# FPGA에서 시간구동 최적화의 배치·배선에 관한 연구

김현호°, 이용희*, 이천희**
도립충북과학대학 전자정보과, 신성대학 컴퓨터응용계열*, 청주대학교 전자공학과**
*kimhh@ctech.ac.kr°, lyhkpl@empal.com*, yicheon@chongju.ac.kr**

# A Study on Place and Route of Time Driven Optimization in the FPGA

Hyeonho Kim°, Yonghui Lee*, Cheonhee Yi**
Chungbuk Provincial Univ. of Science & Technology, Shinsung College*, Chongju Univ.**

**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.

## 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]

## 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]

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.[6]

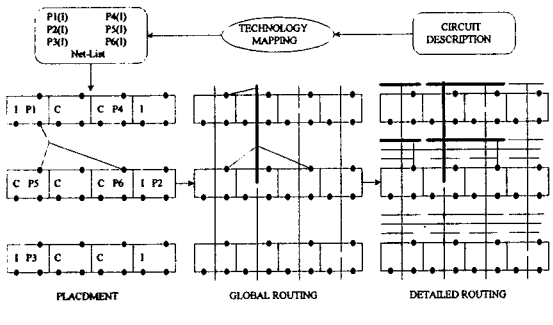


Figure 1. Layout flow for row-based FPGAs.

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.[7] 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.

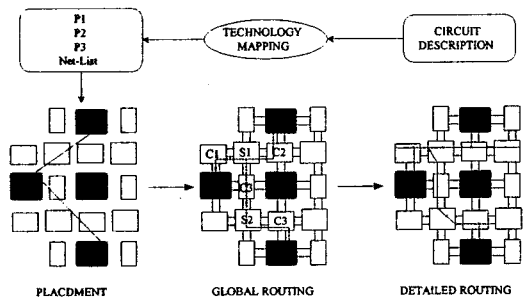


Figure 2. Layout flow for island-style FPGAs.

At the global routing stage, paths are determined for every connection. Paths are 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 for gate-arrays and standard cells like timber wolfe placement and routing package[8] 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 omputed 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).

### 4. Detailed routing

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 chose 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.

```
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.
  }
```

## 5. 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 descrived 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.

## Reference

[1]Meenakshi kaul, Ranga Vemuri, Seriram Govindrajan and Iyad Ouaiss, "Au 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, "Technology mapping of look-up table based FPGA for performance," ICCAD, 1991.

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

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

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