How Much Can Logic Perturbation Help from Netlist to Final Routing for FPGAs

Catherine L. Zhou Wai-Chung Tang Wing-Hang Lo Yu-Liang Wu
Department of Computer Science and Engineering
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong
{lzhou, wctang, whlo, ylw}@cse.cuhk.edu.hk

ABSTRACT
One unique property of an FPGA chip is that any logic perturbation inside its Look-Up-Tables (LUTs) is totally area/delay-free. Amongst others, this free LUT-internal resource perturbation can also be used to trade for critical LUT-external logic/wire removals for EDA improvements, an extra flexibility ignored before. Using rewiring technique for such logic perturbations, we show that significant cut-downs upon already excellent results from the state-of-the-art DAOmap mappings and the TVPR place-and-route can still be obtained. This logic perturbation operation can further reduce the number of LUTs by up to 33.7% (avg. 10%) without delay penalty and also reduce critical path delay by up to 31.7% (avg. 11%) without disturbing placement or sacrificing area in the final routing. For delay reduction, under proper rewiring strategy, the CPU time used by rewiring is only 5% of the total run time consumed by TVPR's placement and routing. This idea of perturbing logic between the free LUT-internal and critical LUT-external circuit resources is simple and proved to be powerful. The encouraging results suggest a new technique for an optimization domain less explored for FPGA design flow.

Categories and Subject Descriptors
B7.2 [**Integrated Circuits**]: Design Aids – *Layout, Placement and routing.*

General Terms
Design, Experimentation, Performance

Keywords
ATPG, FPGA, Routing, Technology mapping

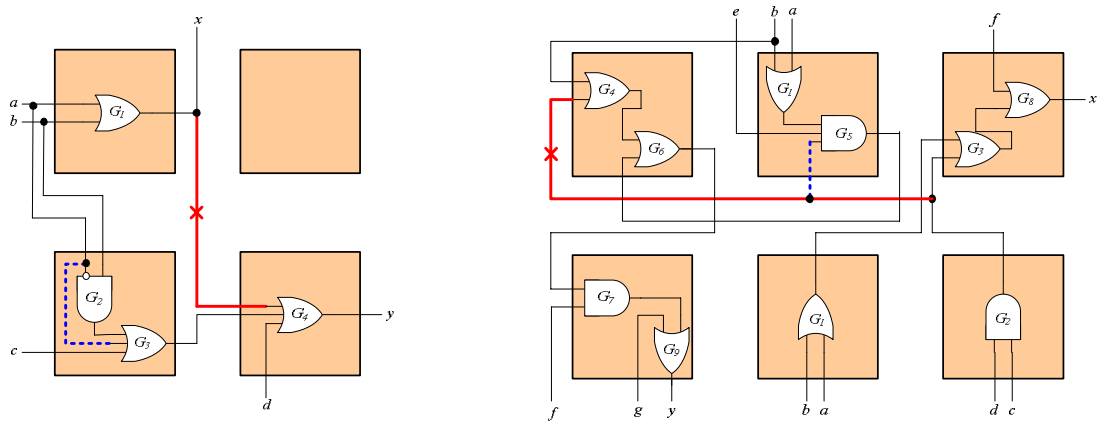
1. INTRODUCTION
The basic idea of logic perturbations (rewiring) is to replace a wire/gate with other wires/gates without changing the logic function of the circuit. Applying rewiring for a circuit netlist can incrementally refine a circuit's structure based on its logical and physical information to improve on many EDA objectives, ranging from circuit area to routability and performance. The known rewiring techniques can be classified into three groups: the Automatic Test Pattern Generation (ATPG) based rewiring method [4] [5] [7] [8] [9] [10] [13] [14], the Set of Pairs of Functions to be Distinguished (SPFD) based rewiring method [12] [18] [20], and the Graph-Based Alternative Wiring (GBAW) method [11] [19].

In [6], the ATPG-based rewiring method was firstly applied to improve FPGA routability. After placement, rewiring is used to find alternative wires for all nets, and then lower routing order priorities are assigned to nets with more alternative wires. During a routing, nets with high priorities are processed first, and nets not routable will be replaced by their alternative wires. Experiments are carried out on two circuits with AT&T ORCA router. These two circuits are not routable in the original ORCA router; however, with the aid of rewiring they are routed successfully. This is the first known work applying rewiring to help complete FPGA routings.

Another approach is the SPFD-based post-layout logic synthesis [12]. In this work, Quartus (Version II 1.0) is used as the placement and routing tool. Based on placement information, net delays are estimated to find ε-critical paths, which are the paths whose delay is larger than $(1 - \varepsilon)D$, where D is the largest path delay and $\varepsilon < 1$. Then SPFD-based rewiring is used to process the nets on ε-critical paths. After transformations, the routing is executed on the new netlist. The application of SPFD rewiring brings an average reduction of 5.1% on critical path delay, but the approach suffers from its quite slow runtime. As the percentage of nets actually transformed is not shown in the paper, it is not clear on the effectiveness produced per their rewiring operation.

In the conventional EDA flow for FPGAs, starting from a given netlist, a technology mapping result is generated first; then, a place-and-route tool is used to produce the final routing. Most mapping algorithms treat the circuit as a pure graph, and improve the routability and performance by applying different heuristics on depth-optimal graph mapping solutions without considering the underlying logic information. Besides the two works mentioned above, not much further effort was done for applying logic perturbations on FPGA routings, and there has been no work investigating how much the room can be if the rewiring technique is integrated inside every step of the whole FPGA EDA flow, starting from the netlist to the final routing.

DAC 2007, June 4–8, 2007, San Diego, California, USA.



(a) (b)

Figure 1. Logic shifting of (a) (critical) LUT-external to (free) LUT-internal (b) (long) LUT-external to (short) LUT-external

The optimality studies on logic synthesis and technology mapping [15] show that by utilizing the logic information, one can improve beyond the optimization limit imposed by DAG-based technology mapping algorithms. Nonetheless, as a *K*-input LUT is able to implement any *K*-variable function, any logic perturbation inside a LUT is *virtually free* of any extra area or delay cost on FPGA chips. In this paper, we will apply the rewiring logic perturbation operations to explore the effectiveness of logic shifting between critical and non-critical areas as well as trading (LUT-internal) free resource for critical LUT-external logic/wire resources to improve from mappings to final routings and to examine the existence of their correlations. Figure 1 illustrates how postlayout FPGA logic perturbations (logic shifting) can be used to affects the performance of a circuit. In Figure 1(a), the target net $G_1 \rightarrow G_4$ is replaced by an internal (the source gate and sink gate are in the same LUT) wire; Figure 1(b) shows that a longer net $G_2 \rightarrow G_4$ is substituted by a shorter net $G_2 \rightarrow G_5$. In both cases, routing becomes easier, and net delay can be reduced.

We apply our logic perturbations on the following two parts:

(1) Technology mapping for area reduction. The input circuit is first perturbed to increase the mapping area using a series of transformations. This enables the optimization process to jump out of the local optimums. Then the circuit is perturbed for a smaller mapping area.

(2) Final routing for delay reduction. Physical layout and logical information are considered together to make decisions in picking rewiring transformations. Alternative wires are selected under a greedy manner that no extra LUT is introduced and the original placement is kept. A cost function is used to ensure that a transformation will reduce the FPGA delay. Our final results show that an efficient scheme can obtain a good trade-off for low CPU run time and significant improvements.

A state-of-the-art technology mapping algorithm DAOmap [3] and the most powerful academic FPGA place and route tool TVPR [1] [2] [17] are experimented for further improvements. With our proposed rewiring operations, we can reduce the number of LUTs by up to 33.7% (avg. 10%) without delay penalty compared to the best results produced by DAOmap. For routing improvement, we apply rewiring upon the well-known TVPR tool, which has been used by more than 1000 universities and companies since its introduction in 1997. Its placement is based on the simulated annealing (SA) algorithm, and its routing is based on the Pathfinder negotiated congestion algorithm. By iteratively trying different routing paths, it can finally produce quite high-quality routing results that are hardly improvable by any other known place-and-route tool. However, under an effective rewiring strategy, it is encouraging that the critical path delay can still be further reduced by up to 31.7% (avg. 11%) without disturbing placement or sacrificing area in our final routing. Besides, our CPU usage is particularly low, compared to [12].

A few findings of our experiments can be drawn as follows.

(1) The largest room for area improvement locates in the technology mapping step. With area reductions, delays can also be reduced slightly.

(2) However, a technology mapping with too high logic density (more saturated usage on logic pins) may adversely yield a routing requiring some extra routing tracks (one extra track on four out of fourteen benchmarks). That is, a best mapping result does not necessarily imply a best final routing.

(3) Performing logic perturbation in routing stage is clearly more effective for delay improvements due to the more accurate timing information that can only be precisely known after routing. Nonetheless, routing tracks (W) can still be cut (in one out of twelve benchmarks). And mostly, delay can be consistently reduced through this rewiring operation in routing step without area penalty.

(4) As proved in our experiments, applying this rewiring operation on the whole flow can be a fruitful approach without much CPU overhead.

The remainder of this paper is organized as follows. Section 2 briefly introduces the ATPG-based rewiring techniques. Section 3 and Section 4 describe the logic perturbation techniques for technology mapping and routing in details. Experimental results are shown in Section 5. Section 6 summarizes the contributions of our work.

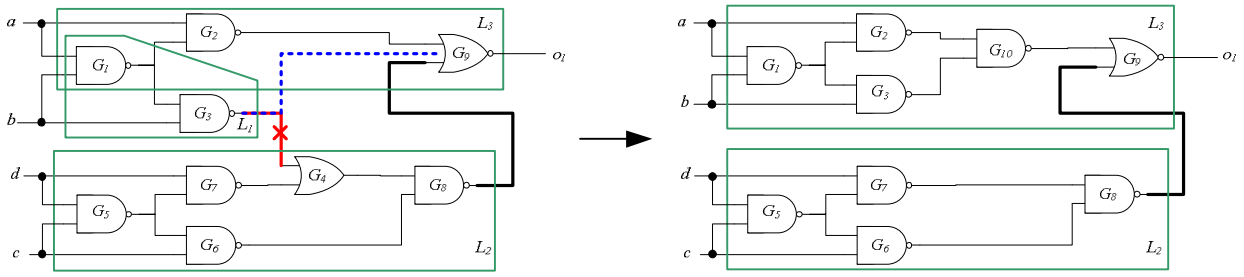


Figure 3. The number of LUTs is reduced from 3 to 2 through exchanging External Resource with Free Internal Resource using Rewiring ($K = 3$)

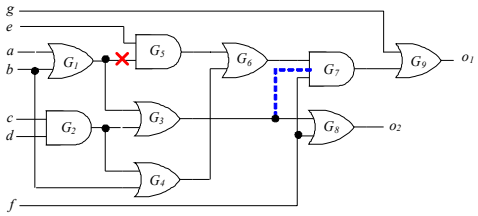


Figure 2. Rewiring on Boolean network

2. REWIRING TECHNIQUES

The basic idea of the ATPG-based rewiring technique is to add a redundant wire/gate to make other wires/gates redundant and removable. A wire/gate is *redundant* if its addition or removal does not change the logic function of a Boolean network [7]. For the example shown in Figure 2, when the redundant wire $G_3 \rightarrow G_7$ is added to the circuit, the wire $G_1 \rightarrow G_5$ will become redundant thus removable.

3. REWIRING-BASED TECHNOLOGY MAPPING

Any logic perturbation inside an LUT is completely free and it is possible that through perturbation we can trade free internal resources for valuable external resources on the FPGA architecture. Such an example for technology mapping is shown in Figure 3, where the external wires are in bold for better illustration. The initial circuit is mapped into three LUTs by DAOMap. The result is optimal if the structure of the circuit cannot be changed. However, using rewiring technique, we identify an alternative wire $G_3 \rightarrow G_9$ for the external wire $G_3 \rightarrow G_4$. Since the alternative wire is internal, we can further reduce the area with the free logic transformation inside an LUT. The modified circuit can now be mapped with two LUTs only. In this work, we propose a rewiring-based area optimization scheme to perturb the circuit so that there are different internal/external resource relocations in the technology mapping.

Area minimization with depth constraint in the technology mapping problem is known to be NP-hard. In most technology mapping algorithms, the initial circuit is kept unchanged and techniques like cut enumeration will try different mappings of the gates in order to minimize the area. Nevertheless, all useful logic information, which allows transformations on the network, is ignored throughout. In this work, we try to perturb the input circuit with logic transformation and evaluate the effect on the final mapping area, and greedily take the transformations which can reduce the mapping area.

Every wire in the input circuit will be used as a target wire. We then use rewiring techniques to find alternative wires for the target wires. Each target wire and alternative wire pair is considered as a transformation. All transformations will be ranked using our area efficiency heuristic, which will be explained later in this section. We take the transformations one by one and evaluate the transformed circuit with a pseudo technology mapping. If the mapping area is reduced, the transformation will be taken; otherwise, it will be unrolled and the next transformation will be tried. The whole process will be terminated after a certain number of sorted transformations found futile in reducing mapping area.

This greedy approach clearly raises the ordering issue of target wires. An efficient ordering of the target wires will allow us to search for area-reducing transformations more quickly. We use the idea of area efficiency (AE) proposed in IMap [16] for making efficient ranking.

The area flow at a node is given by Equation (1).

$$af(v) = A_v + \sum_{u \in input(v)} af(u) \qquad (1)$$

Given a wire $w_t = (u, v)$, we define the area flow at wire as the difference of the area flows of the source node u and sink node v, i.e., $af(w_t) = af(u) - af(v)$. To persist a better wire ranking, for each transformation (w_t, w_a), we again compute the different $af(w_a) - af(w_t)$, which is taken as a score to rank all transformations we found from the rewiring algorithm. As the target wire $w_t = (u, v)$ is removed, its area flow $af(w_t)$ will be re-distributed to the fanout of u and it is desirable to remove a wire with smaller area flow. On the other hand, when we add a new wire $w_a = (p, q)$ to the network, the area flow on the node p will be distributed to the new wire as well. This encourages us to add a new wire with higher area flow and in consequence, a transformation with a high score should be used for area reduction earlier in the optimization process.

We would like to emphasize that the heuristic ranking plays an important role in runtime reduction since we are using a greedy approach in searching for useful transformations. Furthermore, our optimization process maintains the mapping depth by checking the depth constraint at every node. If the transformation is going to increase the mapping depth of the circuit, it would not be accepted and we will proceed to a next transformation. This can prevent the delay performance from worsening in the routing phase due to the area optimization.

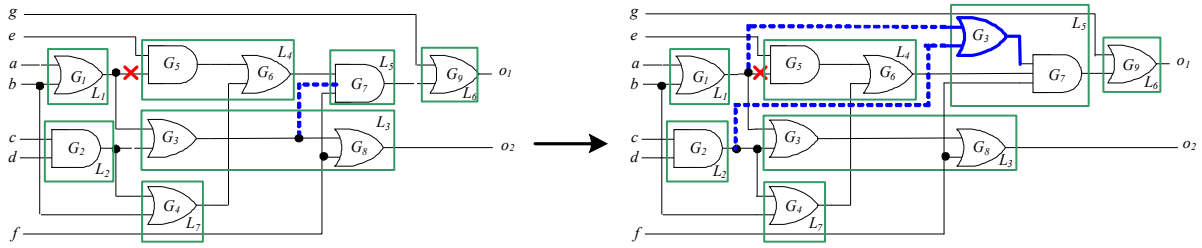


Figure 4. Destination LUT expansion ($K = 4$)

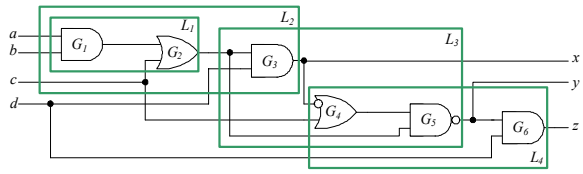


Figure 5. Example of minimum TFI cone

4. REWIRING-BASED ROUTING

The main target of a rewiring-based routing is to improve FPGA delay performance. By performing logic perturbations on the mapped circuit, we can replace long nets with shorter alternative nets or internal wires. Thus, a critical path delay is likely to be reduced.

When an alternative wire is added to the mapped circuit, new LUTs may be required to maintain the logic equivalence. For example, in Figure 4, $G_3 \rightarrow G_7$ is added to replace $G_1 \rightarrow G_5$. However, as G_3 is not the output node (root) of L_3, a new LUT with G_3 being the output node will be generated, as suggested by [6], which may not be feasible as there might be no available space for the added LUT. To apply this type of alternative wires, we propose the destination LUT expansion method. The alternative wire ($u \rightarrow v$) satisfying the following condition will be chosen: u is neither a PI nor the root of a LUT. Given that M_1 is the input set of u's Transitive Fanin (TFI) cone inside the LUT containing u, and M_2 is the input set of the LUT containing v, $|M_1 + M_2| \leq K$ (maximum input pin number of a LUT). For example, in Figure 5, the TFI cone of G_6 inside the LUT containing G_6 only covers G_4, G_5, and G_6. Given an alternative wire $u \rightarrow v$, if $|M_1 + M_2| \leq K$, we can then duplicate the whole logic producing u, with the input set M_1, inside the LUT containing v. Thus, we do not have to introduce an extra LUT. This process is called *expansion*.

For example, in Figure 4, $G_3 \rightarrow G_7$ is to be added to make $G_1 \rightarrow G_5$ redundant and removable. Considering $M_1 = \{G_1, G_2\}$, $M_2 = \{G_6, f\}$, $K = 4$, and $|M_1 + M_2| = K$, we can then expand L_5 by connecting M_1 (G_1 and G_2) to the duplicated logic G_3 inside L_5. Thus, the transformation is completed by updating the mapping of L_5 with the connection of two new wires without any LUT addition. As shown in Figure 5, after technology mapping, some gates may be duplicated inside several LUTs, therefore a gate can have more than one TFI cone. Obviously, if this gate is the source node u of the alternative wire $u \rightarrow v$, then choosing this gate's smallest related input set, the *minimum TFI cone*, might increase the chance of successfully expanding all LUTs containing v. For example, in Figure 5, G_4 is duplicated in L_3 and L_4. Its TFI cone in L_3, $Cone_3$, contains G_3 and G_4 with input set $\{G_2, c, d\}$. Whereas the TFI cone of G_4 in L_4, $Cone_4$, only contains G_4 with input set $\{G_3, c\}$. So the minimum TFI cone of G_4 is $\{G_3, c\}$. When an alternative wire starting from G_4 is to be added, G_3 and c will be connected to all LUTs containing its sink node.

Before performing a transformation, we take two steps to determine if the alternative wire can be used: mapping depth checking and cost function evaluation. We use Equation (2) [1] [2] to evaluate the chosen candidate. This cost function reflects the cost contribution from the netlist by exploring its bounding box inside the placement. If it costs more than the target net, it will be discarded; otherwise, the transformation will be performed.

$$Cost = \sum_{i=1}^{N_{nets}} q(i)\left[\frac{bb_x(i)}{C_{av,x}(i)^{\beta}} + \frac{bb_y(i)}{C_{av,y}(i)^{\beta}}\right] \quad (2)$$

5. EXPERIMENTAL RESULTS

As illustrated in Figure 6(b), we conduct experiments on the following three flows with rewiring injected differently to find out the respective effectiveness margins. (1) DAOmap → Rewiring → TVPR (2) DAOmap → TVPR → Rewiring (3) DAOmap → Rewiring → TVPR → Rewiring.

The logic perturbation method is implemented in the C language. The experimental platform is a 3.2GHz Linux machine with 1GB memory. All the benchmark circuits are mapped into 4-input LUTs, and each CLB contains one LUT.

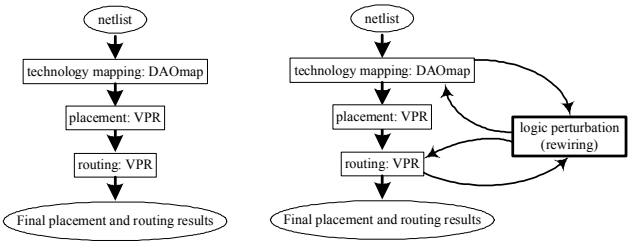


Figure 6. (a) Original FPGA CAD flow; (b) Performing logic perturbation on technology mapping and routing

5.1 Rewiring for Technology Mapping only

Table 1 shows the effects of the flow (1) DAOmap → Rewiring → TVPR. This approach provides a reduction upon the DAOmap mapping results of nearly 10% in LUTs.

For some benchmark circuits, as some LUTs are removed, fewer nets and shorter critical paths can cause direct delay reduction. While for some circuits in which no LUT is removed from the critical paths, transformations outside may also cause a new topology requiring even longer new critical path after placement, which is why these circuits get slight delay penalty along with area reduction. Another reason for delay increase is that when the

number of LUTs in a circuit is reduced a lot, the FPGA architecture may become much tighter. As the channel width is not raised, the high channel density makes some nets take longer routing paths.

The above analysis reveals that logic perturbation in technology mapping is an efficient way to reduce FPGA area by removing LUTs, but does not promise delay performance improvement because of the lack of accurate layout information at this stage.

5.2 Rewiring for Routing only

Table 2 shows the effects of the flow (2) DAOmap → TVPR → Rewiring. Column 2-4 show that 3.7% of all nets are replaced by their alternative wires for routing improvement. Although rewiring can find much more alternative wires according to [6], only a small part of them are useful in delay reduction. Column 8-10 are the comparison results of critical path delay. At the same time the comparison results of channel width are included in Column 5-7. The channel width of C1908 is reduced by one after seven transformations. We do not include it in delay comparison because the delay of a circuit is very likely increased if the circuit is routed with a smaller channel width. The average delay reduction is more than 10%. From Column 11–13, the CPU time consumed by rewiring is only 5% of the total time for TVPR's placement and routing, which is much faster than the SPFD approach. Because we have different starting set up from SPFD rewiring, we cannot make a direct comparison.

To the best of our knowledge, this is the first work giving quantitative analysis on the power of the ATPG-based rewiring techniques when applied in LUT-based FPGA routing. This part of work reveals that rewiring is powerful in delay reduction, especially under very low CPU overhead and without area penalty. Considering its high efficiency in area reduction in technology mapping, we believe that rewiring is a strong tool for postlayout logic synthesis to improve FPGA performance and routability. Most importantly, it is known that any effective delay reduction scheme is relying on the accuracy of physical layout information, which in not available until a routing is completed. That is why the delay performance cannot be improved in rewiring-based technology mapping according to the experiments in Section 5.1.

5.3 Rewiring for both Technology Mapping and Routing

Table 3 reflects the results for applying flow (3) DAOmap → Rewiring → TVPR → Rewiring. It shows that applying rewiring on both stages, though reduces LUTs by 10% too and reduces (routing) area by 3% but the delay reduction in only 3.8%, which is worse then the flow (2). As most FPGA chips do not down-scale sizes continuously, LUT reductions do not always bring routing area reductions proportionally (e.g. 10 % LUT reduction only brings 3.8% routing area reduction). This result also implies an anomaly point: *it is not necessarily true that a best technology mapping always yields a best final routing result*. Therefore, we need an EDA flow with more stages integrated together and a powerful logic perturbation tool to shift optimization resources between them for a globally best final solution.

6. CONCLUSIONS

Area and delay are the two core issues for FPGA designs. However, the area optimization is mainly attributed to the technology mapping stage while the delay can only be correctly handled in the final routing stage. Optimizing both simultaneously has always imposed a tough challenge to us. In this paper, we further show that in a conventional EDA flow divided into several stages, a best result obtained in a certain stage according to its cost function may not necessarily be the best for later stages. In today's commonly adopted FPGA design flows, a technology mapping result with a few less LUTs may adversely yield a routing with one or more tracks. As a result, it may be useful to have a design flow being able to shift optimization resources across boundaries between different stages and a universal technique applicable to all stages would be worthwhile to develop. As rewiring is a both physical- and logical-information sensitive transformation technique that can be universally adaptable to nearly most EDA stages, it makes a good sense for us to design a flow with rewiring integrated into all stages, from netlist to final routing, and analyze its impact margins on the various stages.

As a first known effort of this kind, our experimental results show that the rewiring logic perturbation can still bring large improvements on area and delay simultaneously, under acceptable CPU overhead and no penalty of other objectives. Compared with the already excellent DAOmap+TVPR results, we can reduce the number of LUTs by up to 33.7% (avg. 10%) and critical path delay by up to 31.7% (avg. 11%), which is a result with practical significance too. In the future, we would like to improve the speed of the rewiring engine and further extend the flow to allow for more resource shifting flexibility between different stages. And as a longer term goal, to investigate a new flow with all stages merged together under the help of rewiring technique. According to our current experimental results, this direction seems promising.

7. ACKNOWLEDGEMENT

This work was partially supported by Hong Kong RGC Grant Ref. No. 2150500, RGC CUHK Direct Grant 2050351, and NSFC 90607001.

8. REFERENCES

[1] V. Betz and J. Rose, "VPR: a New Packing, Placement and Routing Tool for FPGA Research," in *Proc. Int'l Workshop on Field Programmable Logic and Applications*, London, UK, Sept. 1997, pp. 213-222.

[2] V. Betz, J. Rose, and A. Marquardt, *Architecture and CAD for Deep-submicron FPGAs*. Boston, USA: Kluwer Academic, 1999.

[3] D. Chan and J. Cong, "DAOmap: a Depth-optimal Area Optimization Mapping Algorithm for FPGA Designs," in *Proc. Int'l Conf. on Computer-aided Design,* 2004, pp. 752-759.

[4] C. W. Chang and M. Marek-Sadowska, "Single-Pass Redundancy Addition and Removal," in *Proc. IEEE/ACM International Conference on Computer-aided Design'01*, San Jose, CA, USA, Nov. 2001, pp. 606-609.

[5] C. W. Chang and M. Marek-Sadowska, "Who Are the Alternative Wires in Your Neighborhood?" in *Proc. ACM Greak Lakes Symposium on VLSI'01*, West Lafayette, USA, Mar. 2001, pp. 103-108.

[6] S. C. Chang and K. T. Cheng, "Postlayout Logic Restructuring Using Alternative Wires," *IEEE Trans. Computer-Aided Design*, vol. 16, pp. 587-596, June 1997.

[7] S. C. Chang, L. P. v. Ginneken, and M. Marek- Sadowska, "Fast Boolean Optimization by Rewiring," in *Proc. IEEE/ACM International Conference on Computer-aided Design'96*, San Jose, CA, USA, Nov. 1996, pp. 262-269.

[8] S. C. Chang, L. P. v. Ginneken, and M. Marek- Sadowska, "Circuit Optimization by Rewiring," *IEEE Trans. Computers*, vol. 48, pp. 962.970, Sept. 1999.

[9] S. C. Chang, M. Marek-Sadowska, and K. T. Cheng, "Perturb and Simplify: Multi-Level Boolean Network Optimizer," *IEEE Trans. Computer-Aided Design*, vol. 15, pp. 1494-1504, Nov. 1996.

[10] K. T. Cheng and L. A. Entrena, "Multi-Level Logic Optimization by Redundancy Addition and Removal," in *Proc. IEEE European Design Automation Conference'93*, Paris, France, Feb. 1993, pp. 373-377.

[11] C. C. Cheung, Y. L. Wu, and D. I. Cheng, "Further Improve Circuit Partitioning Using GBAW Logic Perturbation Techniques," in *Proc. IEEE Conference and Exhibition on Design, Automation and Test in Europe'01*, Munich, German, Mar. 2001, pp. 233-239.

[12] J. Cong, J. Y. Lin, and W. N. Long, "A New Enhanced SPFD Rewiring Algorithm," in *Proc. IEEE/ACM International Conference on Computer-aided Design'02*, San Jose, CA. USA, Nov. 2002, pp. 672-678.

[13] L. A. Entrena and K. T. Cheng, "Combinational and Sequential Logic Optimization by Redundancy Addition and Removal," *IEEE Trans. Computer-Aided Design*, vol. 14, pp. 909-916, July 1995.

[14] M. A. Iyer and M. Abramovici, "FIRE: A Fault-Independent Combinational Redundancy Identification Algorithm," *IEEE Trans. VLSI Syst.*, vol. 4, pp. 295-301, June 1996.

[15] A. Ling, D. P. Singh, and S. D. Brown, "FPGA Technology Mapping: a study of optimality," in *42nd ACM/IEEE Design Automation Conference*, 2005, pp. 427-432.

[16] V. Manohararajah, S. D. Brown, and Z. G. Vranesic, "Heuristics for Area Minimization in LUT-based FPGA Technology Mapping," *IEEE Trans. Computer-aided Design*, vol. 25, pp. 2331- 2340, Nov. 2006.

[17] A. Marquardt, V. Betz, and J. Rose, "Timing-driven Placement for FPGAs,", in *Proc. Int'l Symp. on Field Programmable Gate Arrays*, Monterey, CA, USA, Feb. 2000, pp. 203-213.

[18] S. Sinha and R. K. Brayton, "Implementation and Use of SPFDs in Optimizing Boolean Networks," in *Proc. IEEE/ACM International Conference on Computer-aided Design'98*, San Jose, CA, USA, Nov. 1998, pp. 103-110.

[19] Y. L. Wu, W. N. Long, and H. B. Fan, "A Fast Graph-Based Alternative Wiring Scheme for Boolean Networks," in *Proc. IEEE International Conference on VLSI Design'00*, Calcutta, India, Jan. 2000, pp. 268-273.

[20] S. Yamashita, H. Sawada, and A. Nagoya, "A New Method to Express Functional Permissibilities for LUT based FPGAs and Its Applications," in *Proc. IEEE/ACM International Conference on Computer-aided Design'96*, San Jose, CA, USA, Nov. 1996, pp. 254-261.

Table 3. Performing logic perturbation on technology mapping and routing to affect FPGA area and delay

	Channel Width			Critical Path Delay		
Circuit	RW+TM	(TM+RT) +RW	Red. (%)	no RW	(TM+RT) +RW	Red. (%)
5xp1	4	4	0	2.49	1.93	22.49
C1355	6	6	0	3.41	3.41	0
C2670	6	5	16.67	6.66	5.47	17.87
C880	6	6	0	3.60	3.60	0
alu2	6	6	0	5.00	5.00	0
b9_n2	4	4	0	2.08	2.08	0
f51m	4	4	0	2.17	2.17	0
misex3	6	6	0	5.03	5.45	-8.35
pcler8	4	4	0	1.87	1.87	0
term1	5	5	0	2.29	2.29	0
ttt2	5	4	20.00	2.49	2.05	17.67
x3	5	5	0	3.80	3.98	-4.74
Average			3.06			3.75

RW: rewiring TM: technology mapping RT: routing Red.: reduction

Table 1. Rewiring-based technology mapping's impact on FPGA area and delay performance (K = 4)

	# CLBs			# Slots			Routing Area			Critical Path Delay (e-08 s)		
Circuit	no RW	TM + RW	Red. (%)	no RW	TM + RW	Red. (%)	no RW	TM + RW	Red. (%)	no RW	TM + RW	Red. (%)
5xp1	36	33	8.33	36	36	0	48685.7	48685.7	0	2.49	1.82	26.90
C1355	80	78	2.50	100	100	0	192733	192733	0	3.41	3.29	3.52
C1908	133	122	8.27	144	144	0	274493	317414	-15.64	4.99	5.03	-0.80
C6288	979	649	33.71	1024	676	33.98	1882590	1053290	44.05	14.22	13.50	5.06
C880	120	119	0.83	121	121	0	231817	231817	0	4.11	4.26	-3.65
alu2	158	130	7.59	169	144	14.79	320762	274493	14.42	5.00	5.47	-9.4
apex6	240	220	8.33	900	900	0	1657100	1397080	15.69	3.97	4.32	-8.82
Comp	32	30	6.25	36	36	0	37438.3	48685.7	-30.04	3.06	2.75	10.13
duke2	153	135	11.76	169	144	14.79	320762	317414	1.04	3.77	3.09	18.04
f51m	42	39	7.14	49	49	0	65304.4	65304.4	0	2.17	2.17	0
pcler8	38	37	2.63	49	49	0	65304.4	65304.4	0	1.87	2.11	-12.83
term1	70	59	15.71	81	64	20.99	105786	105802	-0.01	2.29	2.61	-13.97
ttt2	64	56	12.5	64	64	0	105802	105802	0	2.49	2.05	17.67
x3	243	224	7.82	900	900	0	1397080	1397080	0	3.80	4.00	-5.26
Average			9.53			6.04			2.11			1.90

RW: Rewiring TM: technology mapping Red.: reduction

Table 2. Rewiring-based routing's impact on FPGA area and delay performance (K = 4)

	#Nets	#Trans.	Ratio	Channel Width			Critical Path Delay (e-08 s)			CPU Time (s)		
Circuit			%	no RW	RT+RW	Red. (%)	no RW	RT+RW	Red. (%)	VPR	Engine	Ratio
5xp1	43	2	4.65	4	4	0	2.49	1.70	31.74	1.31	0.12	0.09
C1355	121	0	0	6	6	0	3.41	3.41	0	10.03	0.07	0.07
C1908	166	7	4.22	7	6	14.29	4.99	5.59	-	6.56	0.41	0.06
C6288	1011	0	0	5	5	0	14.22	14.22	0	139.90	0.98	0.01
C880	180	6	3.33	6	6	0	4.11	3.48	15.33	13.81	0.19	0.01
alu2	168	18	10.71	6	6	0	5.00	4.79	4.15	30.65	2.59	0.08
apex6	375	0	0	5	5	0	3.97	3.97	0	101.65	2.33	0.02
comp	64	2	3.13	3	3	0	3.06	2.47	19.37	1.30	0.03	0.02
duke2	175	6	3.43	6	6	0	3.77	3.30	12.57	25.18	1.87	0.07
f51m	50	1	2.00	4	4	0	2.17	1.93	11.06	1.60	0.20	0.12
pcler8	65	0	0	4	4	0	1.87	1.87	0	1.34	0.02	0.01
term1	104	11	10.58	5	5	0	2.29	1.99	12.91	4.74	0.24	0.05
ttt2	88	7	7.95	4	4	0	2.49	1.81	27.11	3.44	0.14	0.04
x3	378	6	1.59	5	5	0	3.80	3.68	3.10	71.20	1.62	0.02
Average			3.69			1.02			10.56			0.05

Trans.: transformation RW: rewiring RT: routing Red.: reduction