Using Logic Duplication to Improve Performance in FPGAs

Karl Schabas
Dept. of Electrical and Computer Engineering
University of Toronto
Toronto, Canada
karl.schabas@utoronto.ca

Stephen D. Brown
Dept. of Electrical and Computer Engineering
University of Toronto
Toronto, Canada
brown@eecg.toronto.edu

ABSTRACT
The purpose of this paper is to introduce a modified packing and placement algorithm for FPGAs that utilizes logic duplication to improve performance. The modified packing algorithm was designed to leave unused basic logic elements (BLEs) in timing critical clusters, to allow potential targets for logic duplication. The modified placement algorithm consists of a new stage after placement in which logic duplication is performed to shorten the length of the critical path. In this paper, we show that in a representative FPGA architecture using .18 μm technology, the length of the final critical path can be reduced by an average of 14.1%. Approximately half of this gain comes directly from the changes to the packing algorithm while the other half comes from the logic duplication performed during placement.

Categories and Subject Descriptors
B.7.2 [**Integrated Circuits**]: Design Aids – *placement and routing*.

General Terms
Algorithms, Performance.

Keywords
FPGA, Logic Duplication.

1 INTRODUCTION
The purpose of this paper is to present a new packing and placement algorithm which uses logic duplication to shorten the critical path of a circuit. Logic duplication is a technique that seeks to make use of unused logic elements to shorten the length of the critical path and hence improve the performance of the circuit. For example, consider the partial placement result shown in Figure 1 where the critical path goes from cluster A to cluster B and terminates in cluster C. Figure 1a) represents the critical path before duplication, while Figure 1b) shows how duplicating a single basic logic element (BLE) from cluster B can shorten the length of the critical path by eliminating cluster B from this path.

FPGA'03, February 23-25, 2003, Monterey, California, USA.

Even if the cluster C has no empty BLEs to permit this duplication, the length of the critical path can still be shortened using logic duplication by finding another cluster D with sufficient room and a location that will still result in a more direct path from source to sink. This is shown in Figure 1c). Repeated application of this technique can greatly reduce the length of the critical path of a circuit, leading to an improvement in performance of the circuit being placed.

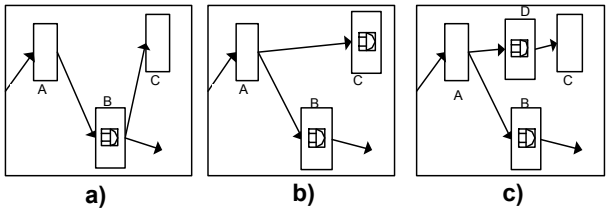

Figure 1 Sample Logic Duplication

The implementation of logic duplication devised for this paper is based on the combination of the T-Vpack packing algorithm [4,8] that packs the logic elements into clusters, and VPR [2,3,4] which places and routes the resulting clusters. In order for logic duplication to be successful, there must be empty BLEs inside clusters that lie on or near the critical path. Unfortunately, T-Vpack attempts to pack as many BLEs as possible into any given cluster. Therefore, one important contribution of this paper is to modify T-Vpack to allow for more empty BLEs in timing critical clusters. Second, the VPR placement algorithm must be modified to introduce logic duplications. This is accomplished by adding a new final stage to the VPR placer in which BLEs on the critical path are examined for possible duplications. The combination of the new packing algorithm and the modification to VPR results in a gain of 14.1%.

The paper is organized as follows. Section 2 describes the existing tools used for clustering and timing-driven placement. Section 3 describes the modified clustering and placement algorithms that use logic duplication. Section 4 presents the experimental results of the technique. The conclusions along with suggestions for future work are presented in section 5.

2 BACKGROUND
There are five major steps in transforming a circuit netlist into a physical implementation of a circuit on an FPGA. The first step is to perform multi-level logic synthesis. For this paper this was done using SIS [10]. In the next step, the logic gates and registers that form the circuit must be technology-mapped into Basic Logic Elements (BLEs). Each BLE consists of a combination of a 4-LUT and a register. For this paper, this task was accomplished

using Flowmap [6]. The next step consists of packing the individual BLEs into clusters, which was performed using T-Vpack [4,8]. Subsequently, the BLEs must be mapped to a specific physical location on the FPGA. Finally, the circuit must be routed to connect the clusters in the desired manner. For this paper, both the placement and routing steps were performed using the VPR timing-driven routing algorithm [2,3,4]. The sections below provide a more in-depth description of the packing and placement steps of the CAD flow, as these are the stages that must be modified to allow logic duplication to be successful.

2.1 Clustering With T-Vpack

T-Vpack [4,8] is a timing-driven logic block packing algorithm which attempts to minimize the number of inputs used in a cluster while also reducing the number of inter-cluster connections on near–critical paths to improve performance. It is a bottom up algorithm, meaning that initially each BLE is its own cluster, and that the clusters are packed together to create larger clusters. T-Vpack first performs a timing analysis by using constant delays for the delay through a BLE, the intra-cluster connection delay between BLEs in the same cluster and the inter-cluster connection delay between BLEs in different clusters. The timing analysis computes the *slack* of every connection, where *slack(i)* is the amount of delay that may be added to connection *i* before it becomes the longest path in the circuit. The criticality of a connection *i* can then be defined as:

$$(1) \quad Crit\,(i) = 1 - \frac{slack\,(i)}{Maxslack}$$

where *Maxslack* is the maximum slack of any of the connections in the circuit. The criticality of a BLE is defined as the maximum criticality of all connections incident to the BLE. T-Vpack first selects a seed, the initial BLE for a cluster, by determining which unclustered BLE has the highest criticality. Once a seed is chosen, each remaining BLE B is assigned an attraction value for cluster C based on the formula

$$(2) \quad Attraction(B) = \lambda \cdot Crit(B) + \frac{|Nets(B) \cap Nets(C)|}{MaxNets}$$

where λ is .75 by default and *MaxNets* is the maximum number of nets that could connect to any BLE. Packing occurs by adding the BLE with the highest attraction to the current cluster into that cluster. This continues until it is impossible to fit any more BLEs into the current cluster, in which case a new seed is chosen and the process is repeated.

In [11], the authors describe a different packing algorithm that restricts the number of input pins that may be used on any given cluster, possibly resulting in empty BLEs inside a cluster. However, this approach is based on optimizing routability instead of timing.

2.2 Placement With VPR

The placement algorithm used by VPR is based on the Simulated Annealing algorithm [7]. After a random initial placement the algorithm considers swapping the location of two randomly selected clusters. The effectiveness of this potential swap is measured using a cost function. The cost function has two components. The first component, called the bounding box cost or bb_cost, is a measure of how the swap will affect the ability to route the circuit. It can be calculated from the following formula:

$$(3) \quad bb\_cost = \sum_{i=1}^{Nnets} q(i)\left[\frac{bb_x(i)}{C_{av,x}(i)^{\beta}} + \frac{bb_y(i)}{C_{av,y}(i)^{\beta}}\right]$$

In this equation, q(i) is a number which is 1 for nets with less than 4 terminals, and rises slowly to 2.79 as the number of terminals increases to 50 [5]. $C_{av,x}(i)$ and $C_{av,y}(i)$ represents the average channel capacity (in tracks) in the x and y directions, and the terms $bb_x(i)$ and $bb_y(i)$ represent the x and y dimensions of the bounding box of net i. Effectively, this cost function penalizes nets with a large fanout and a large bounding box. The second part of the cost function represents the effect of the swap on the length of the near-critical paths of the circuit. To compute this portion of the cost function, a timing analysis is done similar to that performed by T-Vpack. The slack of connection *i* is defined as the amount of delay which may be added to the connection before it becomes the longest connection in the circuit. The criticality of the connection *i* is defined as:

$$(4) \quad crit(i) = 1 - \frac{slack(i)}{d\_max}$$

where *d_max* is the length of the longest path in the circuit. The total timing cost can now be written as:

$$(5) \quad timing\_cost = \sum_{all\ connections\ i} crit(i) \cdot delay(i)$$

where delay(i) is the delay of connection i. The *bb_cost* and *timing_cost* are then combined to form the cost function

$$(6) \quad \Delta cost = \lambda \cdot \frac{\Delta timing\_cost}{previous\_timing\_cost} + (1-\lambda)\frac{\Delta bb\_cost}{previous\_bb\_cost}$$

where λ is .5 by default. Any swap where the cost is negative is automatically accepted. If the cost is positive, the swap is still accepted with probability $e^{-\Delta c/t}$, where Δc is the change in cost and t is the current temperature.

3 IMPLEMENTING LOGIC DUPLICATION TO IMPROVE PERFORMANCE

This section describes a new method of utilizing logic duplication to achieve substantial gains in the maximum frequency of a circuit placed and routed in an FPGA. For duplication to be successful there are several issues that must be considered. These include modifications of the logic packing algorithm to create empty BLEs at ideal locations for logic duplication, a cost function to attempt to measure the effectiveness of potential duplications and an algorithm to incorporate duplication into the existing VPR placement algorithm. This section examines each of these issues, and presents the solutions we used to achieve our performance gains.

3.1 Logic Duplication

To understand the algorithm used for this paper it is important to define exactly what is meant by logic duplication. To perform a logic duplication there are three key parameters, the location of a

BLE to be duplicated (the source BLE), the location of an empty BLE (the target BLE) and a set of connections C that represent the fanout of the new BLE. If F is the set of all blocks in the fanout of the source BLE, then C, the fanout of the target BLE, must be a subset of F as described in section 3.3. If F=C then the source BLE has become unnecessary and should be automatically deleted. This effectively transforms the logic duplication into a BLE level move. Otherwise, the fanout of the source BLE after duplication will be F – C. Finally, the inputs must be connected to the target BLE. For any given input signal, if the signal is already available in the target cluster, either from another input pin on the cluster or as the output from another BLE in the cluster, then no new inter-cluster connections are necessary. Otherwise, a new connection must be added to an unused input pin of the target BLE's cluster to carry the required input signal.

3.2 New Packing Algorithm

The motivation behind a new packing algorithm is to introduce some empty BLEs into timing critical clusters. These will become the target BLEs used for logic duplication. To achieve this, a modification to T-Vpack was implemented. T-Vpack will select as its seed what it believes to be the most timing critical BLE, as determined by Equation (1). However, it will then attempt to fill this cluster with as many BLEs as possible. The modification to T-Vpack requires that for the first n clusters created during packing, instead of packing each cluster until it is full, a minimum of s BLEs will be left unused. The determination of n is based on the following equation:

$$(7) \quad n = \frac{N \cdot DeviceSize - numBLEs}{s}$$

where *DeviceSize* is the number of clusters available on the target device, *numBLEs*, is the number of BLEs in the circuit, and N is the number of BLEs in a cluster. For the results in this paper, N was chosen to be 10, s was set to 4 and *DeviceSize* was set to 20% greater than the number of BLEs in the circuit. Lower values of s would lead to more clusters with empty BLEs, but the number of BLEs left empty in these timing critical clusters would be smaller. After the first n clusters, the unmodified T-Vpack algorithm is used to continue packing the remaining BLEs into clusters.

3.3 New Placement Algorithm

This section describes an addition made to the current VPR placement algorithm that allows logic duplications. A separate stage was inserted after the end of the annealing process, but before routing. By waiting until the end of the placement algorithm before performing any duplications, the duplication algorithm can make use of the most accurate estimate of the inter-cluster connection delays in the circuit. The new duplication stage begins with a timing analysis. Next, each connection on the critical path is examined individually. For each of these connections, the BLEs at both ends are examined to determine if this critical connection can be either eliminated or shortened through logic duplication (see Figure 1). If more than one BLE in a cluster is on the critical path, then clearly moving only one of the BLEs from this cluster will not result in an improvement. To avoid this situation, the algorithm would consider moving all BLEs on the critical path and in the same cluster in one step. This is shown in Figure 2.

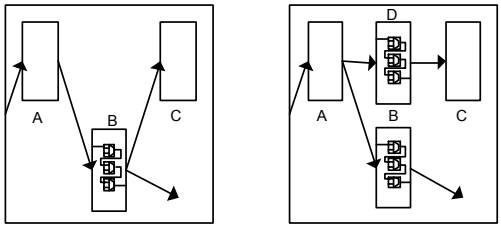

Figure 2 Duplication of Multiple BLEs

Once potential source BLEs have been identified the next step is to identify potential targets. A potential target cluster must have a sufficient number of empty BLEs and input pins. Furthermore, to be considered as a potential target, the resulting duplication must result in a shortening of the critical path. The cost of performing a duplication is calculated for every potential target, using Equation (6). For each of these duplications, the fanout of the new BLE will consist only of the previously critical connection. Any duplication in which the original BLE had a fanout of one will be transformed into a BLE level move, as the original BLE will become unnecessary. For each connection, the target with the lowest cost is stored, along with its associated cost.

Once all the critical connections have been considered the potential duplication with the lowest cost is performed. The algorithm then restarts with another timing analysis. A summary of this algorithm is shown in Figure 3. This continues until one of two exit conditions is met. First, it is possible that no duplications can be found using this algorithm that would decrease the length of the critical path. The other exit condition is more complex. Almost all logic duplications result in a net gain in the number of connections. This leads to a more difficult task for the router, compared to routing the circuit prior to logic duplication, since the number of near-critical paths has increased while the routing resources have remained constant. If this congestion becomes too severe, it may become impossible to route all of the critical or near-critical paths in an efficient manner. This can lead to circuits which after routing have a longer critical path than if duplication had never been performed. To avoid this, a variable *congestion* is calculated after every duplication. This congestion is defined as follows:

$$(8) \quad congestion = \frac{\sum_{\text{all connections } i} Crit(i)}{num\_connections}$$

where *Crit(i)* is as defined in Equation (4), and *num_connections* is the total number of point-to-point connections in the circuit. If the congestion is higher than a certain threshold value (set at .9 for this paper), then the circuit is deemed too congested for further duplications and the duplication algorithm is terminated.

```
For any connection (i,j), let BLE i be called the source and BLE j the sink
While (congestion < max_congestion) {
   Perform timing analysis
   dup = FALSE;
   for (each connection on critical path,c) {
      current_bles = set of all BLEs in the same block as source on the critical path;
      t = ∞;
      for (each block, b) { /*find best destination for duplicating source*/
         if (duplicating current_bles to b is legal and would shorten critical path) {
            q = compute cost of duplicating current_bles to b
            if (q < t) {
               t = q;
               src = current_bles;
               dest = b;
               fan = c;
               dup = TRUE;
            }
         }
      }
   }
   current_bles = set of all BLEs in the same block as sink on the critical path
   for (each block,b) { /*find best destination for duplicating target*/
      if (duplicating current_bles to b is legal and would shorten critical path) {
         q = compute cost of duplicating current_bles to b
         if (q < t) {
            t = q;
            src = current_bles;
            dup =TRUE;
            dest = b;
            fan = output of sink which is part of critical path
         }
      }
   }
   if (dup == FALSE) { /*no beneficial duplications found*/
      return;
   }
   duplicate src to destination dest with fanout fan
   congestion = average criticality of a connection in the circuit
   }
}
```

Figure 3 New Placement Algorithm

4 EXPERIMENTAL RESULTS

To test the effectiveness of logic duplication, we implemented the proposed modified packing and placement algorithms that allow logic duplication. The target FPGA consisted of an island-style architecture. Each BLE contained a 4-LUT and a register, and each cluster consisted of 10 BLEs. This architecture is similar to the Altera Stratix family [1]. The channel width for the device was set at 1.2 times the minimum channel width required to successfully route the circuit.

To obtain baseline values for the critical path, each of 20 large MCNC circuits were clustered, placed and routed in the target device using the unmodified version of T-Vpack [4,8] and VPR [2,3,4]. We then conducted two further experiments, first using only the modifications to T-Vpack and then using the modifications to both T-Vpack and VPR. Each of the three experiments was repeated five times with different random seeds and the results were averaged.

The results are summarized in Table 1. The baseline results are given in the second column. The third column shows the effect of using the modified packing algorithm, while fourth third column gives the effect of using both the modified packing algorithm and the modified placer that allows logic duplications. The final column shows the number of logic duplications and BLE level moves performed to achieve the resulting gains. As shown in the final column, an improvement of 14.1% can be obtained in the final critical path length over the current T-Vpack and VPR algorithms. Of this gain, 6.4% is due directly to the modifications to the packing algorithm, while the remaining 7.7% is due to the extra stage in the placement algorithm which implements logic duplication. Looking only at the performance benefit resulting from logic duplication, a 1.4% gain can be achieved by allowing only the BLE level moves permitted by the algorithm. To realize the remaining 6.3% gain, logic duplication must be used. The modified algorithm also results in an increase in the processing time required to complete the placement algorithm of 26%. Of this gain, 10% is due to the increased number of clusters that result from the modification to the packing algorithm. The remaining 16% is due to the extra stage at the end of the placement algorithm, with most of the extra time being used to perform the timing analysis after each duplication.

Table 2 shows the effect of the modified packing algorithm and logic duplication on the pre-routing critical path of the circuit. This is essentially the length of the critical path if there was an infinite channel width and hence all connections could be routed in the quickest possible manner. There are two interesting points from this table. First, when the modified packing algorithm is

Table 1 Overall Results

Circuit	Chip Size (nx,ny)	Final Critical Path Length (ns)			(duplications, moves)
		No Modifications	Modified Packing algorithm	Modified Packing and Placement Algorithm	
alu	(14,14)	15.8	13.5	13.1	(40.4,68.2)
apex2	(16,16)	15.3	15.8	14.1	(82.8,169)
apex4	(13,13)	16.0	13.8	13.3	(81.2,70.6)
bigkey	(16,16)	7.93	7.87	7.04	(113,106)
clma	(32,32)	30.1	30.4	26.0	(116,187)
des	(18,18)	15.8	13.1	12.9	(30.6,46.7)
diffeq	(14,14)	14.2	16.0	14.0	(36.2,14)
dsip	(16,16)	7.48	7.21	6.85	(20.2,40.8)
elliptic	(21,21)	22.1	21.3	19.6	(143,30.6)
ex1010	(24,24)	22.6	21.0	18.7	(236.4,365)
ex5p	(12,12)	14.5	14.4	13.4	(80.2,46.4)
frisc	(21,21)	31.0	27.3	24.8	(148.6,37.4)
misex3	(14,14)	13.2	13.1	12.3	(62.2,79.4)
pdc	(24,24)	25.5	20.7	20.6	(235,309)
s298	(16,16)	31.2	25.6	23.4	(37.6,149)
s38584	(28,28)	16.4	13.8	14.8	(61.8,77.8)
s38417	(28,28)	18.5	17.7	13.4	(479,132)
seq	(15,15)	14.0	13.4	12.5	(60.0,91.6)
spla	(22,22)	21.8	19.3	18.1	(194,283)
tseng	(12,12)	16.1	16.3	13.4	(45.6,16.6)
Total Percentage Gain			6.4%	14.1%	

used without duplication, the pre-routing critical path length actually increases, despite the fact the final, post-routing critical path length has decreased by 6.4%. By spreading the circuit out over the entire chip, the revised packing algorithm has reduced the congestion on the channels allowing the router to do a more effective job routing the circuit. The second point is that duplication is successful in greatly reducing the length of the critical path in each of the benchmarks. Much of this gain is lost by the router, but enough of the gain in the critical path length is preserved to make logic duplication a useful modification to the placement algorithm. The problems the router encounters in attempting to preserve the shorter critical path length are mostly due to a high level of local congestion somewhere in the circuit. It is particularly prevalent in circuits with a very short critical path (*dsip and bigkey)*. This is due to the fact that very short near-critical paths will have very small slack values that greatly restrict the number of possible routes for a near-critical path without it becoming the critical path.

The most important parameter in determining the effectiveness of the techniques presented in this paper proved to be the value of *s* (see Equation 7) and the associated interaction between the logic-packing and logic duplication algorithms. Figure 4 shows how the performance of logic duplication varies as the value of *s* is altered. When *s* is 2 or less, the logic-packing algorithm performs well on its own. However, the small value of *s* greatly harms the effectiveness of logic duplication. This is because there were often an insufficient number of empty BLEs in an ideal destination cluster to perform all of the logic duplications desired. Conversely, at values of *s* above 5, the logic-packing algorithm performs very poorly, so that even though logic duplication significantly increases performance, the overall performance gain remains less than optimal. The selection of s=*4* is a compromise in which neither the logic-packing algorithm nor the logic duplication algorithm is performing at its peak individually, but the overall performance gain is highest. The effect of several other variables, such as the value of *max_congestion* and the channel width were also investigated, and the results can be found in [9].

Table 2 Pre-Routing Critical Path Lengths

Circuit	Pre-Routing Critical Path Length (ns)		
	No modifications	Modified Packing Algorithm	Modified packing and placement algorithm
Alu4	12.7	13.0	11.6
Apex2	14.3	15.0	12.7
Apex4	11.9	12.3	10.6
Bigkey	6.55	7.40	5.87
Clma	26.6	28.9	22.3
Des	10.9	11.7	10.6
Diffeq	13.6	16.3	13.0
Dsip	6.05	6.40	5.51
Elliptic	18.9	19.3	17.3
Ex1010	19.3	20.5	16.8
Ex5p	12.3	12.9	10.7
Frisc	25.3	26.0	22.2
Misex3	11.9	12.2	10.6
Pdc	18.9	19.0	16.3
S298	21.5	23.0	19.3
S38417	13.5	17.1	13.3
S38584	15.9	12.8	11.4
Seq	12.0	12.5	10.6
Spla	17.1	17.4	14.3
Tseng	14.4	16.2	12.4
Total Percentage Gain		-5.8%	+11.4%

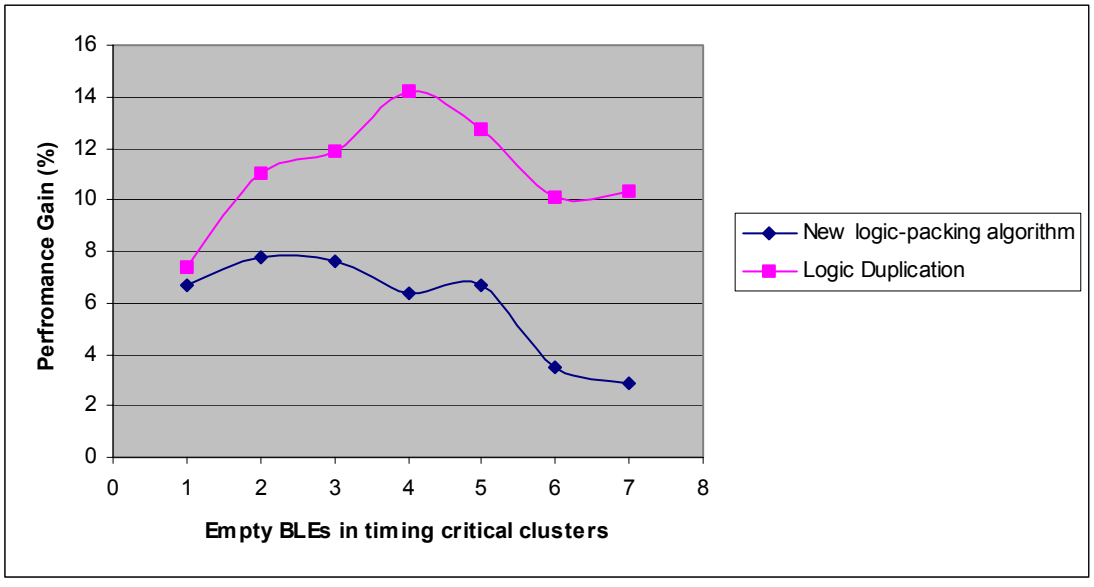

Figure 4 The effect of s on Logic Duplication

5 CONCLUSIONS

This paper has presented a new algorithm for placing a circuit on an FPGA which uses logic duplication to achieve significant gains in performance. The new algorithm consisted of two changes, a new packing algorithm that leaves empty BLEs in timing critical clusters, and a new placement algorithm that permits logic duplications to reduce the critical path length. The new packing algorithm resulted in a reduction of 6.4% in the length of the critical path compared to T-Vpack and VPR algorithms, while the addition to the placement algorithm resulted in an additional 7.7% reduction. This paper showed how logic duplication as an extra stage after placement could significantly reduce the length of the pre-routing critical path. This paper also presented the need for a cost function to determine the best possible duplications, and the need for a congestion metric to prevent logic duplication from overly congesting the circuit, leading to disastrous routing results.

There are many possible refinements that could be examined to try and further improve the effectiveness of logic duplication. At present, if the packing algorithm decides that a cluster is timing critical, it will always leave a set number of empty BLEs. An alternate method worthy of investigation would be to leave a variable amount of space in these clusters, ensuring that no BLEs which will be difficult to cluster elsewhere are left out of the current cluster. Another optimization would be to attempt to reduce the 26% additional processing time required for the new placement algorithm. This could be done by creating a method of approximating the criticality of new connections created by logic duplication without requiring a complete timing analysis after every duplication. Finally, the cost function used to determine the best duplication is currently based on the cost function used to determine the effectiveness of a swap during simulated annealing. It may be possible to develop an alternate cost function that will more effectively determine the effect of a duplication on the congestion of a circuit. This could potentially lead to a superior selection of logic duplications and could conceivably eliminate the need for a congestion function entirely.

6 REFERENCES

[1] Altera. “Stratix Datasheets”, Available from: http://www.altera.com/products/devices/stratix/stx-index.jsp

[2] Betz V. Architecture and CAD for Speed and Area Optimization of FPGAs. Ph. D. Dissertation, University of Toronto, 1998.

[3] Betz, V., and Rose, J. VPR: A New Packing, Placement and Routing tool for FPGA research. Int. Workshop on Field-Programmable Logic and Applications, 1997, pp. 213-222.

[4] Betz, V., Rose J., and Marquardt A. Architecture and CAD for Deep-Submicron FPGAs. Kluwer Academic Publishers, 1999.

[5] Cheng C. RISA: Accurate and Efficient Placement Routability Modeling. *ICCAD,* 1994, pp. 690-695.

[6] Cong, J. and Ding, Y. FlowMap: An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs. IEEE Trans. On Computer-Aided Design, Jan. 1994, pp. 1-12.

[7] Kirkpatrick, S., Gelatt, C., and Vecchi M. Optimization by Simulated Annealing. Science, May 13, 1983, pp. 671-680.

[8] Marquardt, A., Betz, V., and Rose, J. Using Cluster-Based Logic blocks and Timing-Driven Packing to Improve FPGA Speed and Density. ACM/SIGDA International Symposium on Field Programmable Gate Arrays, Monterey, CA, February 1999, pp. 37-46.

[9] Schabas, K. M.A.Sc. Thesis in Progress: Using Logic Duplication to Improve Performance in FPGAs. University of Toronto, 2002.

[10] Sentovich, E. M., et al. SIS: A System for Sequential Circuit Analysis. Tech. Report No. UCB/ERL M92/41, University of California, Berkeley, 1992.

[11] Singh, A. and Marek-Sadowska, M. Efficient Circuit Clustering for Area and Power Reduction in FPGAs. International Symposium on Field Programmable Gate arrays, Monterey, CA, February 2002, pp 59-66.