Hindawi Publishing Corporation
International Journal of Reconfigurable Computing
Volume 2010, Article ID 158602, 11 pages
doi:10.1155/2010/158602

Research Article

Timing-Driven Nonuniform Depopulation-Based Clustering

Hanyu Liu and Ali Akoglu

Department of ECE, University of Arizona, 1230 E. Speedway Blvd., Tucson, AZ 85721, USA

Correspondence should be addressed to Ali Akoglu, akoglu@ece.arizona.edu

Received 14 June 2009; Accepted 16 November 2009

Academic Editor: Elías Todorovich

Low-cost FPGAs have comparable number of Configurable Logic Blocks (CLBs) with respect to resource-rich FPGAs but have much less routing tracks. For CAD tools, this situation increases the difficulty of successfully mapping a circuit into the low-cost FPGAs. Instead of switching to resource-rich FPGAs, the designers could employ depopulation-based clustering techniques which underuse CLBs, hence improve routability by spreading the logic over the architecture. However, all depopulation-based clustering algorithms to this date increase critical path delay. In this paper, we present a timing-driven nonuniform depopulation-based clustering technique, T-NDPack, that targets critical path delay and channel width constraints simultaneously. T-NDPack adjusts the CLB capacity based on the criticality of the Basic Logic Element (BLE). Results show that T-NDPack reduces minimum channel width by 11.07% while increasing the number of CLBs by 13.28% compared to T-VPack. More importantly, T-NDPack decreases critical path delay by 2.89%.

1. Introduction

Field-programmable gate arrays (FPGAs) were first introduced in 1980s. While they are less efficient than ASICs, FPGAs are becoming more popular because of their low nonrecurrent engineering cost and fast time-to-market. Currently, commercial FPGAs can be categorized as low-cost and resource-rich families. As shown in Table 1, low-cost FPGA family (Spartan) has comparable number of Configurable Logic Blocks (CLBs) with resource-rich family (Virtex), but less memory, multipliers, and routing tracks. Limitation on interconnect resources increases the probability of nets being routed through longer paths and nets becoming unroutable due to congestion. For the sake of routability, when nets go through longer paths, critical path delay may also increase. We may solve these problems by migrating to the resource-rich FPGA device which has more routing resources by paying 7× price. In order to avoid this, FPGA CAD flow must improve the routability as well as timing performance to make the low-cost device a feasible option.

FPGA CAD flow includes four stages: technology *mapping* to form a netlist of logic blocks, *clustering* to combine blocks into CLBs, *placement* to allocate physical positions to each CLB, and *routing* to define paths for all nets in the design. Clustering is the foundation of layout and has strong influence on area efficiency, timing, and power [1]. Figure 1 categorizes clustering techniques. Based on the target utilization objectives, we identify two types of clustering techniques: targeting maximum logic utilization and targeting less than maximum logic utilization. Most clustering approaches fully populate CLBs with optimization goal for routability (area), timing, or power.

However, maximum logic utilization may cause routing congestion in some parts of the FPGA. A CLB contains N basic logic elements (BLEs), where a typical BLE used in many academic studies is formed of a 4-input LUT, a flip-flop, and a MUX to choose the output from either the LUT or the flipflop. A group of CLBs is strongly connected if they share a large number of nets. After placement, such CLBs appear close together in a specific region on the FPGA. Filling these CLBs to the limit (N) increases the demand on the interconnect resources through this region to be able to route the connections among them. As a result, channel width requirements for such regions become higher than others. This leads to an increase in peak channel width and hence the design requires more routing resources.

Table 1: Features and prices of two xilinx FPGA families (spartan: low-cost and virtex: resource-rich).

Device	CLBs	RAM	Multi- pliers	Interconnect dble.+hex+1 g.[1]	Price (2008)
Spartan 3s500e	1164	36 k	20	8 + 8 + 24	$23
Virtex 2vp7	1232	792 k	44	40 + 120 + 24	$179
Spartan 3s1200e	2168	504 k	28	8 + 8 + 24	$42
Virtex 2vp20	2320	1584 k	88	40 + 120 + 24	$304

[1]dble.:double, hex, lg.: long (types of wires).

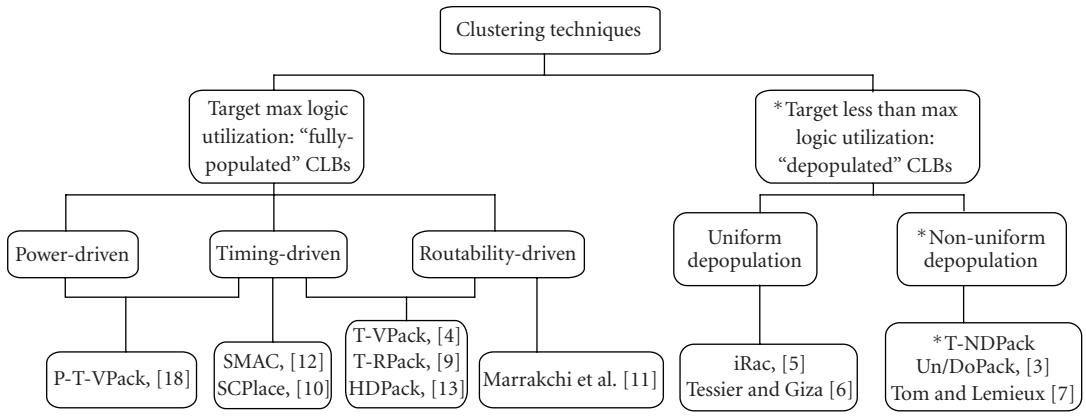

Figure 1: Categorization of clustering techniques based on logic utilization approach and optimization goals. *represents the category of our technique (T-NDPack).

It has long been known that, as CLBs are depopulated, better channel widths can be achieved. First proposed in [2], the depopulation-based clustering techniques can lower peak channel width and improve routability. Instead of targeting maximum logic utilization, depopulation is a technique that underuses CLBs by not filling them to capacity. The regions with strongly connected CLBs are spread over a larger area on the FPGA. This reduces the demand for routing resources; hence in such a region, more resources become available to route the connections.

However depopulation leads to more number of external connections among CLBs and typically results with an increase in critical path delay, because the inter-CLB delays are much larger than the intra-CLB delays [1]. For example, the latest depopulation-based clustering technique [3] decreases minimum channel width by 15% while increasing the number of CLBs by 17.32% compared to T-VPack [4]. Additionally, total area increases by 5%, along with 7% increase in critical path delay. All depopulation-based clustering algorithms [3, 5, 6] increase critical path delay, while enhancing the routability.

In this paper, we propose the first depopulation-based clustering approach that takes timing into account. Categorized in Figure 1, we develop a seed-based routability and timing-driven nonuniform depopulation technique, T-NDPack. We adjust the CLB capacity under construction based on the criticality of the BLE under consideration. For example, we cluster the nets on the critical path to full capacity. That way, we reduce the inter-CLB delay which helps decrease critical path delay. Meanwhile, we depopulate on the paths with low criticality to avoid routing congestion and hence reduce the channel width requirements. To achieve this idea, we modify both the algorithm flow and cost function of T-VPack. Results show that T-NDPack decreases minimum channel width by 11.07% while increasing the number of CLBs by 13.28% compared to T-VPack. More importantly, as opposed to the trend we see in other depopulation techniques, T-NDPack decreases critical path delay by 2.89%. With the new technique, instead of moving to a resource-rich FPGA (in the case of Spartan 3s500e versus Virtex 2vp7 of Table 1), designers may, for example, move to Spartan 3s1200e and pay 2 × instead of 7 × cost. Furthermore, this paper stands as a guide when it comes to understanding the effects of depopulation on area and delay performance for FPGAs.

Rest of the paper is organized as follows. Section 2 presents the review of the related work on depopulation-based clustering techniques. Section 3 introduces our clustering technique, T-NDPack. Section 4 presents and analyzes our experimental results. Section 5 compares our work with both depopulation- and nondepopulation-based approaches. Section 6 presents our conclusion and future work.

2. Related Work

Several depopulation techniques were proposed previously. We categorize them into two types (Algorithm 1): uniform depopulation [5, 6] and nonuniform depopulation [3, 7]. Uniform depopulation sets a fixed "upper limit" per CLB and each CLB is filled to that "upper limit" capacity. In nonuniform depopulation, the "upper limit" varies

among CLBs. Let us assume that cluster size is 8. While a uniform depopulation scheme may use a fixed "upper limit" of 6 for all CLBs, a nonuniform scheme will result in a CLB distribution with sizes from 1 to 8. nonuniform depopulation sets a very low "upper limit" to prioritize routability for a congested area and sets a high "upper limit" for the congestion free area to save more CLBs. Therefore, nonuniform depopulation results with better routability in congested area and higher CLB utilization in uncongested area compared to uniform scheme.

Tom and Lemieux [7] proposes the first nonuniform depopulation methodology. Tom uses 20 MCNC benchmark circuits [8] and connects them with three different topologies (independent, pipelined, and clique). Each topology represents an SoC. Each benchmark is an IP block and uses its own "upper limit." Results show that the SoC design with the help of depopulation technique requires less channel width compared to T-VPack. However, total area increases while maintaining similar critical path delay relative to T-VPack. Tom's approach [7] stands as a good study in terms of showing the potential benefit of nonuniform depopulation. However, the methodology determines the "upper limit" for each IP block manually based on the congestion inside the same IP block and no algorithm is given.

The nonuniform depopulation technique, Un/DoPack, was proposed by Tom et al. in [3]. This technique runs the FPGA CAD flow twice. First iteration is the regular CAD flow. In the second iteration, clustering stage uses the layout result of the first iteration and depopulates the congested regions. While reducing the channel width, Un/DoPack, similar to the other depopulation-based clustering approaches, observes an increase in total area and critical path delay.

3. T-NDPack

In this section, we describe our seed-based routability and timing-driven nonuniform depopulation clustering technique, T-NDPack. We present the pseudocode and notable implementation insights.

3.1. Algorithm Flow. T-NDPack chooses the seed block based on criticality first. The first block that is clustered into a CLB is called the seed block of this CLB. Then T-NDPack packs more blocks into the CLB by following the nonuniform depopulation clustering scheme.

We define two strategies for depopulation:

(i) BLE-limit: limit the number of BLEs used in a CLB [3, 6, 7],

(ii) input-limit: limit the number of inputs used for a CLB [5].

T-NDPack employs either "BLE-limit" or "input-limit" strategy to achieve variable utilization level. We evaluate the performance of each and present their effect on minimum channel width and critical path delay separately. In this paper, the "utilization level" measures the amount of resources used by a CLB in terms of the number of BLEs or inputs where "high utilization level" means most resources are used. For the "BLE-limit" strategy, utilization level refers to the number of BLEs used in a CLB, and for the "input-limit" strategy, utilization level refers to the number of inputs used by a CLB.

Table 2: A maximum utilization table (MUT).

The ranking percentage of the seed's criticality	Maximum utilization level	Cluster size
90%–100%	8	8
40%–90%	7	8
0%–40%	6	8

Algorithm 1 shows the pseudocode for T-NDPack. First, the algorithm computes the criticality of each block (Ln. 1) and sorts them based on their criticality (Ln. 2).

Then T-NDPack begins to fill CLBs. Algorithm keeps clustering blocks into CLBs until no unclustered block is left (Ln. 3). In each iteration, we have the following.

(1) T-NDPack packs the seed block that has the maximum criticality (Ln. 4). We regard that the criticality of the seed block represents the criticality of the net. Therefore, we determine the "maximum utilization level" based on the ranking of the seed's criticality value (Ln. 5). In this paper, the "maximum utilization level" means the maximum number of BLEs or inputs that are allowed to be used for a CLB. If the ranking of the seed's criticality value is high, algorithm sets the "maximum utilization level" to a high value to decrease the critical path delay. Nevertheless, if the ranking is low, the "maximum utilization level" is set to a low value to reduce the routing requirement.

Table 2 shows a "*maximum utilization table" (MUT)* used to look up the value of "maximum utilization level" for "BLE-limit" strategy. This MUT allows more BLEs to be used in a CLB for the seed with 95% criticality ranking than the seed with 50% criticality ranking. We explain how to generate MUT in Section 4.2.

(2) Then, T-NDPack starts to cluster blocks, till the CLB under consideration reaches its maximum utilization level (Ln. 6). Figure 2 shows the algorithm flow for packing one block into the CLB (Ln. 7–Ln. 20).

(a) T-NDPack builds a candidate block list (Ln. 7) by comparing the criticality of blocks with respect to the "*candidate block threshold" (CBT)*. CBT increases as the "current utilization level" of the CLB under consideration increases. We define "current utilization level" as the number of currently used BLEs or inputs for the CLB under consideration. The candidate block list includes all the blocks whose criticality is above the CBT value. If the CLB is already highly utilized, only the blocks with high criticality are allowed to be clustered. Furthermore, if the list is empty, T-NDPack stops clustering more blocks into this CLB (Ln. 8–Ln. 10).

(b) Then a cost function, described in Section 3.2, computes the gain value for each block in the candidate block list.

```
(1) Compute the criticality of each block
(1) Sort the blocks with criticality
(3) While unclustered blocks available
(4)    Find a seed with maximum criticality
(5)    Determine the maximum utilization level in the cluster based on the ranking of the seed's criticality
       through looking up "maximum utilization table" (MUT)
(6)    While maximum utilization level is not reached
(7)       Build the candidate block list that criticality > "candidate block threshold" (CBT) (CBT is
          determined by the current utilization level)
(8)       If candidate block list is empty
(9)          Break
(10)      End if
(11)      If the related blocks available
(12)         Choose a related block with highest gain from the candidate block list
(13)      Else if (current utilization level < "unrelated block threshold" (UBT))
(14)         Choose a unrelated block with the highest gain from the candidate block list //unrelated
             block Clustering
(15)      Else
(16)         Break
(17)      End if
(18)      Remove block from unclusered block list
(19)      Add block to current cluster
(20)      Current clustered block number ++
(21)   End while
(22) End while
```

Algorithm 1: The pseudocode for T-NDPack.

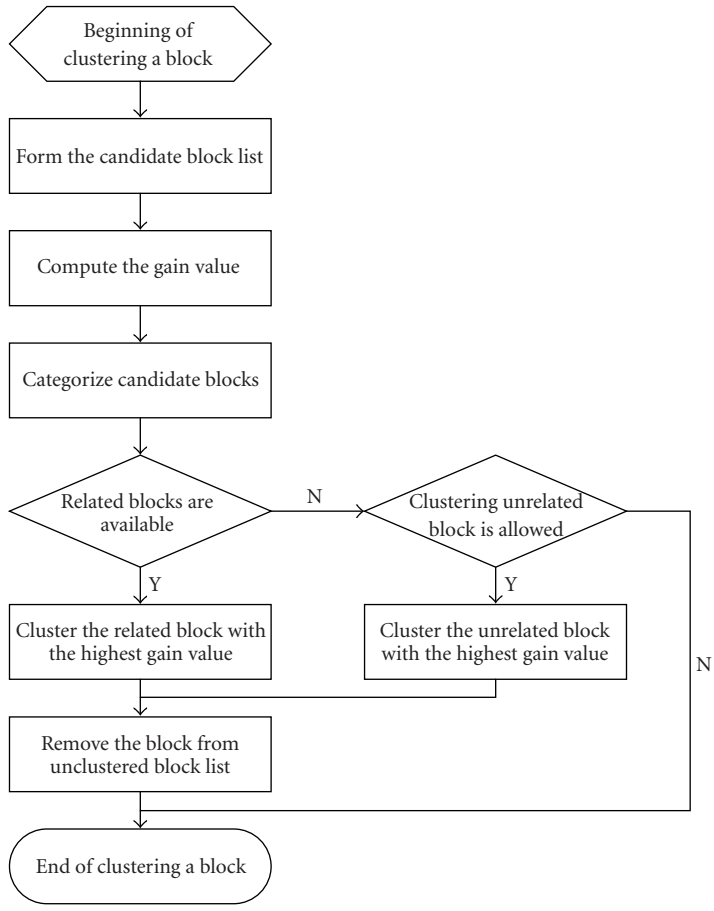

Figure 2: Algorithm flow for clustering a candidate basic logic element (BLE) into configurable logic block (CLB).

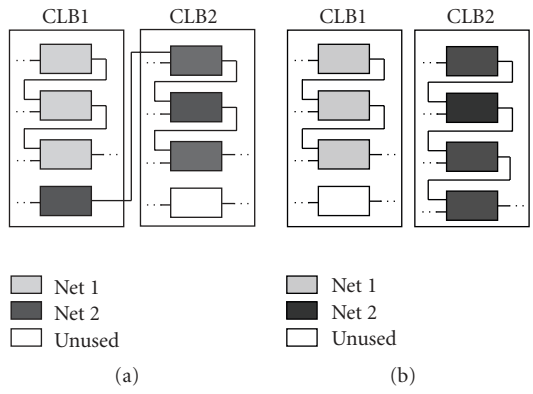

Figure 3: Impact of unrelated block clustering, Net 1 followed by Net 2. (a) introduces an inter-CLB delay and an external connection between CLB1 and CLB2. (b) no inter-CLB delay, no external connection, and less routing resource demand when CLB1 is underutilized.

(c) Next, we categorize the blocks in the candidate block list into related and unrelated blocks: a block is related if it shares inputs or outputs with the current CLB under consideration.

(d) T-NDPack tries to cluster the related block with the highest gain value first (Ln. 11-Ln. 12). If the related block is not available and clustering the unrelated blocks is allowed, T-NDPack clusters the unrelated block with the highest gain value (Ln. 13-Ln. 14). We explain this mechanism in Section 3.3.

(e) Finally, T-NDPack removes the block from the unclustered block list with the next iteration.

3.2. Cost Function. The cost function in T-NDPack considers the criticality in terms of delay and routability simultaneously (1) similar to the clustering cost function of the T-VPack [4]. The "α" parameter balances the criticality and the routability. The criticality is defined in [4] and calculated based on the sensitivity of a connection to the delay of the whole circuit. T-NDPack introduces the current utilization level as a factor to the routability component in the clustering cost function of the T-VPack. As current utilization level increases, the probability of sharing inputs and outputs increases. Therefore the value of the routability component increases. T-NDPack gradually scales more on routability part to provide informed attention to criticality:

$$\text{gain} = \alpha * \text{criticality} + (1-\alpha) * \left(\frac{|\text{NetA} \cap \text{NetB}|}{G}\right) \Big/ \text{current utilization level}. \quad (1)$$

3.3. Unrelated Block Clustering. In this section, we explain how to cluster an unrelated block. T-NDPack tries to cluster the related block with the highest gain value first. If no related block is available and only if the current utilization level is less than the "*unrelated block threshold*" (UBT), T-NDPack allows clustering the unrelated block (Ln. 13 - Ln. 14). This rule avoids clustering very few unrelated blocks and the possible inter-CLB delay. Also, this rule reduces the connections between CLBs to improve routability. For example, as shown in Figure 3, we want to cluster two nets into two CLBs, in the order of Net 1 followed by Net 2. In Figure 3(a), after Net 1 is clustered, a block of Net 2 is clustered in CLB 1. This introduces an inter-CLB delay for Net 2 and a connection between CLB 1 and CLB 2. As an alternative solution, in Figure 3(b), all blocks of Net 2 are clustered in CLB 2. In this solution, there is no inter-CLB delay or connection between CLBs. Compared to Figure 3(b), the solution in Figure 3(a) requires more routing resources and has a larger delay. Therefore if few available BLEs are left in a CLB and related block is not available, it is wiser to leave the BLEs unused.

Typically, clustering techniques modify the cost function ([4, 5, 9]) or the algorithm flow [10] or both ([11–13]). Here we summarize in what capacity the well-known approaches enhance the clustering flow and highlight where our approach stands relative to them.

T-RPack [9] uses the same algorithm flow as T-VPack and modifies the cost function. T-RPack modifies the routability part in the cost function by taking into account the individual contributions of both shared and nonshared nets between the CLB under construction and the block under consideration. T-RPack improves minimum channel width compared to T-VPack.

iRAC [5] develops a new method to choose the seed block. This technique chooses the unclustered block with the most used inputs and minimum connectivity as the seed block. iRAC then clusters each BLE into the CLB under construction using a new cost function that is based on the weight of the intersecting net and its pins that are already in the CLB. Furthermore, it uses the uniform depopulation with input-limit strategy. The algorithm flow is similar to T-VPack. However, with the modifications, iRAC achieves large reduction in the number of external nets which leads to reduction in minimum channel width.

The latest clustering technique, HDPack [13], uses a global placer to determine approximate BLE locations. Then the algorithm uses this placement information (physical information) in the clustering cost function. HDPack further incorporates a prepacking step. However, major contribution for improvement is based on clustering with the usage of physical information. The prepacking step leads to little improvement over the modified cost function.

In summary, as shown in Algorithm 1 and Figure 2, we adjust the cost function of T-VPack to pay informed attention to routability and timing by taking utilization level into account. We also modify the clustering algorithm significantly by

(i) adjusting the "maximum utilization level" at run time with maximum utilization table (MUT);

(ii) forming the "candidate block list" with candidate block threshold (CBT);

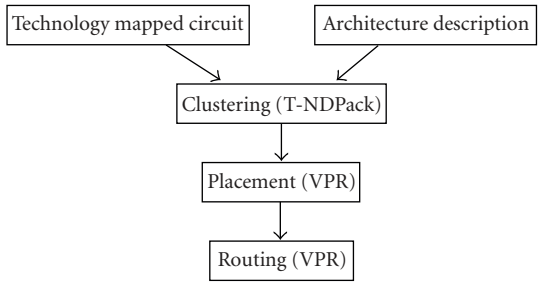

Figure 4: CAD flow.

Table 3: FPGA architecture parameters.

Architecture feature	Value
LUT inputs	4
CLB size	8
Inputs/CLB	18
Segment length	1
Fc	0.5
Fs	3

(iii) setting the "unrelated block threshold" (UBT) for clustering.

4. Experimental Results

4.1. Methodology. We implement T-NDPack based on T-VPack and conduct several experiments with the 20 largest MCNC benchmarks. We examine the performance of our proposed clustering technique and explore the effects of two depopulation strategies ("*BLE-limit*" and "*input-limit*"). Table 3 lists the main architecture parameters that we used in the experiments where segment length is the number of CLBs that a wire length spans, Fc describes the flexibility of connection blocks, and Fs describes the flexibility of switch blocks [14]. Figure 4 shows the CAD flow. The VPR version used in the experiments is v4.30.

As opposed to [3], our method runs the CAD flow once. Technology-mapped circuit and the architecture description are the inputs to the clustering stage. T-NDPack carries out clustering and VPR [15] handles placement and routing. We obtain the number of used CLBs, minimum channel width, and critical path delay for performance comparison against [4, 5, 9, 13].

4.2. Tuning the Parameters for BLE-Limit. We tune various parameters in our algorithm to identify the configuration which gives the best performance. In order to find the suitable value of "α", "UBT", "MUT", and "CBT", we performed a set of experiments following the CAD flow described in Section 4. Here we only discuss the parameter tuning study based on "BLE-limit" strategy. The parameter values for "input-limit" strategy rely on the observations on the "BLE-limit." We will discuss this in Section 4.3.

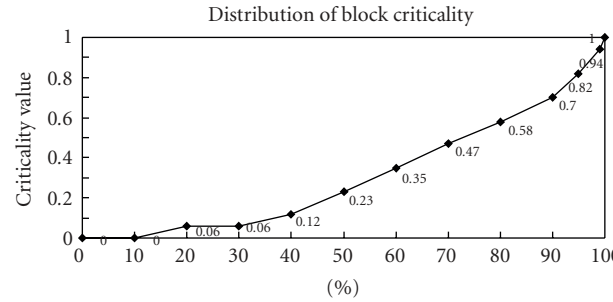

Figure 5: Distribution of CLBs based on criticality value: CLBs are uniformly distributed between 40% and 96% forming the range of Maximum Utilization Table (MUT).

Table 4: Maximum utilization table (MUT) with four partitions shows three examples for potential range configurations. configuration in table (b) results with good quality measurements; then we fine tune the range and form table in (c).

(a)

The ranking percentage of the seed's criticality	Maximum utilization level
95%–100%	8
40%–95%	7
0%–40%	6

(b)

The ranking percentage of the seed's criticality	Maximum utilization level
95%–100%	8
50%–95%	7
0%–40%	6

(c)

The ranking percentage of the seed's criticality	Maximum utilization level
95%–100%	8
51%–95%	7
0%–40%	6

(i) α: This coefficient balances the tradeoff between routability and delay. Marquardt et al. [4] shows that the value of 0.75 results with best area and delay efficient design. We believe that the behavior of the α value in our cost function is similar to [4]. Therefore, we varied α within 0.6, 0.7, and 0.75 in our experiments.

(ii) "UBT": Unrelated block threshold is used for allowing an unrelated block to be clustered into CLB. We assigned the values of 2, 4, and 6 for this parameter. During our preliminary experiments, we observed that a large value led to CLBs with too many unrelated BLEs, whereas a small value led to under utilization of CLBs. Therefore we fix UBT to 4.

(iii) "MUT": Maximum utilization table is used for setting the maximum utilization level for a CLB. We

```
(1) For (each α value)
(2)   For (each MUT)
(3)     For (each CBT)
(4)       For (each benchmark)
(5)         Run T-NDPack to obtain the number of used CLBs
(6)         Run VPR to obtain the minimum channel width and critical
            path delay
(7)       End for
(8)       Caculate the average number of used CLBs, minimum channel
          width and critical path delay
(9)     End for
(10)  End for
(11) End for
```

Algorithm 2: Experiments for identifying best configuration values for α, MUT, and CBT (note that UBT is set to be 4).

divide 0% to 100% range into 2 to 5 partitions. The maximum utilization level for the partition with the highest range is set to be the CLB capacity, 8, and this value descends by 1 relative to the ranking. Figure 5 shows the criticality value distribution of netlist "elliptic." We observe that 40% of the CLBs have criticality less than 0.12, and afterwards, criticality value is more or less evenly distributed till 0.82 criticality value (40% to 95% range). We also observe that few CLBs have criticality value larger than 0.82. We capture the nature of this distribution in MUT. Firstly, we set the upper boundary of the partition with lowest range near 40% and the lower boundary of the partition with highest range near 95%. We then partition 40% to 95% evenly based on the MUT size. Table 4(a) shows an MUT with three partitions. We also adjust the boundary by 5% or 10% to derive alterative MUTs as shown in Table 4(b). If any of the MUTs results with a good performance, we fine tune that MUT by adjusting the boundary by 1% or 3% (Table 4(c)). If not, we continue adjusting the boundary by 5% or 10%. After finding the MUT configuration that results with a good performance, it is fixed and used for all benchmarks. We do not use a different MUT for each benchmark.

(iv) "CBT": Candidate block threshold is used for allowing clustering a block into a CLB based on its criticality. CBT ranges from 0 to 1. If the current utilization level of the CLB is low (the number of BLEs used at that time for this CLB is smaller than 6), then we do not take criticality into account, and set CBT to be 0 to focus on routability. Otherwise we set CBT to be 0.2 or 0.4 for current utilization level of 6 and set CBT to be 0.8 or 0.9 for current utilization level of 7.

4.3. Effect of BLE-Limit and Input-Limit Exploration. As shown in Algorithm 2, we sweep through α, MUT, and CBT within their predefined ranges to evaluate the "BLE-limit" strategy. For each configuration, we run the clustering algorithm over 20 MCNC benchmarks and compute averages for the number of used CLBs, minimum channel width and critical path delay. Figure 6 shows the minimum channel width and critical path delay reduction of T-NDPack with "BLE-limit" relative to T-VPack. "x-axis" shows the increase in the number of CLBs and "y-axis" shows the reduction in minimum channel width and critical path delay. For each configuration, we generate two data points: a triangle representing channel width reduction and a diamond representing critical path delay reduction. We show each pair of data points (a triangle and a diamond) with a link indicating that they use the same parameter configuration. We then draw solid lines passing through the data points resulting with best reduction value in channel width and critical path delay separately. We then label the points on the line with solid triangle and diamond. We will use these solid points for analysis in Section 4.4.

For the "input-limit" strategy, instead of sweeping all parameters, we choose sample points from Figure 6 that are on the best-line (solid triangle and diamond points) and run them with "input-limit" constraint. In our experiments, cluster size (N) and the number of inputs per CLB (I) hold $I = 2N+2$ expression, which generates the best area and delay

Table 5: Conversion of parameters for input-limit strategy.

(a) Maximum Utilization Table (MUT) conversion.

The ranking percentage of the seed's criticality	Maximum utilization level (BLE-limit)	Maximum utilization level (input-limit)
95%–100%	8	18
40%–95%	7	16
0%–40%	6	14

(b) Candidate Block Threshold (CBT) conversion.

Current utilization level (BLE-limit)	Current utilization level (input-limit)	CBT
7	16	0.8 or 0.9
6	14	0.2 or 0.4
0–5	0–12	0

Table 6: α, UBT, MUT, CBT: best configuration for performance analysis of T-NDPack.

Parameter	Value	
α	0.65	
UBT	4	
MUT	The ranking percentage of the seed's criticality	Maximum utilization level
	95%–100%	8
	45%–95%	7
	0%–45%	6
CBT Table	Current utilization level	CBT
	7	0.9
	6	0.2
	0–5	0

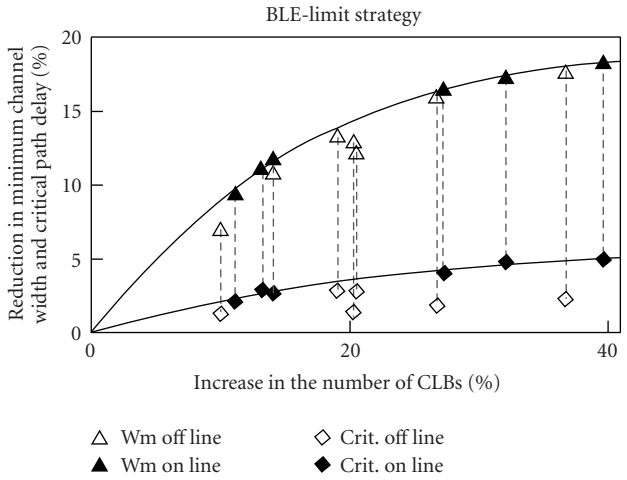

Figure 6: Minimum channel width (Wm) and critical path delay (crit) reduction of T-NDPack with "*BLE-limit*" strategy relative to T-VPack. Triangles represent channel width savings of T-NDPack relative to T-VPack. Diamonds represent critical path delay savings. Solid line represents best line drawn over the data points. Points that are on the solid line are filled (Wm on line and crit. on line). Dotted line represents triangle and diamond data points collected based on the same experimental configuration.

Table 7: Execution time over 20 MCNC benchmarks on pentium iv, core 2 Duo, 2.8 GHz with 4 GB RAM. timing measurements are based on running experiments ten times and taking their average.

Clustering algorithm	Clustering	Clustering + placement + routing
T-VPack	0.4 seconds	30 minutes 41 seconds
T-NDPack	0.56 seconds	32 minutes 58 seconds

product [16]. (As Table 3 shows, $N = 8$ and $I = 18$ in our architecture.) Therefore, we use this relationship and adjust the MUT and CBT used for "BLE-limit" to accommodate "input-limit" strategy as shown in Table 5 (converted based on Table 4(a)). Similarly, we adjust UBT to 10. Figure 7 shows the minimum channel width and critical path delay reduction of T-NDPack for the "input-limit" strategy. We will also use the solid points on this chart for analysis in Section 4.4.

Table 8: T-NDPack versus other clustering techniques: based on reported performance over T-VPack (positive percentage means improvement), #CLBs: increase in number of CLBs; Wm: minimum channel width reduction; Crit.: critical path delay reduction.

Clustering tech.	# CLBs	Wm	Crit.
[1]iRAC + iRAP	−8.78%	25.09%	N.A.
[2]Un/DoPack	−17.32%	15%	−7%
[3]Un/DoPack	−52.78%	40%	−20%
[4]T-NDPack	−13.28%	11.07%	2.89%
[5]T-NDPack	−27.31%	16.31%	4.13%

[1]Not comparable because of different placement tool. [2 3]Un/DoPack in moderate and aggressive amounts of depopulation, respectively. [4 5]T-NDPack in moderate and aggressive amounts of depopulation, respectively.

4.4. Evaluation of BLE-Limit and Input-Limit. Based on Figures 6 and 7, we tune various parameters in our algorithm to identify the good configurations whose performances are shown in Figures 8 and 9. Figure 8 shows reduction in minimum channel width and Figure 9 shows reduction in critical path delay for T-NDPack with respect to T-VPack based on "BLE-limit" and "input-limit" strategies, respectively. Solid line represents "BLE-limit" strategy and dashed line represents "input-limit" strategy. Each point with the same x-value in Figures 8 and 9 is generated with the same configuration of the parameters. Figures 8 and 9 show the following.

(i) As the number of CLBs increases, we observe a reduction in both channel width and critical path delay.

(ii) The amount of channel width and critical path delay savings gradually decreases as the number of CLBs increases.

Furthermore, we observe that the "BLE-limit" strategy is better than the "input-limit" strategy. We see a couple of reasons for this behavior. For example, if the criticality of the seed block is high, algorithm sets a high value for

Table 9: Comparison between T-NDPack and T-VPack on the critical path delay, minimum channel width, and number of used CLBs metrics over 20 MCNC benchmark circuits (positive percentage means improvement).

	Critical pathdelay(10^{-8}s)			Minimum channel width			Number of used CLBs		
	T-VPack	T-NDPack	Change	T-VPack	T-NDPack	Change	T-VPack	T-NDPack	Change
alu4	3.47	3.61	−4.03%	41	37	9.76%	193	219	−13.47%
apex2	4.30	4.46	−3.88%	58	49	15.52%	240	272	−13.33%
apex4	4.23	4.43	−4.73%	58	55	5.17%	165	183	−10.91%
bigkey	2.31	2.06	10.75%	28	27	3.57%	214	233	−8.88%
clma	10.5	7.84	25.72%	72	64	11.11%	1055	1234	−16.97%
des	4.42	5.12	−15.86%	25	21	16.00%	200	227	−13.50%
diffeq	4.19	4.13	1.24%	34	27	20.59%	189	220	−16.40%
dsip	2.52	2.07	18.01%	24	23	4.17%	172	179	−4.07%
elliptic	6.56	5.82	11.28%	53	45	15.09%	454	511	−12.56%
ex1010	6.80	6.60	2.81%	63	56	11.11%	601	679	−12.98%
ex5p	3.96	4.36	−10.30%	55	47	14.55%	138	154	−11.59%
frisc	7.96	7.30	8.25%	59	54	8.47%	446	514	−15.25%
misex3	4.00	3.47	13.16%	46	42	8.70%	179	203	−13.41%
pdc	6.61	6.47	2.11%	88	74	15.91%	582	659	−13.23%
s298	7.50	7.09	5.40%	34	33	2.94%	243	274	−12.76%
s38417	4.89	4.98	−1.85%	43	41	4.65%	802	947	−18.08%
s38584	4.17	4.15	0.54%	45	41	8.89%	806	946	−17.37%
seq	3.87	4.20	−8.39%	51	43	15.69%	221	252	−14.03%
spla	5.81	5.45	6.24%	67	63	5.97%	469	535	−14.07%
tseng	4.21	4.15	1.31%	34	26	23.53%	133	150	−12.78%
Ave.			2.89%			11.07%			−13.28%

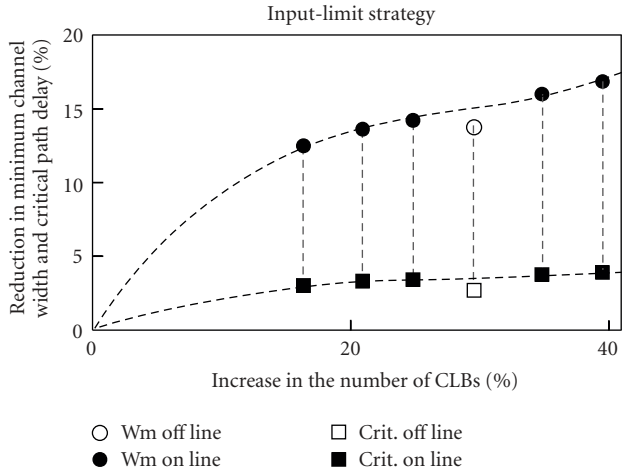

Figure 7: Minimum channel width (Wm) and critical path delay (crit) reduction of T-NDPack with "*input-limit*" strategy relative to T-VPack. Circles represent channel width savings of T-NDPack relative to T-VPack. Squares represent critical path delay savings. Dashed line represents best line drawn over the data points. Points that are on the line are filled (Wm on line and crit. on line). Dotted line represents circle and square data points collected based on the same experimental configuration.

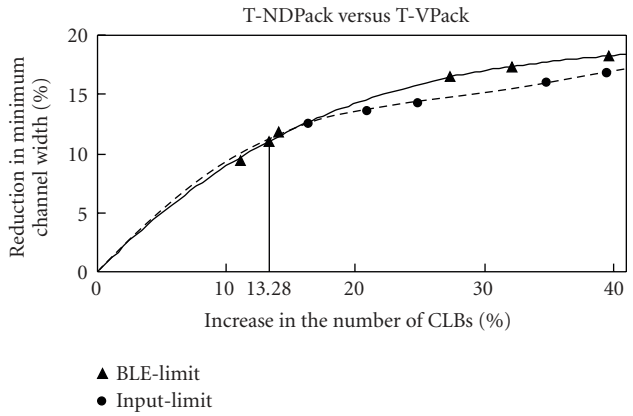

Figure 8: "*x*-axis": increase in the number of CLBs, "*y*-axis": reduction in minimum channel width relative to T-VPack. Solid line represents: T-NDPack with "BLE-limit" strategy. Dashed line represents: T-NDPack with "input-limit" strategy. As number of CLBs increase, minimum channel width decreases for both strategies.

the maximum utilization level of the CLB under construction (e.g., 16 out of 18 inputs). This affects the logic utilization significantly in a CLB. We observed cases like usage of 4 out of 8 BLEs. In another case, for a seed that has low criticality, our algorithm allows 12 inputs for that CLB. However, due to the input sharing, most of the inputs were absorbed (6 BLEs). Therefore "input-limit" technique in some cases worked against the objective of depopulation technique.

Based on these observations, we choose "BLE-limit"-based technique for performance comparison against other

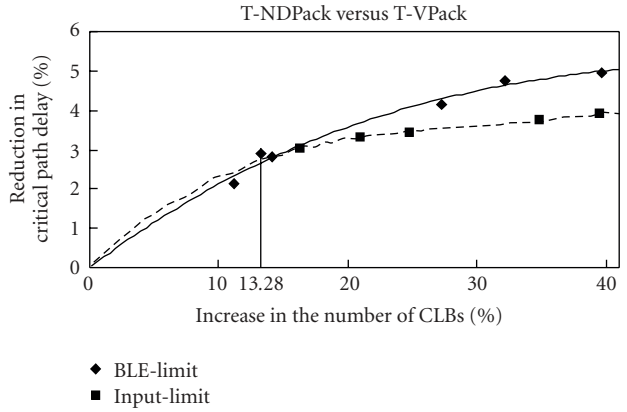

Figure 9: "x-axis": increase in the number of CLBs; "y-axis": reduction in critical path delay relative to T-VPack. Solid line represents: T-NDPack with "BLE-limit" strategy. Dashed line represents: T-NDPack with "input-limit" strategy. As number of CLBs increase, critical path delay decreases for both strategies.

clustering techniques. As shown in Figure 8, the channel width increases along with an increase in the number of CLBs. We decompose total area into logic and routing and use (2) as a model to derive the area estimate. In this paper, we regard 70% for routing area as a good estimation for the commercial FPGAs [5]. As used in [3], let "L" be the number of CLBs and let "W" be the channel width, then

$$\frac{\text{Area}_{\text{new}}}{\text{Area}_{\text{old}}} = 0.3 * \frac{L_{\text{new}}}{L_{\text{old}}} + 0.7 * \frac{W_{\text{old}}}{W_{\text{new}}} * \frac{L_{\text{new}}}{L_{\text{old}}}, \qquad (2)$$

where new represents after depopulation, and old represents before depopulation.

Among the points in Figures 8 and 9, we find that 13.28% average increase in the number of CLBs is the data point that leads to the best area-delay product. Table 6 shows the parameter values used for this data point. We run 20 MCNC benchmarks with the configuration parameters shown in Tables 3 and 6. We then compare minimum channel width, critical path delay, and the number of CLBs with T-VPack in Table 9. On average T-NDPack reduces minimum channel width by 11.07%. This results with 4.50% area increase. On average, the critical path delay decreases by 2.89%. Spreading the logic among the available CLBs is expected to increase the critical path delay. We observe this trend for some of the benchmarks with T-NDPack; however for most of the benchmarks we observe a reduction in critical path delay.

4.5. Run-Time. Table 7 compares T-VPack and T-NDPack based on the time it takes to run the clustering stage for all 20 MCNC benchmarks. Adjusting the level of depopulation-based on the criticality contributes to the execution time; therefore T-NDPack increases the run-time of the clustering stage on average by 0.16 seconds. However, this overhead is minor when the execution time for the CAD flow is considered. Since T-NDPack generates more number of CLBs to be placed and routed, we also observe an increase in the execution time for the placement and routing stages.

5. Discussion

In this section, we compare T-NDPack with other depopulation-based state-of-the-art clustering techniques and Table 8 summarizes it.

Un/DoPack [3] is a nonuniform depopulation technique. Un/DoPack achieves up to 40% channel width reduction through aggressive depopulation with a critical path delay penalty of 20%. In contrast, T-NDPack reduces critical path delay as the intensity of the depopulation increases. The trend line in Figure 8 shows that T-NDPack can further improve on channel width and continue reducing the critical path delay by using more CLBs (e.g., T-NDPack[4] versus T-NDPack[5] in Table 8). However, this leads to a significant area penalty which may prevent the designer from mapping the design onto a low-cost FPGA.

iRAC [5] is a routability driven uniform depopulation clustering technique. iRAC achieves 25.09% reduction in channel width. However, [5] reports its results based on a different placement algorithm, iRAP, which reduces channel width over VPR. We use VPR for the placement. Since neither iRAC nor iRAP is publicly available, it is not feasible to make a fair comparison without implementing their algorithms. iRAC [5] does not report timing results. It is also not feasible to reach a conclusion on overall performance without considering area and delay simultaneously.

6. Conclusion and Future Work

It has long been known that, as CLBs are depopulated, better channel widths can be achieved. However depopulation leads to more external connections among CLBs and typically results with an increase in critical path delay. While enhancing routability through depopulation is essential for utilizing the low-cost FPGAs, at the same time there is a need for addressing the critical path delay. We achieve this goal with T-NDPack by adjusting the capacity of the CLB under construction based on the criticality of the logic block under consideration.

In this study, we show that the depopulation-based clustering techniques while reducing the stress on routing can also achieve reduction in critical path delay. This is significant as this study shows that depopulation-based clustering potentially allows the designer to stay with the low-cost FPGA family instead of migrating to the costly resource-rich FPGA family.

In [17, 18], Pandit introduces a wirelength prediction techni-que that accurately estimates postplacement individual wirelength information for a given netlist before the clustering stage. As future work, we plan to incorporate this mechanism into our clustering cost function to further improve the performance of the T-NDPack.

References

[1] A. Marquardt, V. Betz, and J. Rose, "Speed and area tradeoffs in cluster-based FPGA architectures," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 8, no. 1, pp. 84–93, 2000.

[2] A. DeHon, "Balancing interconnect and computation in a reconfigurable computing array," in *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA '99)*, pp. 69–78, Monterey, Calif, USA, 1999.

[3] M. Tom, D. Leong, and G. Lemieux, "Un/DoPack: re-clustering of large system-on-chip designs with interconnect variation for low-cost FPGAs," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD '06)*, pp. 680–687, San Jose, Calif, USA, November 2006.

[4] M. Marquardt, V. Betz, and J. Rose, "Using cluster-based logic blocks and timing-driven packing to improve FPGA speed and density," in *Proceedings of the ACM/SIGDA 7th International Symposium on Field Programmable Gate Arrays (FPGA '99)*, pp. 37–46, Monterey, Calif, USA, February 1999.

[5] A. Singh and M. Marek-Sadowska, "Efficient circuit clustering for area and power reduction in FPGAs," in *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA '02)*, pp. 59–66, Monterey, Calif, USA, 2002.

[6] R. Tessier and H. Giza, "Balancing logic utilization and area efficiency in FPGAs," in *Proceedings of the International Workshop on Field Programmable Logic and Applications (FPLA '00)*, pp. 535–544, Villach, Austria, 2000.

[7] M. Tom and G. Lemieux, "Logic block clustering of large designs for channel-width constrained FPGAs," in *Proceedings of the Design Automation Conference*, pp. 726–731, Anaheim, Calif, USA, 2005.

[8] S. Yang, "Logic synthesis and optimization bench-marks, version 3.0," Tech. Rep., Microelectronics Center of North Carolina, 1991.

[9] E. Bozorgzadeh, S. O. Memik, X. Yang, and M. Sarrafzadeh, "Routability-driven packing: metrics and algorithms for cluster-based FPGAs," *Journal of Circuits, Systems and Computers*, vol. 13, no. 1, pp. 77–100, 2004.

[10] G. Chen and J. Cong, "Simultaneous timing driven clustering and placement for FPGAs," in *Proceedings of the International Conference on Field Programmable Logic and Applications (FPLA '04)*, pp. 158–167, Antwerp, Belgium, August 2004.

[11] Z. Marrakchi, H. Mrabet, and H. Mehrez, "Hierarchical FPGA clustering to improve routability," in *Proceedings of IEEE International Conference on Reconfigurable Computing and FPGAs*, Puebla, Mexico, 2005.

[12] J. Y. Lin, D. Chen, and J. Cong, "Optimal simultaneous mapping and clustering for FPGA delay optimization," in *Proceedings of the Design Automation Conference*, pp. 472–477, San Francisco, Calif, USA, 2006.

[13] D. T. Chen, K. Vorwerk, and A. Kennings, "Improving timing-driven FPGA packing with physical information," in *Proceedings of the International Conference on Field Programmable Logic and Applications (FPL '07)*, pp. 117–123, Amsterdam, The Netherlands, August 2007.

[14] V. Betz, J. Rose, and A. Marquardt, *Architecture and CAD for Deep-Submicron FPGAs*, Kluwer Academic Publishers, Dodrecht, The Netherlands, 1999.

[15] V. Betz and J. Rose, "VPR: a new packing placement and routing tool for FPGA research," in *Proceedings of the International Workshop on Field-Programmable Logic and Application (FPLA '97)*, pp. 213–222, London, UK, 1997.

[16] V. Betz and J. Rose, "Cluster-based logic blocks for FPGAs: area-efficiency vs. input sharing and size," in *Proceedings of the Custom Integrated Circuits Conference*, pp. 551–554, Los Alamitos, Calif, USA, 1997.

[17] A. Pandit and A. Akoglu, "Wirelength prediction for FPGAS," in *Proceedings of the International Conference on Field Programmable Logic and Applications (FPL '07)*, pp. 749–752, Amsterdam, The Netherlands, August 2007.

[18] J. Lamoureux and S. J. E. Wilton, "On the interaction between power-aware CAD Algorithms for FPGAs," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD '03)*, pp. 701–708, San Jose, Calif, USA, November 2003.