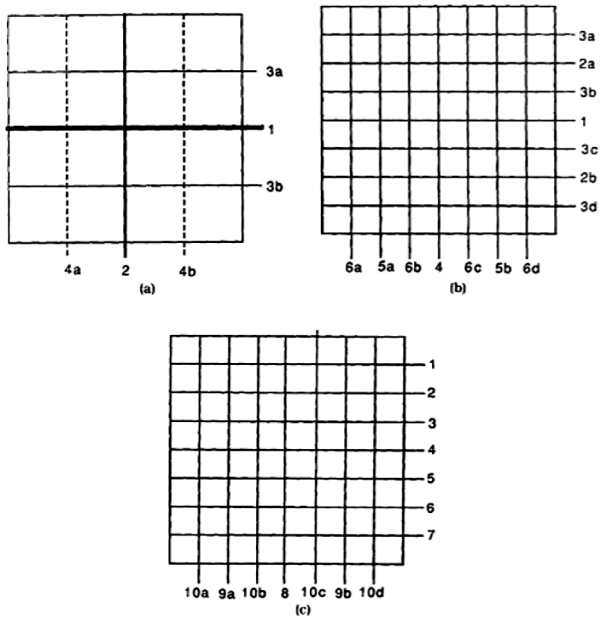
Block-oriented min-cut algorithm has separate cut line for each partition of the chip as compared to cut-oriented min-cut algorithm so it produces better result. The cut lines for partitioning the chip may be selected in any sequence. Breuer (Shahookar and Pinaki, 1991) has given three sequences for partitioning the chip, as shown in Figure 2. These are as follows:

Quadrature Placement: By this algorithm, the first chip is partitioned in the direction of breadth and then partitioned by alternate vertical and horizontal cuts. This process is illustrated in Figure 2a. With each cut, a region is equally divided into two subregions. This method is suitable for high routing density in the center.

Bisection Placement: In this procedure, the chip is repeatedly divided into two equal subregions by horizontal cut lines until each subregion consists of one row. Then each element is assigned to a row without fixing its position. Then vertical cuts repeatedly bisect each row until each resulting subregion contains only one slot. This is a good method for standard cell placement. It does not give any guarantee to minimize the maximum net-cut per channel.

Slice Bisection: In this process, modules are divided into rows by horizontal cut lines, as shown in Figure 2c. Then each row is bisected by vertical cut lines until each resulting

Figure 2: Cut sequences: (a) Quadrature Placement; (b) Bisection Placement; and (c) Slice Bisection



Source: Venugopal and Manimegalai (2015)

subregion contains only one slot. This process is most suitable when periphery has high interconnect density.

Simulated Annealing-Based Placer: In SA-based annealing process (Lee and Kaamran, 2008), material is placed at very high temperature so molecules of material have more space to move, so the probability of displacement is greater. Then the material is cooled slowly in a controlled manner such that molecules are arranged themselves in such a way that the material experiences less strain. As the temperature decreases, the probability of movement of molecules decreases. At a low temperature, the material becomes solid and movement of molecules becomes zero. SA mimics the annealing process. In SA, a range limiting function is implemented which specifies the range first, selects the module to be moved and then selects the destination within a specified range from the target location (Shahookar and Pinaki, 1991). The next step is to evaluate the change in cost function using semiperimeter method. If the cost function decreases, the move is accepted, and if the cost function increases, the move is also accepted with the probability $e^{\Delta c/T}$.

($e^{\Delta c/T}$, where $\Delta c \rightarrow$ change in cost function and $T \rightarrow$ temperature)

This allows the algorithm to avoid premature convergence to local minima and allows for hill-climbing movement that enables the SA process to reach global minimum. Then schedule the temperature as a function of number of iteration or previous temperature (e.g., $T_{i+1} = 0.1\, T_i$). Inner_loop_criterion is the criterion that decides the number of trials at each temperature which is usually fixed. Stopping_criterion terminates the algorithm when temperature or the number of iterations has reached a threshold value. There are no fixed rules for initial temperature, the cooling schedule, the probabilistic acceptance function and stopping criterion. Quality of placement and execution time depends on these parameters. There is no restriction on the type of moves to be used – displacement, interchange, rotation, and so on.

Analytical Placer: The analytical method (Sigl *et al.,* 1991; Lee and Kaamran, 2008; and Ray and Shankar, 2011) includes the force directed and quadratic programming method. Force directed placement technique introduces attracting, repelling and other supplementary forces. In this, force equation is formed by treating the coordinates of each module as variables, and net force exerted on each module by all other modules is equated to zero. We get the coordinates of all modules by simultaneously solving these equations. Quadratic programming solves the placement problem using quadratic equation. These techniques use linear approach to solve the placement problem. But placement problem is nonlinear, so linear approach does not give satisfactory result. These techniques use linear approach to solve placement problem. The general cost function for quadratic placement (Lee and Kaamran, 2008) is as follows:

$$\varphi(x,y) = \frac{1}{2}\sum_{i,j} \omega_{ij}\left[(x_i - x_j)^2 + (y_i - y_j)^2\right]$$

where x and y are the coordinates of a logic of the net-list. W_{ij} is the weight of the edge that connects nodes (x_i, y_i) and node (x_j, y_j).

Evolutionary Placer: Evolutionary placer includes population-based optimization techniques such as Genetic Algorithm (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), etc.

The GA (Eisenmann and Frank, 1998) is a very powerful optimization algorithm, which works by emulating the natural process of evolution as a means of progressing toward the optimum. The algorithm starts with an initial set of all alternatives, called the population. Each alternative or individual in the population is represented by a string of symbols. During each generation, fitness value of each individual is evaluated. Based on this fitness value, two individuals that selected from the population at a time and they are treated as parents. The individual that has higher fitness value, has a higher probability to being selected. A number of genetic operators (crossover, mutation and inversion) are applied to the parents to generate new individuals, called offsprings, which have combined features of both parents. The offsprings are next evaluated, and a new generation is formed by selecting some of the parents and offsprings, once again on the basis of their fitness, so as to keep the population size constant. At the same time, some bad genes are inherited from the previous generation, even though the probability of doing so is quite low. Thus, it is assured that the algorithm does not get stuck at some local optimum. The symbols used in the solution strings are called genes. A solution string of genes is called a chromosome. A schema is a set of genes that make up a partial solution.

ACO algorithm proposed by Wang and Ning (2009) is based on trails and attractiveness. A congestion factor is introduced in ACO-based algorithm which improves the performance. It reduces the channel width without degrading the critical path delay.

The PSO algorithm (Rout *et al.,* 2010) is a population-based optimization algorithm. Each particle guides itself to the best possible solution in its neighboring space, also known as personal best (pbest). The pbest can be related to the particles cognition of its own history in finding different results when it moves through the space. Before taking the next move, each particle considers the location of the particle with best fitness value, which is called global best (gbest). It improves the wirelength of circuit.

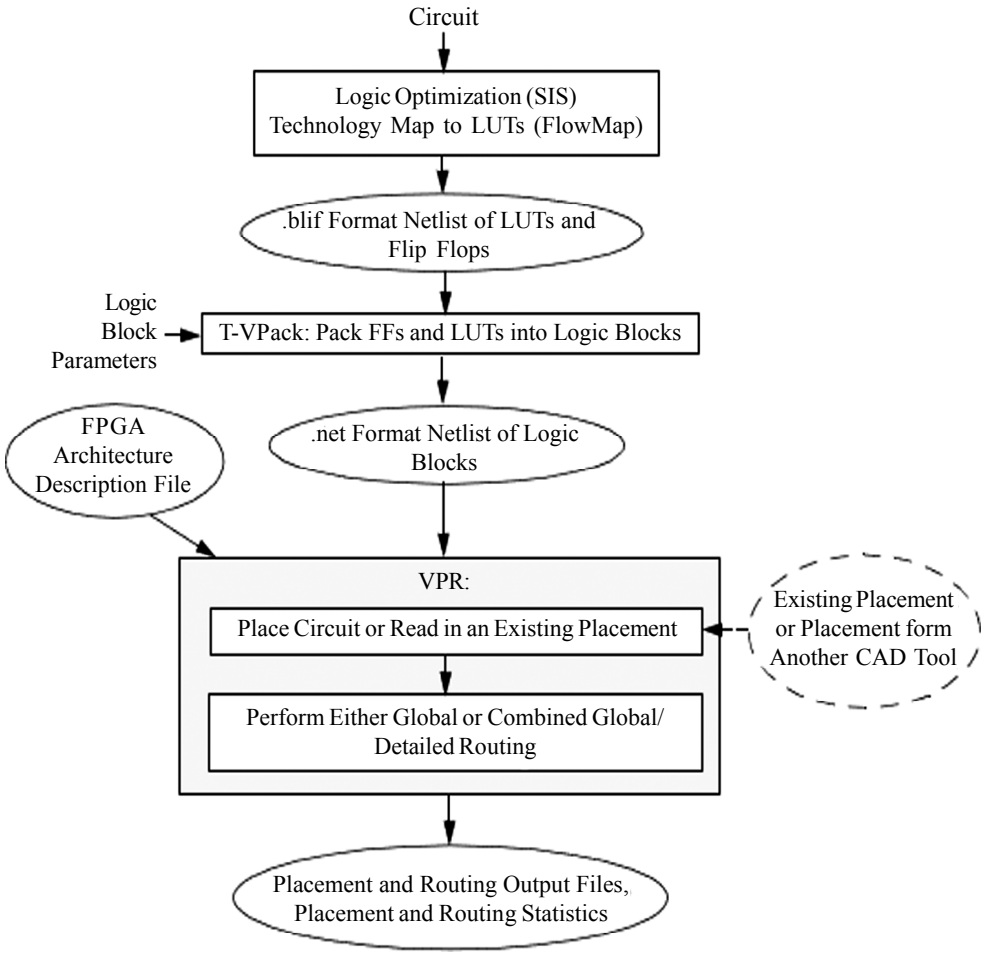
Hybrid Placement: In hybrid approach, a combination of two different techniques is used to find better result. The paper explains GA with SA (Lee and Kaamran, 2008; and Pawan, 2010)—a combination of Genetic Algorithm and Simulated Annealing (GASA) is used for the placement of symmetrical FPGA. This algorithm consists of two stages. In the first stage, GA is used for global placement, and in the second stage, SA algorithm is used for local placement. By this approach, the advantage of GA is used to find global solution, and to overcome the slow conversion of GA in late phase, SA has been used. GA provides very small improvement in late phase process by expense long time. SA is able to provide fast improvement in late phase process as compared to GA. Therefore, after a certain number of generations, SA is used to provide optimization at low temperature.

FPGA Placement Tools

Many tools have been used in literature for FPGA placement such as TimberWolf, Versatile Place and Route (VPR) and MATLAB. The most commonly used tool for placement is VPR. Typical CAD Flow is given by Betz (2000).

Figure 3 illustrates the CAD flow. The first stage is SIS, which provides technology independent logic optimization of each circuit. Then using flow map algorithm, circuit technology is mapped into 4-LUTS and flip flops. The output of flow map algorithm is in .blif format and .blif format netlist is given as input to T-Vpack software which converts 4-LUT and FFs into more crossgrain logic blocks. The output of T-Vpack software is in .net format and this output is given as input to VPR tool which generates two output files, one describing circuit placement and another describing circuit routing.

Figure 3: CAD Flow



Source: Betz (2000)

T-V pack is a packing program software which can be used with or without VPR. It takes a technology-mapped netlist consisting of Lookup Tables (LUTs) and Flip Flops (FFs) as input and then packs the LUTs and FFs together to form more coarse-grained logic blocks. The netlist as input is in .blif format and output is in the .net format. This output can be fed directly into VPR. T-V pack is invoked by typing:

t-vpack input.blif output.net [-options]

Versatile Place and Route (VPR) is used for FPGA placement and routing. It is invoked by typing:

vpr netlist.net architecture.arch placement.p routing.r [-options]

VPR requires four necessary parameters and many optional parameters. Netlist.net and architecture.arch are the inputs netlist of VPR and placement.p and routing.r are outputs files of VPR. Netlist.net describes the circuit to be placed and/or routed, while architecture.arch describes the architecture of target the FPGA, in which the circuit is to be realized. If VPR is placing a circuit, the final placement will be written to placement.p; if VPR is routing a previously placed circuit, the placement is read from placement.p. The final routing of a circuit is written to file routing.r.

Conclusion

In FPGA, placement problem technology mapped netlist is given as input, in which circuit is partitioned into small logic which is directly mappped to the CLB of target FPGA such that wirelength of interconnection is reduced.

The min-cut/partition-based technique is fast. It has open cost function, i.e., it can provide wirelength driving or timing driving placement. Its quality is not good as compared to SA algorithm and it does not provide global optimal solution.

SA placement technique also has an open cost function. Also, it is able to reach global optimal solution. Placement quality of SA is best. However, the SA algorithm is very slow. To improve the speed of SA, a new adaptive placement algorithm, called Greedy Simulated Annealing (GSA), is developed, which employs a short-term memory to record recent search history (Du *et al.,* 2004).

Analytical placer produces good results in short run-time but cannot handle complex constraints. It can handle large designs without affecting the cost of computation time. However, this technique reduced squared wired length but cannot minimize the total wirelength. Also, this is a single optimization technique which provides only wirelength driven optimization, and does not provide any timing driven optimization. It uses linear technique to solve nonlinear placement problem.

The GA has an open cost function since it can provide time driven or wirelength driven placement. It can handle two or more variables at a time. Also, it is able to reach the global optimum solution. A significant time is spent in the late phase of the process of

GA, in which small improvement is obtained extremely slowly. So hybrid technique is proposed which is the combination of SA and GA, in which advantages of GA and SA are combined to form a algorithm called memetic algorithm. It provides faster computation than GA. However, this optimization technique is complex and is difficult to implement.⏮

References

1. Betz Vaughn (2000), *VPR and T-VPack User's Manual* (Version 4.30), March 27, p. 10.
2. Du P, Grewal G, Areibi S and Banerji D (2004), "A Fast Adaptive Heuristic for FPGA Placement", *Circuits and Systems, NEWCAS, The 2nd Annual IEEE Northeast Workshop on* IEEE, pp. 373-376.
3. Eisenmann Hans and Frank M Johannes (1998), "Generic Global Placement and Floorplanning", Proceedings of the 35th Annual Design Automation Conference, ACM, pp. 269-274.
4. Lee Sang-Joon and Kaamran Raahemifar (2008), "FPGA Placement Optimization Methodology Survey", Canadian Conference on Electrical and Computer Engineering, CCECE, IEEE, pp. 001981-001986.
5. Pawan Kumar Dahiya (2010), "Recent Trends in Evolutionary Computation", Doctoral Dissertation, University of Science & Technology.
6. Ray B N B and Shankar Balachandran (2011), "A New Wirelength Model for Analytical Placement", IEEE Computer Society Annual Symposium on VLSI, IEEE, pp. 90-95.
7. Rout Prakash Kumar, Acharya D P and Panda G (2010), "Novel PSO Based FPGA Placement Techniques", International Conference on Computer and Communication Technology (ICCCT), IEEE, pp. 630-634.
8. Shahookar Khushro and Pinaki Mazumder (1991), "VLSI Cell Placement Techniques", *ACM Computing Surveys (CSUR),* Vol. 23, No. 2, pp. 143-220.
9. Sigl Georg, Konrad Doll and Frank M Johannes (1991), "Analytical Placement: A Linear or a Quadratic Objective Function?", Proceedings of the 28th ACM/IEEE Design Automation Conference, ACM, pp. 427-432.
10. Venuopal Nagalakshmi and Manimegalai R (2015), "Wirelength Driven Placement for FPGA Using Soft Computing Technique", International Conference on Soft-Computing and Networks Security (ICSNS), IEEE, pp. 1-5.
11. Wang Kai and Ning Xu (2009), "Ant Colony Optimization for Symmetrical FPGA Placement", 11th IEEE International Conference on Computer-Aided Design and Computer Graphics, CAD/Graphics', IEEE, pp. 561-563.

Reference # 59J-2018-01-03-01