Placement-Driven Technology Mapping for LUT-Based FPGAs

Joey Y. Lin[*], Ashok Jagannathan[+], Jason Cong[+]
[*]Aplus Design Technologies, Inc.
[+]UCLA Computer Science Department
lin@aplus-dt.com, {ashokj, cong}@cs.ucla.edu

ABSTRACT

In this paper, we study the problem of placement-driven technology mapping for table-lookup based FPGA architectures to optimize circuit performance. Early work on technology mapping for FPGAs such as Chortle-d[14] and Flowmap[3] aim to optimize the depth of the mapped solution without consideration of interconnect delay. Later works such as Flowmap-d[7], Bias-Clus[4] and EdgeMap consider interconnect delays during mapping, but do not take into consideration the effects of their mapping solution on the final placement. Our work focuses on the interaction between the mapping and placement stages. First, the interconnect delay information is estimated from the placement, and used during the labeling process. A placement-based mapping solution which considers both global cell congestion and local cell congestion is then developed. Finally, a legalization step and detailed placement is performed to realize the design. We have implemented our algorithm in a LUT based FPGA technology mapping package named PDM (Placement-Driven Mapping) and tested the implementation on a set of MCNC benchmarks. We use the tool VPR[1][2] for placement and routing of the mapped netlist. Experimental results show the longest path delay on a set of large MCNC benchmarks decreased by 12.3% on the average.

Categories and Subject Descriptors

B.7.2 [**Hardware**]: INTEGRATED CIRCUITS – *Design Aids*.

General Terms

Algorithms, Performance, Design, Experimentation.

Keywords: Logic re-synthesis, FPGA synthesis, mapping

1. INTRODUCTION

The interaction between synthesis and physical design has received increased attention from researchers. In general, logic synthesis and optimization techniques have significant impact on the performance of large circuits. The logic structure decided by synthesis stage may cause remarkable differences in the quality of their physical implementations. It is expected that a tight interaction between synthesis and physical design can lead to better circuit performance.

Traditional design methodology for lookup-table (LUT) based FPGAs consists of a technology mapping phase, where a given Boolean circuit is converted into a functionally equivalent network comprised only of LUTs, followed by a placement and routing phase to realize an implementation of the mapped network. As the routing resources in FPGAs are slower and more limited compared to ASIC technologies, the technology mapping process has a significant impact on the performance of the implemented circuit. For early technologies where gate delays were dominant, the objective of the technology mapping process was to minimize the depth of the mapped network. Much of the early work on LUT-based FPGA technology mapping addresses exactly this problem. The Flowmap algorithm [3] solves this problem optimally using elegant network flow computations. Cut enumeration based algorithms such as Cutmap [8] allow a trade-off between the depth and area (in terms of the number of LUTs) in the mapped network.

However, with the advent of deep submicron technology, the delay of the interconnects has started to dominate the gate delay. Under such conditions, minimizing the depth of the mapped network does not accurately capture the performance of the circuit after placement and routing. More recent mapping algorithms such as Flowap-d [7], Bias-Clus[4] and Edge-Map[9] consider the delays of the wires during mapping. Flowmap-d assumes that each net may have a different delay but uses the same delay for every segment of net, while Bias-Clus and Edge-Map accommodate non-uniform delays for different segments of the same net. However, none of these approaches consider the effect of the resulting mapping solution on placement when the solution is generated.

The work in [5] proposes a simultaneous approach to technology mapping and linear placement for tree like circuits. Their work focuses on ASIC technology, and can also be applied to FPGAs. However, experimental results in Flowmap [3] have shown that such a decomposition of the circuit into disjoint trees followed by mapping may result in significant increase in the area of the mapped netlist, which may not be desirable.

We identify two main drawbacks with the existing approaches for technology mapping for FPGAs:

1) They assume that the delays on the interconnect segments are "fixed" during and after mapping, which does not always hold. For example, the delays of the visible edges mentioned in [4] will significantly change depending on the locations of the LUTs connected by these edges, which is determined only during placement. This assumption makes the post-layout delay of these mapped solutions less predictable from the mapping solution.

2) They do not consider the effect of the mapping solution on the final layout when decisions are made during the generation of the mapped solution itself.

In this work, we propose a heuristic solution to the simultaneous technology mapping and placement problem for FPGAs, without decomposing the original network into disjoint trees. In this context, we introduce a placement-driven mapping technique to consider the effect of mapping decisions on the interconnect delay and cell congestion in the placement. Basically, we eliminate the notion of "fixed" delays for interconnects as in [4] and use a table-lookup method to estimate edge delays based on the current mapping solution and its associated placement. Our approach consists of two phases – (a) a cut-enumeration based technique for simultaneous technology mapping and coarse placement generation with consideration of dynamically changing interconnect delays and cell congestion and (b) placement legalization and refinement. In (a), coarse placement implies that there may be overlapping cells in the placement.

The rest of the paper is organized as follows. In Section 2, we formally define our problem and outline our approach. In Section 3, we briefly talk about our timing-driven decomposition. In Section 4, we discuss in detail our simultaneous mapping and placement methodology. Section 5 deals with the legalization and refinement of the placement generated by the mapping algorithm, while Section 6 presents the experimental results. The conclusions and possible future work are presented in Section 7.

2. PRELIMINARIES AND PROBLEM FORMULATION

A general Boolean network N can be represented as a directed acyclic graph (DAG) where each node represents a logic gate, and a directed edge (u, v) exists if the output of gate u is an input of gate v. A primary input (PI) node has no incoming edge and a primary output (PO) node has no outgoing edge. We use *input(v)* to denote the set of nodes which are fanins of gate v. Given a subgraph H, *input(H)* denotes the set of nodes outside H which are the inputs to the gates in H. For a node v in the network, a cone of v, denoted C_v, is a subgraph consisting of v and its predecessors such that any path connecting a node in C_v and v lies entirely in C_v. If $|input(C_v)| \leq K$, we call C_v as a *K-feasible* cone at v and a *K-feasible* cut is the corresponding input set of this cone. A Boolean network is *K-bounded* if $|input(v)| \leq K$ for each node v.

The technology mapping problem for K-LUT based FPGAs is to cover a given K-bounded Boolean network N into a functionally equivalent network M comprised of K-LUTs, such that the circuit delay and/or area is minimized. The circuit delay depends on the delay model used to estimate the delay of gates and interconnects during the mapping process.

Such a Boolean network of K-LUTs can be implemented on a lookup-table based FPGA, using FPGA placement and routing tools. We assume that each programmable logic block on the FPGA is a K-input lookup-table (K-LUT) that can implement any K-input Boolean function. The placement process will assign a pair of coordinates for each LUT u, denoted as *location(u)*. Since two LUTs cannot be placed in the same location, we have *location(u)≠location(v)* when $u≠v$. A natural objective of performance-driven placement algorithms is to generate a placement which minimizes the maximum delay along any path in the placement. The delay along any path in the placement comes from both the gates and the interconnects along that path. The gate delay is usually a constant value d_g for LUT-based FPGAs.

However, the interconnect delay depends on many factors, such as the wire length, capacitive load, etc. We assume that the interconnect delay between two locations a and b, denoted *delay(a, b)*, can be obtained by a table-lookup method.

Ideally, a simultaneous approach to technology mapping and placement would produce an implementation with the best performance. However, due to practical limitations, a two-step approach as discussed above is generally followed. Since the technology mapping and the placement generation are decoupled, we propose that a placement-driven technology mapping of the circuit based on the generated placement could significantly improve the performance of the circuit.

Formally, we state the placement-driven technology mapping problem as follows:

Given a placement solution of K-LUT network M and delay models for gates and interconnects, generate another K-LUT network M' and corresponding placement with better performance.

Figure 1 shows the different stages of our approach. The circuit is first optimized by various technology independent optimization techniques [11]. We use the FPGA technology mapping algorithm Cutmap [8] to generate the initial mapping solution and VPR [2] to place and route the mapped LUT network. The input to our placement-driven mapping approach is the mapped LUT netlist generated by Cutmap and the corresponding placement generated by VPR. Our work has three phases:

(a) Timing-driven logic decomposition of the LUT-network into 2-input gates and assigning locations to these gates.

(b) Placement-driven mapping, which, starting from the initial decomposed network of 2-input gates with locations, generates another K-LUT network to improve the delay and/or area. A placement for this newly generated network is also produced during this step. However, this placement may not be legal – i.e., more than two LUTs can be positioned at the same location.

(c) Placement legalization and refinement phase which removes the overlaps in LUT positions and further refines the placement for performance.
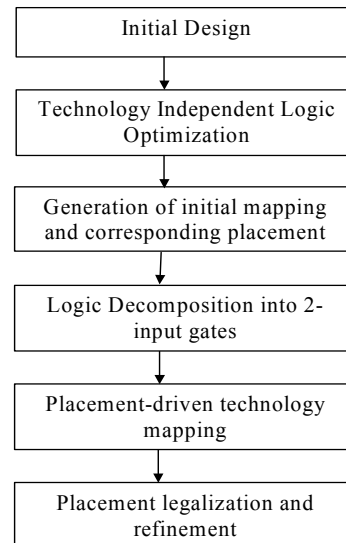

Figure 1. Optimization flow

In the following sections, each of these three steps is discussed in detail.

3. TIMING-DRIVEN LOGIC DECOMPOSITION

In order to search a large mapping solution space, we first decompose the mapped network into a 2-input AND/OR gate network. Since the placement for the initial mapping solution is given, we can derive some useful delay information to guide our decomposition. Our decomposition method is a modified version of the DMIG [13] decomposition algorithm to consider interconnect delay. We decompose the gates in topological order starting from the primary inputs. Therefore, when a gate is considered for decomposition, all of its inputs have already been decomposed. All the 2-input gates formed by decomposing a LUT are assigned the same location as the LUT in the placement. The original DMIG algorithm computes the arrival times of all input signals at a LUT, and combines the two early arriving signals (much like building a Huffman code) to form a 2-input gate. In our version, since we know the locations of the gates being decomposed, we compute these arrival times not only based on gate delays, but also based on the interconnect delays between the fanin gate location and the gate being decomposed. We use a delay lookup-table to get the delay estimate between two locations in the placement. This guarantees a minimum delay decomposition for each node, based on the given gate and interconnect delay models.

4. PLACEMENT-DRIVEN MAPPING

Once the network is decomposed and locations are assigned to the 2-input gates in the decomposed network, we apply our placement-driven mapping technique on this network. The mapping process is done in two phases (1) label each node in network with their best possible signal arrival time, and (2) perform the actual mapping of simple gates into LUTs while considering both delay and area. Both these phases are explained in detail below.

4.1 Labeling phase

In the labeling phase, we will label each node with the best arrival time calculated based on the interconnect delay model associated with our placement. In order to estimate interconnect delays, each mapped LUT must be assigned a physical location. *During the mapping stage we assume that the LUT will be placed at the location of the root node of the corresponding cone*. Based on this assumption, the interconnect delay between two LUTs becomes the delay between the corresponding root node locations.

Given a node v, since each *K-feasible* cut corresponds to a mapping solution of this node, we will first find all of its *K-feasible* cuts Cut_v. Since we work on a 2-bounded network, it is efficient to use cut enumeration technique[10] to calculate Cut_v. Suppose node v has two inputs w and u, and Cut_w and Cut_u are known. We can derive

$Cut_v=\{X_w\cup X_u \mid X_w\in Cut_w\cup\{\{w\}\}, X_u\in Cut_u\cup\{\{u\}\}, |X_w\cup X_u| \leq K\}$

After finding all the cuts, we can calculate the maximum signal arrival time A_X for each cut X. The signal arrival time of node v can be separated into three parts -- the arrival time to an input node u, the interconnect delay between u and v, and the gate delay of v. The arrival time A_X is max$\{label(u)+delay(location(u), location(v))+d_g|u\in X\}$.

Thus, our labeling phase proceeds as shown below:
0. Assign 0 to *label(u)* for node u which is a PI or FF output;
1. For each node v in topological order
2.   Enumerate all *K-feasible* cut into Cut_v;
3.   *label(v)* = ∞;
4.   For each cut X in Cut_v
5.   Calculate A_X as max$\{label(u)+ delay(location(u), location(v))+d_g |u\in X\}$;
6.   if ($label(v)> A_X$)
7. $label(v) = A_X$;

Theorem: Under the assumptions that each LUT will be placed at the root node of the corresponding cone, the above mentioned algorithm finds the best signal arrival time for each node under the given gate and interconnect delay models.

The proof of this theorem is a simple extension of the proof of the labeling phase of existing mapping algorithms. Interested readers are pointed to [3].

4.2 Mapping phase

Although we find the best signal arrival time during the labeling phase, we cannot determine whether we can find a mapping and placement solution to achieve this best arrival time. This problem is mainly due to our assumption made in the labeling phase that every LUT will be placed at the location of the root node of the K-feasible cone. But in the original network, a LUT l can be decomposed to several 2-input nodes. If two of them are the root nodes of different K-feasible cones, we need to place 2 LUTs at the slot of LUT l. This will make the placement solution infeasible. This is the so called *cell congestion problem* and it is the key problem in our placement-driven mapping.

We classify the cell congestion problem into two types: *global cell congestion* and *local cell congestion*. Global cell congestion is caused by the total area increase between our mapping and original mapping. The placement algorithm usually packs the LUT network in a small area and if we increase the total number of LUTs by a large amount, we must change the locations of many LUTs in the corresponding placement. As a consequence, the placement information we exploit in the labeling phase becomes invalid. Local cell congestion means that many LUTs are assigned to a small area. Although neighborhood areas may still be empty, we still need to move those LUTs far away.

Our mapping phase is an iterative procedure and contains multiple mapping passes. Each pass uses the cell congestion information gathered during previous iterations to guide the mapping decisions made during this pass. Mapping in each pass starts from the PO, and maps nodes backward until PI. Since the largest label in our labeling phase is the best critical path we expected, we use it as our optimization target, and set it as the signal required arrival time for each PO.

When we map a node v, first we calculate the required arrival time of this node *require(v)*. This value can be derived from those nodes in its fanout cone that are already mapped. For example, if node u is a node already mapped and v belongs to its cut, *require(v)=min(require(v),require(u)-delay(location(v), location(u))-d_g)*. It is easy to prove that if the signal arrival time at node v is earlier than *require(v)*, we can achieve the minimum delay at the POs in the mapping solution.

After calculating *require(v)*, we can determine the possible candidate cuts from Cut_v whose best arrival time is less than *require(v)*. For each candidate cut, we will evaluate it with a cost function which handles the cell congestion problem. We will choose the cut with the best cost as the mapping solution for node *v*.

The cost function is trying to handle the important cell congestion problem. For each cut, the cost is the sum of all the nodes in this cut. If a cut is selected in the mapping phase, the nodes in this cut will be implemented in the placement. Therefore the cost of these nodes must represent their contribution to the cell congestion. We define the cost with the following priorities.

♦ **1st priority**

If a node is already in the cut of previously mapped node in the same mapping pass, using this node will increase neither global nor local cell congestion. Thus its cost is set to 0.

♦ **2nd priority**

If a node fans out to many LUTs in the previous mapping passes, it is very likely to be reused in this mapping pass. We will assign the node with a very small cost ε.

♦ **3rd priority**

Our goal is to find a mapping solution without cell congestion. Actually the original mapping is a solution without any cell congestion. The only problem is that it cannot meet our timing target. If our new mapping solution only made changes at some critical points, it will introduce less cell congestion. Our 3rd priority cost is based on this observation -- if a node is the root node of any LUT in the original mapping solution, it is assigned a small cost value δ, with δ > ε.

During the procedure of generating mapping solutions backward from primary outputs, when a root node is mapped and the original mapping solution can satisfy the timing requirement, the cost of the original cut will be small due to the small δ. Therefore the original solution will be retained for this node. The exception is when there is another cut whose nodes are of 1st or 2nd priority. In this case the new changes will only result in fewer LUTs than the original solution, and it is less likely to introduce new LUTs.

♦ **4th priority**

We use a hierarchical area control scheme to evaluate the local congestion cost. In this scheme, we count the area increase in several bin levels as in Figure 2. For example, if we are trying to put a new node *v* into our mapping solution, we check whether we will have area overflow in the adjacent bin regions. The bin regions are designed hierarchically, from smallest size to largest size. Penalty costs will be given to bins at every level if the area overflows.

♦ **5th priority**

We also adopt the idea from the FPGA routing tool PathFinder [12]. After each mapping pass, we accumulate the actual number of nodes assigned in each small region. In the ongoing mapping pass, we will use these records to guide the new mapping, i.e., we assign a node with a high cost if this node is in a region which contains a lot of LUTs in previous passes.

The 1st, 2nd and 3rd priority costs are applied to handle global cell congestion, and the 3rd, 4th and 5th priority costs are for local

cell congestion. With these cost evaluations, we get a best timing mapping solution with less cell congestion problems. Since we may still have some congestion, we need a legalization step to give us a legalized placement.

A few experiments are carried out to estimate the effect of each cost. The results show that only 2.3% cuts are using the 4th and 5th priority costs. Most of the cuts are only use 1st, 2nd and 3rd priority cost. Therefore, the majority of LUTs in the original circuits are unchanged.
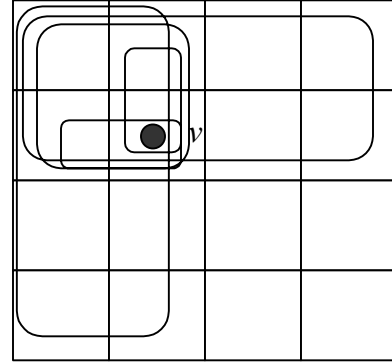

Figure 2. Hierarchical area control

5. PLACEMENT LEGALIZATION AND REFINEMENT

Though the mapping algorithm described in the previous section tries to minimize the cell congestion, it is clear that the algorithm cannot guarantee an overlap-free placement. We need a method to remove these overlaps and obtain a high-quality placement using the location hints provided by the mapping solution, without significantly sacrificing the performance estimated by the mapping algorithm. We perform this in two steps (i) a timing-driven legalization step, where we move overlapping cells into empty locations in their neighborhood based on the timing slack available for the cell and, (ii) a simulated annealing based placement refinement phase, where we further refine the legalized placement to improve the circuit performance.

During the legalization phase, we compute the available timing slack for each LUT and flip-flop in the mapping solution. Cells positioned in the placement where there are no overlaps after the mapping phase are not affected by the legalization process. We sort all the overlapping cells in the placement (both flip-flops and LUTs) in non-decreasing order of their timing slacks. From this list, we move one cell at a time to the closest empty location in the placement till the placement is legalized. It is important to note that the movement of one cell might affect the slacks associated with other cells. In extreme case, we can recompute the slacks after each cell move. In practice, however, performing timing analysis after each cell move can be expensive in terms of runtime, we recompute the slacks after every *n* cell movements, where *n* is a user specified parameter. We choose n=50 in our experiment.

Since the legalization is a greedy approach to remove overlaps, it is oblivious to the effect of individual cell movements on the global circuit performance. Hence, we use a low temperature simulated annealing engine in the second phase to further improve the placement after the legalization step. The idea is to allow movement of cells within a small area around its location in the legalized placement which will potentially result in

Circuit	Chip size	Cutmap			Placement-driven Mapping							
		LUT	CLB	Longest path(ns)	LUT	CLB	Area increase	4x4 overflow	8x8 overflow	Largest label(ns)	Longest path(ns)	Delay reduction
s9234	24	400	446	44.2	444	486	9.0%	1.6875	1.031	37.5	43.6	1.4%
s5378	32	524	551	40	530	558	1.3%	1.1875	1.016	34.7	36.3	9.3%
s13207	40	921	1127	62.3	908	1114	-1.2%	1.3125	1.0625	60.5	63.7	-2.2%
cordic	40	1236	1247	46.2	1242	1253	0.5%	1.25	1.031	45.8	45.1	2.4%
s15850	40	1254	1351	73.4	1225	1322	-2.1%	1.125	1	74.1	73.3	0.1%
dsip	56	1371	1371	61.4	1370	1370	-0.1%	1	1	21.5	40.1	34.7%
mult32	56	2623	2655	334.7	2622	2654	0.0%	1	1	192.2	320.4	4.3%
s35932	64	2937	3216	81.8	2921	3200	-0.5%	1	0.953	33.1	58.2	28.9%
s38417	64	3728	4023	81.6	3646	3941	-2.0%	1.0625	1.016	74.9	68	16.7%
s38584	72	4688	4885	80.8	4460	4657	-4.7%	1.0625	1	72.5	58.1	28.1%
												12.3%

Table 1. Area and delay comparison between cutmap and placement-driven mapping

the reduction of the critical path delay of the placement. Experimental results show significant gains with such a placement refinement approach.

6. EXPERIMENTAL RESULTS

We have implemented our mapping algorithm in C++ on top of SIS [11] framework and tested on a set of MCNC benchmarks. We use VPR's [2] placement and routing engine to generate the interconnect delay table to be used by our mapping algorithm and perform layout of the mapped networks. We use a 4-input LUT architecture for all our experiments. Several experimental setup issues are discussed in this section.

For each circuit, we first perform technology-independent logic optimization using *script.algebraic* in SIS. We then generate a technology mapping solution using the Cutmap [8] algorithm. The mapped netlist is packed into a set of BLEs using the tool T-VPACK [2]. This network of BLEs is placed and routed using VPR to obtain the initial locations for the LUTs and flip-flops in the mapped network. We then use a timing-driven version of the DMIG [13] algorithm discussed earlier to decompose the LUTs in the network into a set of 2-input gates. The delay of a 2-input gate is assumed to be $1/3^{rd}$ of LUT delay during decomposition. All the 2-input gates generated by decomposing a LUT get the same location as that of the LUT.

Our mapping algorithm also generates a mapped netlist and locations for LUTs and flip-flops in the network. Ideally, the placement legalization and refinement step would work directly on a network of LUTs and flip-flops, allowing each element to move independently. However, as we do not have a layout tool which can work on these elements independently, we use T-VPACK to pack our mapped netlist into a network of BLEs. The location of a BLE is decided as follows:
- If the BLE contains a flip-flop (irrespective of whether it also contains a LUT or not), then the location of the flip-flop in the mapped network is the initial location of the BLE.

- If the BLE contains only a LUT, then the location of the LUT in the mapped network is the initial location of the BLE.

We implemented the legalization algorithm on top of VPR data structures, which now legalizes BLEs and not flip-flops and LUTs individually. After legalization, we use VPR's simulated annealing engine to further refine the placement. Since we only want to perturb the solution by a small amount, we start the annealing process at a low temperature and limit the movement of blocks by VPR to a small region around its location in the legalized placement.

In our detail placement experiments with VPR, the temperature is varied from 10^{-4} to 10^{-8}, $\alpha=0.97$, the starting movement range is set to 10. During the detail placement phase, we only focus on the critical path. So the –timing_tradeoff parameter is set to 1 and starting critical path exp is set to 3.

The experimental results for a set of MCNC benchmark circuits are listed in Table 1. We selected sequential benchmarks that have more than 2000 internal nodes in the initial netlist. Column 2 shows the width and height of the placement area. Columns 3, 4, 6 and 7 show the number of LUTs and CLBs respectively after the mapping and packing stages. Column 8 gives the percentage increase in the number of CLBs. Columns 9 and 10 represent the local congestion. The 4x4 overflow value is calculated as the maximum number of LUTs in any 4x4 CLB region divided by 16 after the mapping stage. The 8x8 overflow value is calculated similarly. Columns 5, 12 and 13 are the longest path delay estimated by VPR after the placement stage. Column 11 is the largest delay label obtained after the labeling phase as described in Section 4.1.

Table 1 shows that the placement-driven mapping improves performance by 12.3% on the average. From the data, we can see that there is a consistent improvement in the longest path delay for all the big circuits (with greater than 1000 LUTs). This is probably due to the fact that the large circuits have a lot of paths available for the detailed placement tool to trade-off, which may not be the case with smaller circuits. Also, for all these cases, the local and global cell congestion values are very low. Thus the final detailed placement can do a better job in legalizing these mapped results.

Circuit	Original	Method A	Reduction	Method B	Reduction
s9234	44.2	43.6	1.4%	40.7	7.9%
s5378	40	36.3	9.3%	41.9	-4.7%
s13207	62.3	63.7	-2.2%	67.9	-9.0%
cordic	46.2	45.1	2.4%	44.2	4.3%
s15850	73.4	73.3	0.1%	74.2	-1.1%
dsip	61.4	40.1	34.7%	48.7	20.7%
mult32	334.7	320.4	4.3%	315.7	5.7%
s35932	81.8	58.2	28.9%	89.8	-9.8%
s38417	81.6	68	16.7%	N/A	N/A
s38584	80.8	58.1	28.1%	67.9	16.0%
			12.3%		3.3%

Table 2. Comparison between different cost assignment methods
Note: N/A means that the new solution cannot fit into the original placement grid due to area increase.

On the other hand, circuits that have higher cell congestion values have smaller improvements or even worse longest path delay as in the case of s13207.

In order to evaluate the importance of the components of the cost function, we also collected delay results for different cost function settings. In Table 2, we show results with and without 3rd priority cost. Method A is using all the 5 priority cost assignments. Method B is using all the other cost assignment except for the 3rd priority cost. We can find that the 3rd cost consistently improves the solution quality. Thus, this cost function is important in balancing the timing requirement and congestion well during the mapping process.

7. CONCLUSIONS AND FUTURE WORK

We studied the problem of placement-driven technology mapping for LUT-based FPGAs. A general delay model, which considers dynamically changing interconnect delays based on actual LUT locations is considered. An effective technology mapping algorithm based on the cut-enumeration technique was developed which optimizes the circuit performance with consideration of interconnect delays and cell congestion. Experimental results show significant improvements in the post-layout timing results on a set of large circuits compared to traditional interconnect-delay unaware mapping approaches.

Future work will include developing a better legalization and incremental placement tool that can deal with cell congestion issues better. Also, as routing resources in FPGAs are scarce, we would like to extend our mapping algorithm to consider routing congestion when generating the mapping solution.

8. ACKNOWLEDGEMENT

9. REFERENCES

[1] A. Marquardt, V. Betz, and J. Rose, "Timing-driven placement for FPGAs," *Proceedings of ACM/SIGDA International Symposium on FPGAs*, Monterey, CA, pp 203-213, Feb 2000.

[2] V. Betz, and J. Rose, "VPR: A new packing, placement and routing tool for FPGA research," *Proceedings of Field Programmable Logic, Seventh International Workshop* (Oxford, UK, Sept 1997), pp 213-222.

[3] J. Cong and Y. Ding. "An optimal technology mapping algorithm for delay minimization in lookup-table based FPGA designs," *Proceedings of the IEEE International Conference on CAD*, pp 48-53, Nov. 1992.

[4] A. Mathur and C. L. Liu. "Performance-driven technology mapping for lookup-table based FPGAs using the general delay model," *Proceedings of International ACM/SIGDA Workshop on FPGAs*, Feb 1994.

[5] J. Lou, A. H. Salek and M. Pedram, "An exact solution to simultaneous technology mapping and linear placement problem," *Proceedings of the International Conference on Computer Design*, pages 671-675, Nov. 1997.

[6] J. Cong and Y. Ding, "On area/depth trade-off in LUT-based FPGA technology mapping," *IEEE Transactions on VLSI Systems*, Vol. 2, June 1994.

[7] J. Cong, Y. Ding, T. Gao and K. C. Chen, "LUT-based FPGA technology mapping under arbitrary net-delay model," *Computers and Graphics*, vol. 18, no. 4, pp. 507-516, 1994.

[8] J. Cong and Y. Hwang, "Simultaneous depth and area minimization in LUT-based FPGA mapping," *Proceedings of ACM/SIGDA International Symposium on FPGAs*, Monterey, California, pp. 68-74, February 1995.

[9] H. Yang and D. F. Wong, "Edge-map: optimal performance-driven technology mapping for iterative LUT based FPGA designs," *Proceedings of IEEE/ACM International Conference on CAD*, pp 150-155, Nov. 1994.

[10] J. Cong, C. Wu and E. Ding, "Cut ranking and pruning: enabling a general and efficient FPGA mapping solution," *Proceedings of ACM/SIGDA International Symposium on FPGAs*, Monterey, California, pp. 29-35, Feb. 1999.

[11] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton and A. Sangiovanni-Vincentelli, "SIS: A system for sequential circuit synthesis", *Memorandum No. UCB/ERL M92/41, Electronics Research Laboratory*, College of Engineering, University of California, Berkeley, May 1992.

[12] L. McMurchie and C. Ebeling, "Pathfinder: a negotiation-based performance-driven router for FPGAs," *Proceedings of 3rd ACM/SIGDA International Symposium on FPGAs*, Feb. 1995, pp.111-117.

[13] K. C. Chen, J. Cong, Y. Ding, A. Kahng, and P. Trajmar, "DAG-map: graph-based FPGA technology mapping for delay optimization," *IEEE Design & Test*, pp. 7-20, Sept. 1992.

[14] R. J. Francis, J. Rose and Z. Vranesic, "Technology mapping of lookup table-based FPGAs for performance," *Proceedings of IEEE International Conference on CAD*, pp. 568-571, Nov. 1991.