FPGA Placement and Routing Algorithm: A Survey

Name: Xin Zhao

Class: Cpre583 2011 fall

Abstract

Unlike ASICs, FPGAs have routing fabrics that are pre-manufactured. And because of this prefabrication, FPGAs hardly can achieve high clock frequencies which offered by ASICs. Thus, there is a need for better FPGA timing performance. And design automation or computer-aided design (CAD) tools for field programmable gate arrays (FPGAs) have played a very critical role over the past decades for FPGA design. This mini survey presents the up-to-date overview of placement and routing algorithm for FPGA devices with an emphasis on the recent research from 2008.

1. Introduction

Placement and routing is an interconnecting stage in the design of PCB, integrated circuits, and FPGAs. According to [1], placement is a process that decides where to place all electronic components, circuitry, and logic elements in a generally limited amount of space. Then the following routing process decides the routes of all the wires needed to connect the placed components. This step must achieve desired connections but normally is constrained by the rules and limitations based on manufacturing process. Compared to ASIC, while reconfigurability of the FPGA device contributes to the flexibility that one can change the architecture or the application implemented on the FPGA without requiring any physical action on the device, the placement and routing turn to be more difficult to find a effect solution because the available logic components and routing paths are already pre-established inside of the device. In the recent years, as FPGA devices moving to have more and more components inside to meet the increasing demand of internet, video, audio, and computation, many articles have been published to optimize the placement and routing problem for FPGA device. The rest of this survey will as follows:

1) Placement
   - Serial algorithm
     - Analytical based method
     - Simulated annealing based method
   - Parallel acceleration
2) Routing
   - Serial algorithm
     - Geometric based method
     - Boolean based method
   - Hardware and parallel acceleration

2. FPGA placement

At the placement stage, slot assignment problem should be solved. The input to the problem is a circuit netlist after synthesis and it contains a various types of logic blocks which need to be interconnected in the following stage, routing. The result of placement is a mapping of all blocks onto physical resources on the target FPGA in a way that without violation of all constraints. The very common constraints are minimizing the total wire length among the route signals, because minimum wire length reduces the values of signal delay and power etc. As the amount of interconnect in any given region of the FPGA is fixed, if more interconnection is required than what is available, the circuit may not be routable [2]. The result of this stage has significant impact on the routability of the later step and performance when device is put to work.

The existing approaches to FPGA placement can be divided by serial method and hardware or parallel acceleration. For the traditional serial method, there is simulated annealing based method, analytical based method.

2.1 Serial algorithm

2.1.1 Simulated annealing based method

According to [3], simulated annealing is a heuristic-based search for minimizing an objective function which takes real values over a set of states. Given a perfect cooling schedule, simulated annealing will ultimately converge to an optimal solution. The well-known overall algorithm is as following.

```
P = InitialPlacement ();
T = InitialTemperature ();

while (ExitCriterion () == False) {
    while (InnerLoopCriterion () == False) {  /* "Inner Loop" */
        P_new = PerturbPlacementViaMove (P);
        ΔCost = Cost (P_new) – Cost (P);
        r = random (0,1);
        if (r < e^(-ΔCost/T)) {
            P = P_new;      /* Move Accepted */
        }
    }  /* End "Inner Loop" */
    T = UpdateTemp (T);
}
```

The cost fuction penalizes placement which require more routing in the narrower channels.

$$Cost = \sum_{n=1}^{N_{nets}} q(n)\left[\frac{bb_x(n)}{C_{av,x}(n)} + \frac{bb_y(n)}{C_{av,y}(n)}\right]$$

The simulated annealing method enforces all the legality constraints imposed by the FPGA architecture, and it is possible to model the impact of the FPGA routing architecture on circuit delay and routing congestion.

In the updated simulated annealing method, intelligent deterministic strategies for selecting and replacing "poorly placed" cells were interspersed with random simple moves during annealing. This kind of technique serves to reduce the size of search space by focusing on cells and locations of interest. This technique allows the annealer to converge more quickly and attain a better placement for the same amount of run-time as if random move had been used alone.

A hybrid parallelization method of the simulated annealing based placement algorithm proposed by [4]. The proposed technique used balanced region-based partitioning and multithreading. In the first step of this approach placement sub-problems are created by partitioning and then processed concurrently by multiple worker threads that are run on multiple cores of the same processor. The new hybrid parallel placement algorithm achieves an average speed-up of 2.5× using four worker threads, while the total wirelength and circuit delay after routing are minimally degraded. The algorithm of [4] is shown as below:

Partitioning and Multithreading-based Placement() *{*
- *Step 1: Main thread*
  Multi level 4-way partitioning into subchips
  Subchips added to queue of tasks
- *Step 2: Worker threads*
  Process tasks concurrently until queue becomes empty
- *Step 3: Main thread*
  Ultra fast top-level low temperature SA refinement

}

The main placement problem is partitioned into multiple balanced regions using multi-level 4-way partitioning. These kinds of tasks were placed into a common queue. Then, the worker threads process these tasks from the common queue in parallel and independently. The solution of each task is placed back into the corresponding task object from the queue. The results were then read in and assembled by the central manager which runs on the main thread to generate median result. At last, the median result is further improved by an ultra-fast low-temperature annealing re-finement step.

2.1.2 Analytical method

The best algorithms for placement on FPGAs are based on Simulated Annealing which has been discussed above. However, Simulated Annealing is not scalable due to the long convergence time. In the placement, the wire-length is approximated by half the

perimeter of the smallest bounding box which contains all terminals of a net inside. This method is called HPWL. For a net of more than two terminals, the routing cost obtained by the HPWL model is not accurate. That means the HPWL cannot be efficiently minimized. To avoid this drawback, many analytic methods used a quadratic wire length objective function. But this will over-emphasize the optimization of longer nets and sacrifice the short nets. For solving this problem, an analytical method named StarPlace model is proposed by [2]. This method used star net to model each block which has a connection to the center-of-gravity of the net. The length of each edge is the quadratic distance (x_i-x_{cl}) 2 from blocks i to the center-of-gravity x_{cl}. The difference between this model and other models is that it does not seek to minimize the sum of all squared distances between any pair of blocks that connect a common net, but rather seeks to minimize the sum of the square roots of the sum of the quadratic distances between each block and the center-of-gravity of a net. Thus, by avoiding quadratic distance measures, this model avoided overemphasizing the cost of long nets at the cost of shorter nets when performing optimization. On the other hand, incremental changes in cost resulting from block movement can be computed in O (1) time, regardless of the size of the net. And by using conjugate gradient, the runtime can be reduced much.

2.2 Parallel acceleration

To further improve the runtime of placement for FPGA, a parallel method was proposed by [5] which based still on simulated annealing algorithm. It parallelizes the Simulated Annealing moves. Then the entire moves are calculated by all cores.

Traditional simulated annealing algorithms used random moves which are within a window, and keep shrinking it just like anneal progress. But in [5], the vast majority of moves are partially random, and both the selection of the block(s) to move and the destination location(s) are heavily biased. In this way, the process of searching space is far more efficient. Another new feature was that, this algorithm used combined eight costs from eight cores to optimize wire length, timing, power and localized congestion to produce an overall cost used to accept or reject moves. Meanwhile, similar to the famous VPR algorithm, this algorithm also clusters the netlist into blocks such as RAM, logic and DSP blocks. And performs much of placement with the large blocks. Then after setting the placements of these large blocks, decompose the internal logic clusters back to LUTs and registers such that these small logic can be fine-tuned.

This parallel acceleration of placement can achieve speedup of up to 2.8X and 3.7X, with geometric average speedups of 2.1X and 2.4X compared to traditional simulated annealing algorithm.

3. FPGA routing

The routing algorithm can be divided into serial method or parallel method. For traditional serial method, Boolean based method is the basic one. But nowadays, the geometric based method turned to be more powerful.

3.1 Serial algorithm

3.1.1 Geometric routing algorithm

The geometric routing algorithm is based on rip-up and re-route approach. It allows to describing easily the targeting architecture, but the disadvantage is that it is difficult to achieve convergence of routing solution sometimes. There are 2 phases in this method usually. In phase 1, the algorithm checks for routing resource violations, and in phase 2, it checks for timing violations such as wire delay, net delay and net slack. For searching the routable paths, it can use both of exhaustive search methods, the breadth first search and depth first search. In the breadth first search, the goal is to minimize the total wire length. The cost function is biased towards distance at this time. On the other hand, if using depth first search, the cost function is biased toward congestion and distance. The famous Pathfinder algorithm adapts the 2 phases above to: phase1, initial routing and phase2, rip-up and re-routing.

The cost function: Cost (n) =b (n)*h (n)*p (n)

in which b(n) is base cost, h(n) is the historical congestion, and p(n) is the present congestion penalty. In [6] by the routability-driven routing, the algorithm named SprialRout guarantees that long-path timing closure is achieved when it terminates. For detail, the long path timing-driven routing problem is that, given a circuit and its placement on an FPGA and a long-path timing target of some amount of units, the goal is to find routing trees in this FPGA routing network for each vertices. The constraints were that these trees should be mutually vertex disjoint with respect to the mutex resources; for each edge in this graph, there is a contiguous path in this tree from the starting point of this edge to ending edge of this edge, which means it is topologically complete without opens; for each path the sum of vertex delays through its trace in the routing is less than the target amount. The algorithm [6] was shown as following:

SPIRAL-ROUTE(G_c, G_v, D, n)

1 Route all nets in delay-only mode
2 Obtain minimum delay $d_{\min}(a)$ for each $a \in A_c$ for edges in the circuit graph
3 $budget \leftarrow$ COMPUTE-TIMING-BUDGET$(G_c, d_{\min}, \delta, D)$
4 $d_{\max}(a) \leftarrow d_{\min}(a) + budget(a)$ for each $a \in A_c$
5 $congestion(v) \leftarrow 1$ for each $v \in V_w$
6 **for** $iteration \leftarrow 1$ **to** n
7 **do for** each $v \in V_c$ of G_r
8 **do** $\triangleright$ T_v is the existing routing tree for net source v
9 $T_v \leftarrow$ SPIRAL-SEARCH$(v, T_v, congestion, d_{\max}, G_r, G_c)$
10 **if** Routing is mutually vertex disjoint
11 **then** Return successfully routed solution $\{T_v | v \in V_c\}$
12 **else** $congestion(v) \leftarrow congestion(v) + 1$ if v is overused
13 Report routing failure

The efficacy of SpiralRout was demonstrated by completing the 19 instances of long-path timing-driven routing within timing budget which were not completed by the famous VPR with version of 4.3. The results shown that, routability of SprialRout depends heavily on budget solution quality. SprialRout was about 7-13 times slower than TDVPR depending on how TDVPR is configured.

3.1.2 Boolean based routing algorithm

Boolean SAT-based routing algorithm transforms the FPGA routing task into an atomic Boolean function. The generated Boolean function is satisfiable if and only if the design is routable. Thus, if the generated Boolean formula is satisfiable, then any satisfying assignment corresponds to a feasible routing of the channel; otherwise the channel is un-routable.

To solve the problem of FPGA routing with minimum number of tracks per channel, an ant colony optimization algorithm was proposed by [7] recently. The routing task is transformed into a Boolean satisfiability (SAT) equation with the property that any assignment of input variables that satisfies the equation specifies a valid route. At the first step, this equation is modeled as Constraint Satisfaction problem. Any solution for the equation is a particular routing and reversely, if there is no such a assignment means the routing is impossible. The second step is applying ant colony optimization algorithm on the Boolean equation for solving routing alternatives. Ant colonies can find the shortest paths from their nest to food sources because ants communicate indirectly by disposing traces of pheromone as they walk along a chosen path. Ants most likely prefer those paths possessing the strongest pheromone information. In general, the algorithm is as below:

At first, initialize the pheromone values and generate the entire possible SAT problem. Let M be the number of nets in the routing circuit. In this way, the pheromone information is encoded in a matrix. Then, in 2^M iterations, take a subset of SAT problem and generate ants to represent each chosen SAT problems. Simulate the ant movement and take a subset of ant and insert them into the queue. Generate test pattern and perform logic simulation. Then update pheromone strengths until all the SAT problems are solved.

The experiment results shown that, by compare to other algorithm such as zChaff and GRASP, the ant colony optimization took fewer amount of time and use minimum channel width to route FPGA chips.

3.2.1 Hardware and parallel acceleration

Similarly to placement, the parallel processing can also be used for routing algorithm. In [8], a parallel and hardware acceleration method by using FPGA was proposed which finds a quasi-minimum Steiner tree for multi-point connections in a VLSI chip or a PCB, and also, this algorithm can be used for FPGA itself.

This method adopted the star net algorithm, in which, the center point for a net to be routed must be found at first. This center point then became the common connection of all the other nodes. In this way, the routing net can achieve minimum wire length. For the connecting of center point with all other nodes to achieve minimum wire length, the Lee algorithm was used, which had three phases, waveform expansion, backtracking and cleanup. This method used a special-purpose application-specific SIMD processor to implement the parallel algorithm. The SIMD processor had n*n array of processor elements. And each of them is a finite state machine which was directly mapped to a grid point during routing. VHDL code was used to implement a prototype of 4*4 and 8*8 full grid hardware accelerator. Another PE and Control Unit were designed to accommodate the algorithm. The algorithm had two phase, the first was center point searching. For a multi-node net, each cell communicated with its orthogonal neighbors on a single line. All nodes started wave expansion concurrently, but are multiplexed in time in a round robin style. In the phase two, because the minimum center point was already found by phase one, a path was iteratively determined from the center point to all the nodes in this net. The experiment results shown that, for an 'n' nodes net in terms of clock cycles, the total route time was 2(n*d+F1+F2+F3…) where d is the value of the field, and F was the distance from the node to center point.

4. Conclusion

This survey has explored many issues in the placement and routing world of FPGA devices. While these devices changed dramatically in last decade and the placement and routing algorithm followed. And although we can borrow some kinds of ideas from ASIC placement and routing algorithm, it is clear that many problems still remain and the placement and routing problem became more and more complicated.

References

[1] http://en.wikipedia.org/wiki/Place_and_route, Place and Route, wikipedia.org

[2] M. Xu, G. Grewal, and S. Areibi, "StarPlace: A new analytic method for FPGA placement", *Integr. VLSI J.* 44, 3, pp. 192-204. 2011

[3] K Vorwerk, A Kennings, J W Greene, "Improving Simulated Annealing-Based FPGA Placement with Directed Moves", IEEE Trans. ComputerAided Design of Integrated Circuits and Systems , Volume: 28, Issue: 2, 2009 , pp. 179-192

[4]Cristinel Ababei, "Speeding up FPGA Placement via Partitioning and multi-threading", International Journal of Reconfigurable Computing, Vol 2009, 2009

[5] Adrian Ludwin and Vaughn Betz, "Efficient and Deterministic Parallel Placement for FPGAs". ACM Trans. Des. Autom. Electron. Syst.16, 3, Article 22, June 2011

[6] Keith So. "Enforcing long-path timing closure for FPGA routing with path searches on clamped lexicographic spirals". In Proceedings of the 16th international ACM/SIGDA symposium on Field programmable gate arrays, pp.24—34, 2008

[7] Vinay Chopra and Amardeep Singh "Ant Colony Optimization approach for solving FPGA routing with minimum channel width" International Journal on Computer Science and Engineering, 2011, Vol.3(7), pp.2855

[8] Fatima, K.; Rao, R.; "FPGA implementation of a new parallel routing algorithm," TENCON 2008 - 2008 IEEE Region 10 Conference, pp.1-6, 19-21 Nov. 2008