A Study on Place and Route for FPGA using the Time Driven Optimization

Yi, Myoung Hee*, Yi, Jae Young**, Shuji Tsukiyama***, and Szirmay Laszlo****
* Korea Power Exchange E-mail: yi3253@kpx.or.kr
** Technical Univ. of Budapest, E-mail: yi_young@hotmail.com
*** Chuo University, Japan,
**** Technical Univ. of Budapest, Professor E-mail: szirmay@bme.iit.hu

Abstract:
We have developed an optimization algorithm based formulation for performing efficient time driven simultaneous place and route for FPGAs. Field programmable gate array (FPGAs) provide of drastically reducing the turn-around time for digital ICs, with a relatively small degradation in performance. For a variety of application specific integrated circuit application, where time-to-market is most critical and the performance requirement do not mandate a custom or semicustom approach, FPGAs are an increasingly popular alternative. This has prompted a substantial amount of specialized synthesis and layout research focused on maximizing density, minimizing delay, and minimizing design time.

Keywords: FPGA, Route, Time Driven, Optimization, wirability

1. INTRODUCTION

In recent year FPGAs have become an important alternative technology to Mask Programmable Gate Array(MPGA) due to their shorter design cycle, reprogramability and low costs. FPGAs have found increasing use in prototyping, emulation of large ASICs, and in many low volume products. In general their use has been limited to implementing random logic with non critical timing requirements. FPGAs are an exiting new approach to application specific integrated circuits(ASICs) that drastically reduce the time-to-market and also reduce the cost for low to medium volume production. FPGAs have an array of logic cells connected by a general routing structure, like a MPGA, but they are programmable like PLDs. The complexity of FPGAs has increased to the point where automatic design tools are essential. However, because the routing fabric, connection mechanisms and timing issues are different for FPGAs, specialized layout tools are required.[1][2]

Exiting automated layout tools for FPGAs are inherently sequential in nature with the placement, global and detailed routing steps being distinct and separate. There therefore suffer from the predictability problem specific to FPGAa. Specifically, a placer might bring together interconnected cells very close to each other without realizing that the resulting placement is unroutable due to fixed routing to resources and their specific connectivity. Also, paths that look non-critical during the placement level might become critical after routing. The reason is that it is especially difficult to predict interconnect delays for FPGAs during the placement process.[3][4] Simple heuristic such as path's delay is proportional to its length often do not work. Essentially, for FPGA, it is very difficult to design a good wirability and timing metric at the placement level.

While the ability to predict wirability and timing behavior for FPGAs at the placement stage is very difficult, it continues to be true that the ability to effect a substantial charge in the wirability and timing behavior is much greater at the placement stage compared to the routing stage where the flexibility in very much limited due to the already exiting placement.

2. LAYOUT FLOW FOR FPGAS

FPGAs have fixed logic structures like MPGAs. However, in addition, the routing resources are fixed. The logic structure can be used for a large number of functions. The fixed routing resources appear in the form of a set of disjoint segments. In order to achieves a particular circuit connectivity, there disjoint segment can be connected where desired without having to go through any time-consuming fabrication process.

The two kinds of FPGAs ,that our research targets, are the now-based FPGAs and the island-stage FPGA. The layout flow for now-based FPGAs is shown figure 1. Logic synthesis converts a high level circuit description to a netlist of generic cells. Technology mapping then maps these generic logic elements onto logic nodules there by creating a netlist of logic module sized cells.[5] This netlist forms the input for the placement phase. For row-based FPGAs, logic modules can be of different types. The technology mapper needs to comprehend this. At the placement stage, cells are mapped auto valid logic module locations. It is ensured that logic module types and the mapped cell's type match. At this stage, the routing resources information is absent. Therefore, the place optimizes based on net-length minimization and estimated congestion minimization criteria.[6] At the global routing stage, feed through are assigned to nets which need them.

In figure 1, for example, net needs a feed through and this gets assigned at the global routing stage. Once the global routing is done, the channel problems are defined. At the detailed routing stage, the horizontal routing resources or segments are assigned to net in channels.[7]

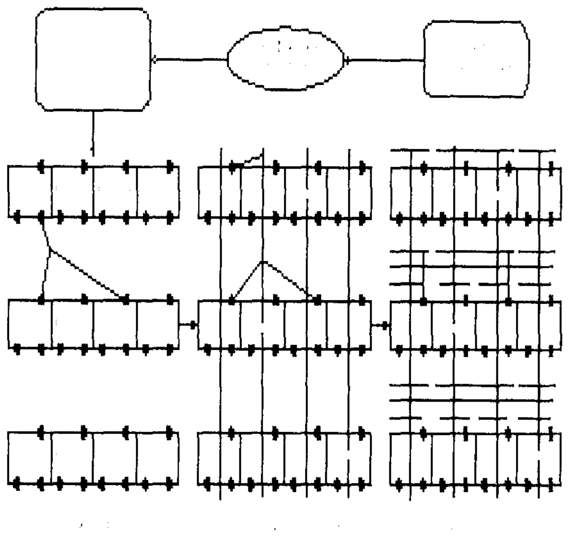

Figure 1. Layout flow for row-based FPGAs.

The design flow for island-style FPGAs is shown in Figure 2. As in the earlier case, the technology mapper converts a high-level circuit description into a net-list of I/O blocks and CLBs.[8] The placer's job is similar to the earlier case since the difference in these two styles of FPGAs are primarily in their routing resources. At the global routing stage, paths are determined for every connection. Paths connect ports of CLBs through connection and switch blocks. The global router does not have any information regarding of the contents of the connection switch blacks and makes the path decisions based on estimated congestion.

In figure 2, for example, the auto paths for the three-terminal nets are found at the global routing stage. At the detailed routing stage, explicit routing resources are assigned to nets, i. e., the detailed router decides which routing resources to use for a particular path of a net.

3. TIMING DRIVEN PRE-PLACEMENT

For the placement phase, existing placers

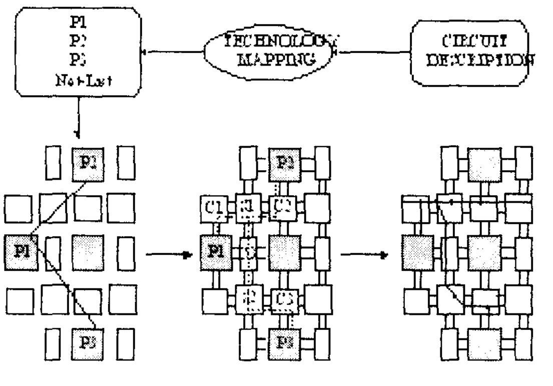

Figure 2. Layout flow for island-style FPGAs.

for gate-arrays and standard cells like timber wolfe placement and routing package[9] are used with modifications to suit the specific architecture. An example of such a modification for row-based FPGAs where vertical routing resources are few, is to increase the weight of the height of the bounding box of nets while doing the bounding-box based optimization.

In this paper, we approached to timing driven placement as follows ;

1) A sign wire length targets to each connection to exploit the allowable slacks in each path during placement. We use a simplified version of the zero-slack approach.

2) Arrive at a placement configuration that minimizes the deviation from assigned wire lengths using a simulated annealing based placement and global routing. The global routing for each net is preformed using a single trunk steiner tree approximation. The cost function minimized is ;

$Cf = \alpha * wl + \beta * Tp + \gamma * Cp$
wl : Total wire length
Tp : Total timing penalty which is computed as the square of the wire length for each connection.
Cp : Total congestion penalty which is computed as the square of the oversubscribed track resources for each channel segment associated with each LUT(Look up Table).

We are a simple move set(i. e. displace and swap) and a cooling schedule consisting of fixed number of temperature.[9]

4. PRE-ROUTING PHASE

The two requirement of the routing phase are to achieve 100% wirability and to meet the timing requirements. Specific FPGA routing architectures require correspondingly targeted routing algorithm. J. Rose[6] considers routing for the segmented channel architecture of row-based FPGAs. An additional constraint for such architectures is that only one track can be used for connection, i. e., no doglegs are allowed. In the connections[10] are sorted and processed according to their left-edge. At every intermediate point, a frontier is maintained which represents the tracks available and corresponds to a particular selection of tracks for previous connections. When processing the (i+1)th connection, there could be L different frontiers depending on the selections made for the previous i connections. An exhaustive search would require updating the frontier by testing the (i+1)th connection with all the L frontiers.

Although the general problem is NP-complete, the K-segment problem can be attacked with efficient dynamic programming techniques. In order to make the run-time practical, heuristics can be used to pure the frontier based on a wirability and timing driven cost-function.

J. Rose[6] addresses the problem of routing for inland-style architectures. The global routing here is based on standard-cell global routing[11] which assigns paths for connections so as to balance channel densities. This is followed by a detailed routing which uses the global routing solution to guide the assignment of specific routing segments to connections. Many paths for each connection are explored concurrently via a search heuristic with an intelligent pruning mechanism. At the end of the search phase, multiple paths are determined for each connection. The next phase selects a path from among the alternatives. This is done using a cost function which assigns a weight to a routing resource based on the number of alternatives requiring it. This paper includes timing and other wirability factor in the cost-function. The path with the lowest cost or an essential path is chosen and the updates are done to include the effect of this selection. If a connection looses all its alternatives, that past of the routing graph is re-expanded. Also this paper addresses the timing issue by incorporating pre-routing critical path/net information in their cost-function and generally minimizing the usage of antifuses/switches. To address performance based routing more directly, J. Frankly[12] attempts to iteratively improve the performance by modifying the delay bounds on connections which guide the router. For each connection, a lower bound on delay is supplied. This so-called "limit-bumping algorithm" reassign the upper bounds of delays allowed on each connection i. e., the slacks allowed for the connections.

The reassignment is guided by the room that the connections have for delay improvement based on the derived delays with existing routing and the lower bound. The router itself is essentially a cost-based maze routes which router while attempting to respect the delay bounds on the nets. This process of assigning delay bounds and re-routing is continued till some stopping criterion is net.

Simultaneous placement and routing techniques have been explored for areas like custom analog cells. However, for FPGAs, we are not aware of any attempts to date exploring the utility of full performance-driven, simultaneous placement and global and detailed routing.

5. DETAILED ROUTING

The conventional channel is defined to be a rectangular area with terminal rows along its top and bottom sides. The main role of a channel router is to connect the given terminals according to the net list with minimum area. For over true decades, many researchers have tied to solve various channel routing problems. D. F. Wong[13][14] presented a multi-large channel router that transforms routing solutions from existing two-layer routers. Recently, the concept of over-the-call routing has been introduced to minimize layout area. In over-the-call routing, the cell layout area, as will as the channel area between two cell rows is used a routing resource. Some heuristic have been developed to achieve a 20% to 35% reduction in channel height as compared to those for non-over- the-call channel routers.[15] A recent algorithm yields as much as a 65% reduction, in the channel height with a triple-layer over-the- call router.

Routing algorithm in the study is developed for generic row-based design structure. Hence, the algorithm does not impose any strict constraints on the cell structure or terminal positions. Also, this algorithm allows both constrained and unconstrained dogleg, while others only allow unconstrained ones. The dogleg usage should comply with the design goal. If compactness is the primary goal, both types should be used. Such a strategy enables us to apply the algorithm to various design methodologies such as custom cell-based systems as will as to standard cell-based systems. This strategy, however, in creases the problem complexity. Since it is very difficult to optimize the usage of all resources in the over-the-cell area at once, we divide the whole over-the-cell routing problem into several subproblem and solve one at a time for the optimal solution of each subproblem.

This algorithm is divided into piece steps :

1) Initial double-layer routing,
2) Track-to-metal 3 transformation,
3) Segment-to-meta 3 transformation,
4) Segment-to-meta 1 transformation,
5) Triple-layer channel routing.

The first and the last steps use conventional channel routings. In the first step, we use a double-layer channel routing to obtain a good seed for over-the-cell routing. Many conventional double-layer channel routers have been developed and used to produce optimal solutions for most channel routing problems, the conventional channel routers resolve cyclic constraints between nets and give the track assignment of each net. Thus, double-layer channel routing provides a good starting point for the over-the-cell routing. Obviously, the quality of our over-the-cell routing is dependent on the result of the first step.

In the second step, the transformation is attempted for each entice track. The transformation of tracks always alleviates the routing difficulty in the most congested section of channel routing obtained from the first step. In the third and the fourth steps, every horizontal segment is the operational object for the transformation. Since the transformation of any segment dose not necessarily yield the better results, the segment selection is carefully done in order to choose the best segment for the transformation. If all over-the-cell tracks are occupied while there are still some untransformed segments left in above steps, a real channel is formed in the fifth step by the triple-layer channel router.[16]

Detailed routing algorithm is :

```
for (all channels that net spans) {
 (sx, sy) = Find span of net in channel
 T=Find tracks free for span(sx,sy)

 if (T==null) {
  Net detailed unrouted in this channel ;     continue ; /*
with next channel */
 }
 for (each T) {
  Assignment coat =Ww* segment_wastage        + Wh
* antifuse_wastage
 }
 A sign net to track with minimum coat.
}
```

6. CONCLUSION

A new triple-layer over-the-cell routing algorithm was presented. For channel-less routing, defined virtual channel and real channel to address the over-the-cell routing problem precisely. A salient feature of our over-the-cell routing is that on assumption has to be made on the cell structure. Thus, it can be applied to any compact layout using standard or customized cells.

We have created a new performance driven simultaneous place and route algorithm for FPGAs. We have described our overall strategy for simultaneous place and route within on optimization frame work. We have also described our chosen optimization technique. We then described how the placement and routing are incrementally perturbed to achieve the desired layout meeting the wirability and timing requirements.

References

[1] Meenakshi kaul, Ranga Vemuri, Seriram Govindrajan and Iyad Ouaiss, "An Automated Temporal Partitioning and Loop Fission Approach for FPGA based Reconfigurable Synthesis of DSP Application," Proc. ADC 36th, pp. 616, June, 1998.

[2] Madhukar R. Korupoly, K. K. Lee, D. F. Wong, "Exact Tree-based FPGA Technology Mapping for Logic Blocks with Independent LuTs," Proc. ADC 37th, pp. 708, June, 1999.

[3] Xiaohan Zhu, Bill Lim, "Hardware Compilation for FPGA-based Configurable Computing Machines," Proc. ADC 36th, pp. 697, June, 1998.

[4] Balakrishna Kumthekar, Luca Benini, Enrico Macii, Fabio Somenzi, "In-Place Power Optimization for LuT-based FPGAs," Proc. ADC 37th, pp. 718, June, 1999.

[5] R. J. Francis, J. Rosc and Z. Vranesic, "Techno ogy mapping of look-up table based FPGA for performance," ICCAD, 1991.

[6] J. Cry, Y. Ding, "On Area/Depth trade off in LuT based FPGA Technology Mapping," IEEE. Trans. on VLSI System, 1994.

[7] J. Greene, V. Roy chawdhury, S. Kaptanagln, A. El Gamal, "Segmented Channel Routing," Proc. of 27th ACM/IEEE DAC, 1990.

[8] P. S. Sawkar and D. E. Thomas, "Performance Directed Technology Mapping for Table-look-up based FPGAs," Proc. of 30th DAC, 1993.

[9] C. Sechen and A. Sangioranni-Vincentelli, "The Placement and Routing Package," IEEE J. of Solid-State Circuits, Vol. 20, No. 2, 1985.

[10] S. Broun, J. Rose and Z. Vranesic, "A Detailed Router for FPGAs," Proc. of ICCAD, 1990.

[11] J. Rose, "Parallel lobal Routing for Standard Ce ls," IEEE Trans. on CAD, 1990.

[12] J. Frankle, "Iterative and adaptive slack allocation for performance-driven layout and FPGA Routing," Proc. of DAC, 1992.

[13] J. Cony and C. L. Lic, "Over-the-Cell Channel routing," IEEE Trans. on CAD, Vol. 9, 1990.

[14] H. Yang, D. F. Wong, Edge-Map, "Optimal Performance driven Technology mapping for iterative Lut based FPGA Design," ICCAD, 1994.

[15] L. Chany, P. Hsiav, J. Yan, and P. Shew, "A robust over-the-cell channel router," IEEE Trans. on CAD, 1993.

[16] Supid Kumar Mag, "Performance-Driven Simultaneous Place and Route for FPGAs," Research Report No. CMUCAD, 95-12, 1995.