Placement and Placement Driven Technology Mapping for FPGA Synthesis

Tong Gao, Kuang-Chien Chen
Fujitsu America, Inc.
San Jose, CA 95134

Jason Cong, Yuzheng Ding
Computer Science Dept.
University of California
Los Angeles, CA 90024

C. L. Liu
Computer Science Dept.
University of Illinois
Urbana, IL 61801

Abstract: Because of the more restrictive placement and routing constraints in Xilinx FPGA designs, conventional physical design tools for general placement and routing architectures usually do not work well for FPGA designs. Moreover, to generate high quality circuits which are easy to place and route, it is important to consider the specific physical design constraints during the technology mapping process. In this paper, we first present a performance driven placement algorithm specifically developed for the Xilinx FPGAs. We then present a new placement driven technology mapping algorithm which uses placement information to guide the mapping process.

Introduction

For most ASIC designs, it is important to have short turn around times. At the same time, it is important to keep the design cost low. Field programmable Gate Arrays (FPGAs) can be easily programmed and reprogrammed, and therefore, are natural choices for rapid and low cost prototyping. Special FPGA architectures make FPGA designs different from conventional designs, and special logical and physical design tools are needed for FPGA designs.

Two architectures are most commonly used in FPGAs, one based on the programmable cells of small granularity such as the multiplexers, and the other based on lookup-tables which can realize complex functions. Lookup-table based FPGAs, where each lookup-table can realize any function with up to a fixed number of inputs, are particularly interesting because of their flexibilities and challenges posed for the design process. In this paper, we shall specifically deal with Xilinx FPGA architectures. A Xilinx FPGA is called a logic cell array (LCA). An LCA consists of an array of configurable logic blocks (CLBs), a set of configurable input/output blocks (IOBs), and a set of switching matrices which can be programmed for interconnections. A CLB can be programmed to implement one or two lookup-tables with a limited number of inputs. The IOBs can be programmed either as inputs or outputs of the LCA, and they are located at the peripheral of the LCA. Three kinds of routing resources are supplied in the LCA, namely, the general purpose interconnections, the direct interconnections, and the long lines.

The programmable switches for interconnection introduce extra circuitries and extra delays. To reduce the interconnection delays and the size of the LCAs, the programmable switches are designed to be partially programmable (e.g., only a subset of the possible interconnections can be programmed). As a result, relatively more restrictive physical design constraints exist in LCAs in comparison with other architectures. Consequently, conventional physical design tools which consider general placement and routing architectures usually do not work well for Xilinx FPGA, and special physical design tools which consider the specific physical design constraints in the LCAs are needed. Since the interconnections in an LCA is made by programming the switches, the interconnection delays in LCAs also tend to be large. To compensate for the relatively large interconnection delays, it is very important to have placement and routing algorithms which are capable of generating physical designs with small circuit delay. In Section 2, we shall present a performance driven placement algorithm specifically developed for Xilinx FPGAs. Our placement algorithm is capable of generating placements with relatively smaller maximum circuit delay in relatively shorter time in comparison with the placement algorithm used in Xilinx tools.

Since Xilinx FPGAs have relatively more restrictive physical design constraints, a circuit generated by the technology mapping algorithm without considering the possible placement and routing information of the resulting circuit could be hard to place and route, and as a consequence, the final design may have large maximum circuit delay. Therefore, it is important for the technology mapping algorithm to consider the physical design constraints. Unfortunately, most existing technology mapping algorithms for FPGAs are not capable of doing so. Even though some performance-driven technology mapping algorithms exist [2,4], most of them try to generate circuits with minimum number of levels of CLBs. The circuits produced by these technology mapping algorithms will have small maximum circuit delay under the assumption that all the nets have the same delay after placement and routing. However, this is almost never the case in reality. Since the routing is mostly decided by the placement, it is most important to consider the placement information during the technology mapping process. In [5], a layout driven technology mapping algorithm

was presented. In Section 3, we shall present a new technology mapping algorithm which uses the placement information to guide the technology mapping process.

Both our placement and placement driven technology mapping algorithms were implemented for Xilinx 3000 series FPGAs. The reason for us to choose the 3000 series FPGAs is that it is hard to implement our own routing algorithm without all the information about Xilinx FPGA architectures, and to complete a design, we need to use Xilinx routing algorithm to route the circuit generated by our algorithms. The routing algorithm for the 3000 series FPGAs can be easily used on the results generated by our algorithms, while it is impossible to do the same thing with the routing algorithm for the 4000 series FPGAs because it is integrated with the technology mapping and the placement algorithms in the tools for the 4000 series FPGAs. Our technology mapping and placement algorithms can be easily modified to work for Xilinx 4000 series FPGAs.

Performance Driven Placement Algorithm

Simulated annealing algorithm is used by Xilinx to place circuits in the 3000 series FPGAs. Even though the simulated annealing algorithm is known to generate high quality placements, its long running time contradicts with the short turn around time advantage of the FPGA designs. In order to have a fast placement algorithm which is capable of generating placements with small maximum circuit delay, we developed a min-cut based performance driven placement algorithm for the Xilinx FPGAs.

Our placement algorithm is based on the algorithm presented in [1]. A convex programming problem is first formulated to compute a set of upper-bounds on the net wire lengths according to the time requirements. A min-cut based placement algorithm is then used to place the CLBs and IOBs under the guidance of the upper-bounds. Even though the algorithm described in [1] is capable of generating placements with small maximum circuit delay in reasonably short time, it is only capable of generating placements for gate-array circuits. Since the possible IOB locations (IOB slots) and the possible CLB locations (CLB slots) in an LCA are very specific, the algorithm in [1] needs to be modified to handle different possible distributions of the IOBs and the CLBs in different FPGA chips. In our algorithm, a list of IOB slots and a list of CLB slots is kept for each region to be partitioned, and the IOB slots and the CLB slots can be distributed in arbitrary fashion.

In [1], it is assumed that the IOBs have pre-determined locations. This early commitment of IOBs to IOB slots with no information about the possible placement of CLBs may lead to poor placements. In our algorithm, the IOBs are placed together with the CLBs. Our algorithm can also accept IOBs and CLBs with pre-determined locations. Since the distributions of the CLB slots and the IOB slots are different in different regions, during the partitioning process, two separate gain lists are maintained for the IOBs and the CLBs in each region, and the best IOB or CLB to move is selected among these gain lists under the guidance of the balancing rules for the IOBs and CLBs.

Balancing rules are the rules that the placement algorithm uses to control the balance between the number of CLBs (IOBs) and the number of CLB slots (IOB slots) in each region. The quality of the placement generated depends heavily on the balancing rules. Overly restrictive balancing rules will restrict the exploration of possible placement solutions and lead to poor placement results, and overly loose balancing rules may lead to the generation of unevenly distributed and possibly congested regions that are hard to place and route later on. Therefore, it is very important to have balancing rules that lead the placement algorithm to generate placements with evenly distributed regions while still not overly restricting the partitioning process. There is one exception to above criteria for balancing rules. If the number of IOBs and/or CLBs in the circuit is much smaller than the number of corresponding slots, distributing the IOBs and/or CLBs evenly among different regions will lead to an overly sparse placement which will have large net wire lengths and large maximum circuit delay. In such case, controlled unevenly distributed regions should be allowed.

In our algorithm, similar balancing rules are used for CLBs and IOBs. Here, we shall only describe the balancing rules for the CLBs. For a region r, let *CLB balance factor* $B_r = C_r/S_r$, where C_r is the number of CLBs in r and S_r is the number of CLB slots in r. After r is initially partitioned into r_1 and r_2, the maximum allowable CLB balance factor β_{r_1} for r_1 is computed as follows:

```
1. β_r1 = (B_r + 1)/2.0;
2. if (C_r1 + 1 > β_r1 S_r1)
3.   {
4.       β_r1 = (C_r1 + 1)/S_r1;
5.       if (β_r1 > 1)
6.         β_r1 = 1;
7.   }
8. if (β_r1 < 0.7)
9.   β_r1 = 0.7;
```

The maximum allowable CLB balance factor β_{r_2} for r_2 is computed in a similar way. To distribute CLBs evenly among r_1 and r_2, it is desirable to have β_{r_1} and β_{r_2} as close as possible. However, in order not to overly restrict the partitioning process, we should give the partitioning algorithm a certain amount of leeways during partitioning. Therefore, we compute the initial value of β_{r_1} as in line 1. Line 2 checks whether the value of β_{r_1} computed forbids the movement of any CLB into r_1. If that is the case, line 4 increases the value of β_{r_1} to allow one CLB to be moved into r_1. Lines 5 and 6 make sure that β_{r_1} computed in line 4 does not exceed 1 so that the number of CLBs in r_1 does not exceed the corresponding number of CLB slots in r_1. In the case that r was sparse, lines 8 and 9 increase the value of

β_{r_1} to 0.7 to avoid the generation of overly sparse placement.

Placement Driven Technology Mapping

As discussed in [2], simple gate circuits in which each gate has only two inputs is a good starting point for technology mapping process. We shall call the circuit before technology mapping *the initial circuit*, and the circuit after technology mapping the final circuit. As mentioned earlier, the technology mapping algorithm should consider as much placement information of the final circuit as possible. Since the final circuit is not known until the end of the technology mapping process, it is impossible to obtain the precise placement information of the final circuit before or during the technology mapping process. However, if the technology mapping algorithm which maps the simple gates locally (e.g., the algorithm tries to map adjacent gates into the same CLB) in the initial circuit, the placement topology of the initial circuit will reflect the placement topology of the final circuit. Therefore, the placement information of the initial circuit can be used to approximate the placement information of the final circuit. In our algorithm, the performance driven placement algorithm described in Section 2 is first used to place the initial simple gate circuit. The placement information is then extracted from the placement generated, and a modified FlowMap [3] algorithm is used to map the simple gate circuit under the guidance of the placement information.

Before the simple gate circuit can be placed, an artificial LCA with a set of IOBs and CLBs is generated to hold the simple gates and the primary *IO* pins. The number of CLBs (simple gates) in the initial circuit is much larger than the number of CLBs in the final design, while the number of IOBs (primary *IO* pins) in the initial circuit is the same as the number of IOBs in the final design. To make the placement topology of the initial circuit close to the placement topology of the final circuit, the same number of IOB slots are first assigned to the same locations in the artificial LCA as in the LCA used for the final design. The number of CLB slots needed to hold the simple gates are then computed and the CLB slots are distributed in a similar way as in the LCA used for the final design. To make the delay information of the simple gate circuit reflect the delay information of the final circuit as much as possible, the delay of the IOBs and the delay per unit length of interconnecting wire are set to be the same as the corresponding values in the LCA used for the final design. Since several simple gates may be mapped into one CLB by technology mapping, the CLB delay in the artificial LCA is set to be a fraction of the CLB delay in the LCA used for the final design.

After the placement, interconnection delay information of the placement can be easily extracted, and the delay information is used to guide the technology mapping process. To map the simple gate circuit under the guidance of the delay information, a modified FlowMap algorithm was developed which guarantees a minimum-delay mapping solutions under any given net delay estimation [3]. During the technology mapping process, to simplify the computation of the interconnection delays, it is assumed that the delay of a net which is completely mapped into a CLB (e.g., all the simple gates that the net interconnects are mapped into the same CLB) becomes 0, and the delay of a net which is not completely mapped into a CLB remains unchanged. The objective of the modified FlowMap algorithm is to minimize the maximum delay from the primary input pins to the primary output pins in the final circuit. Under the assumption that the placement topology of the simple gate circuit reflects the placement topology of the final circuit, the modified FlowMap algorithm generates a circuit with the minimum maximum circuit delay.

One side effect of using the modified FlowMap algorithm is that the number of CLBs in the final circuits might be significantly larger than the number of CLBs in the circuits generated by the original FlowMap algorithm [2]. The increase in the number of CLBs are mainly caused by the heuristics used in both FlowMap algorithms during the postprocessing process for minimizing the number of CLBs. In the FlowMap algorithms, a cut set that gives the minimum delay and the maximum volume (number of simple gates) is chosen to be mapped into an CLB to minimize the delay and the number of CLBs. In the unit delay model [2], because all the nets have the same delay, there are many minimum delay cuts and it is easy to find a cut that packs large number of simple gates. However, in the non-unit delay model [3], there are few minimum delay cuts and they often pack fewer simple gates. Since the CLBs generated in the non-unit delay model contain fewer simple gates, more CLBs are needed to implement the circuit. More CLBs usually leads to more CLB delays, more and longer interconnections, and larger maximum circuit delay. The increases in the number of CLBs and interconnections also cause the resulting circuit to be hard to place and route. In order to reduce the number of CLBs in the circuits generated by the modified technology mapping algorithm, we can round off the net delays to reduce the possible number of different net delay values. By rounding off the net delays, we are trading the accuracy of delay information with the number of CLBs. In [3], a relaxation technique was also introduced to reduce the number of CLBs by relaxing the minimum delay requirement.

Experimental Results

Our placement and technology mapping algorithms were implemented in C and integrated into Berkeley logic synthesis tool *sis*. The algorithms were tested on 12 combinational benchmark circuits which are summarized in the second and third columns of Table 1.

To compare our placement algorithm with Xilinx placement algorithm, the FlowMap algorithm [2] is first used to

map the simple gate circuits. The resulting circuits are summarized in the last three columns of Table 1. Xilinx placement and routing algorithms are then used on the resulting circuits, and the final designs obtained are summarized in the second and third columns of Table 2. Our placement algorithm and Xilinx routing algorithm are then run on the same set of mapped circuits, and the final designs obtained are summarized in the last two columns of Table 2. On average, we were able to get about 7.5 percent improvement in the maximum circuit delay by using our placement algorithm comparing with the results obtained using Xilinx placement algorithm. Our algorithm is also about 46 percent faster than Xilinx placement algorithm.

circuit name	simple gate circuits		mapped circuits		
	no. of gates	no. of nets	no. of CLBs	no. of nets	no. of levels
cordic	68	91	17	40	4
count	129	164	34	69	5
bw	156	161	28	33	1
f51m	157	165	42	59	7
frg1	167	195	69	88	6
comp	204	236	68	199	7
my_add	208	241	24	57	8
9sym	274	283	96	105	5
term1	326	360	117	151	5
C499	398	439	70	111	4
C880	418	478	123	183	9
alu2	492	502	216	226	10

Table 1. Circuit Specifications.

circuit name	Xilinx placement		our placement	
	maximum delay(ns)	placement time(min.)	maximum delay(ns)	placement time(min.)
cordic	41.3	1.3	35.9	0.8
count	70.0	4.7	56.5	1.7
bw	22.3	3.0	22.4	0.8
f51m	82.9	1.8	82.8	2.0
frg1	70.1	3.4	72.1	3.0
comp	88.0	7.7	78.9	3.1
my_add	96.1	2.5	90.5	1.4
9sym	81.3	21.5	76.6	6.5
term1	79.1	24.3	74.3	9.0
C499	97.3	9.5	82.8	6.4
C880	136.7	28.5	117.3	16.8
alu2	206.8	91.1	199.1	30.9

Table 2. Placement Results after Routing.

To check the quality of the placement driven technology mapping algorithm, the mapping algorithm is run on the same set of simple gate circuits specified in the second and third columns of Table 1. Table 3 compares the final results generated by Xilinx tools and the results generated by using our placement and technology mapping algorithms and Xilinx routing algorithms. As was mentioned in Section 3, the modified FlowMap algorithm tends to generate more CLBs comparing with the FlowMap algorithm using unit delay model. To reduce the number of CLBs, the circuits generated by our technology mapping algorithm were obtained by allowing 4 different net delay values for all the nets. Notice that even though the circuits generated by our technology mapping algorithm still have more CLBs than the circuits generated by the FlowMap algorithm using unit delay model, in most case, we were still able to obtain final designs with smaller maximum circuit delays comparing with the results obtained by Xilinx tools. This is because the circuits generated by our technology mapping algorithm are easier to place and route. The incompletion of the last two designs using our algorithms are caused by the large number of CLBs generated in the two circuits.

circuit name	Xilinx tools			our mapping algorithm		
	no. of CLBs	no. of nets	max. delay	no. of CLBs	no. of nets	max. delay
cordic	17	40	41.3	23	46	39.6
count	34	69	70.0	34	69	62.5
bw	28	33	22.3	28	33	21.1
f51m	42	59	82.9	53	61	88.4
frg1	69	88	70.1	58	86	67.1
comp	68	199	88.0	76	98	91.8
my_add	24	57	96.1	30	66	87.7
9sym	96	105	81.3	97	106	81.1
term1	117	151	79.1	117	151	84.7
C499	70	111	97.3	77	118	83.1
C880	123	183	136.7	140	200	-
alu2	216	226	206.8	247	257	-

"-" means there are too many CLBs to be fitted onto the FPGA chip or there are unrouted pins.

Table 3. Placement Driven Technology Mapping.

Conclusion

In this paper, we discussed the importance of considering physical design information during the technology mapping process. We then present a min-cut based performance driven placement algorithm and a placement driven technology mapping algorithm. The experimental results are encouraging.

References

[1] T. Gao, P. M. Vaidya, and C. L. Liu, "A New Performance Driven Placement Algorithm," in Proc. ICCAD, 1991, pp. 44-47.

[2] J. Cong and Y. Ding, "An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs," in Proc. ICCAD, 1992, pp. 48-53.

[3] J. Cong, Y. Ding, T. Gao, and K. Chen, "An Optimal Performance-Driven Technology Mapping Algorithm for LUT based FPGAs under Arbitrary Net_Delay Models," in Proc. CAD & CG, 1993, in press.

[4] R. J. Franics, J. Rose, and Z. Vranesic, "Technology Mapping for Lookup Table-Based FPGAs for Performance," in Proc. ICCAD, 1991, pp. 568-571.

[5] M. Pedram and N. Bhat, "Layout Driven Technology Mapping," in Proc. 28th DAC, 1991, pp. 99-105.