Logic Block Clustering of Large Designs for Channel-Width Constrained FPGAs[*]

Marvin Tom
marvint @ ece.ubc.ca

Guy Lemieux
lemieux @ ece.ubc.ca

Dept of ECE, University of British Columbia, Vancouver, BC, Canada

ABSTRACT

In this paper we present a system level technique for mapping large, multiple-IP-block designs to channel-width constrained FPGAs. Most FPGA clustering tools [2, 3, 11] aim to reduce the amount of inter-cluster connections, hence reducing channel width needs. However, if this exceeds the FPGA's channel width (a *hard constraint*), then the circuit still cannot be routed. Previous work [11, 12] depopulates logic clusters (CLBs) to reduce channel width. By depopulating non-uniformly, *i.e.* depopulate more in hard-to-route regions, we show a graceful trade-off between channel width and CLB count. This makes it possible to target specific channel-width constraints during clustering with minimal CLB inflation. Results show channel width decreases of up to 20% with a 5% increase in area. Further decreases of nearly 50% are possible at 3.3 times the original area. Despite the area increase, this technique creates routable solutions from otherwise-unroutable circuits.

Categories and Subject Descriptors: B.7.2 [Integrated Circuits]: Design Aids

General Terms: Algorithms, Design, Experimentation

Keywords: Field-Programmable Gate Arrays (FPGA), Clustering, Packing, Channel Width Constraints

1. INTRODUCTION

A commercial FPGA family consists of a number of devices, each with a different logic capacity. Figure 1 illustrates the logic resources: CLBs and BLEs. Logic capacity is measured by the number of BLEs, or basic logic elements. Alternatively, it can be measured by the number of CLBs, or configurable logic blocks, which are simply fixed-size *clusters* of BLEs. Device logic capacity is determined by the logical dimensions of the CLB array.

Interconnect capacity is determined by the *channel width*, *i.e.* the number of wiring tracks in each channel. These channels appear on all four sides of each CLB. Since the same layout tile is used throughout the FPGA family, the channel width is fixed. This creates a strict *channel-width constraint*: if a circuit needs more routing tracks, it cannot be implemented regardless of available CLBs.

[*]Code available at http://www.ece.ubc.ca/∼lemieux/downloads

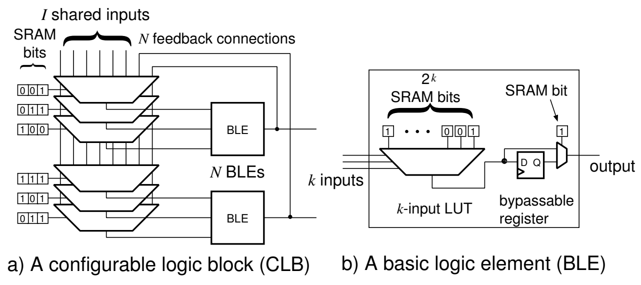

Figure 1: A CLB and a BLE

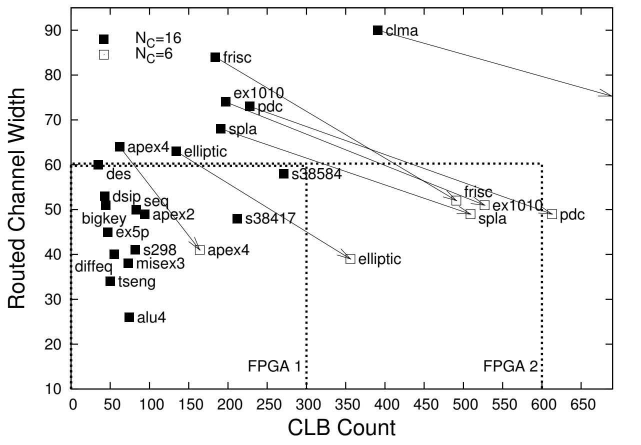

Figure 2: Channel width / CLB count tradeoff ($k = 6$, $N_A = 16$)

CAD tools should be able to target the channel-width constraint of an FPGA family, possibly at the expense of using additional CLBs. Today, it is quite common to specify a timing constraint to FPGA CAD tools, but it is quite unusual to specify a channel width constraint (even implicitly). We propose a systematic method for addressing these channel-width constraints at the clustering stage.

First, we note it is possible to reduce the channel width needs of a circuit through clustering. Traditional clustering algorithms, such as T-VPack [2], fully pack the clusters to minimize the total number of CLBs needed. However, DeHon [6] and Tessier [12] have shown that the channel width needs of a circuit can be decreased by packing fewer BLEs into each CLB. The resulting "under-utilization" of CLBs is known as *depopulation*.

To see how depopulation works, consider the two large dashed boxes in Figure 2 representing the logic and routing capacities of two FPGA devices. These FPGAs contain 16 BLEs per CLB and

60 wiring tracks per routing channel. The channel width needs of 20 MCNC benchmark circuits [4] after clustering (T-VPack) and routing (VPR) are shown. Notice that circuits with similar CLB counts can require vastly different channel widths (varying from 25 to 65). Similar results for industrial benchmarks are shown in [9].

FPGA 1 contains 300 CLBs and can implement all circuits inside its box. In comparison, FPGA 2 has 600 CLBs and the same channel-width constraint of 60 because it is based on the same layout tile. Even though it is larger, FPGA 2 is incapable of realizing any circuits that require a channel width greater than 60, *e.g.* **apex4** or **elliptic**. After depopulation (limiting to 6 BLEs per CLB), **apex4**'s channel width shrinks from 62 to 41 tracks. Although the CLB count increases, it still fits into FPGA 1. More importantly, **apex4** now has a viable, routed solution. Similarly, some circuits like **elliptic** can now fit FPGA 2.

The problem with depopulation is that it quickly leads to an inflated CLB count. In the example, circuits **pdc** and **clma** are too large for FPGA 2.[1] They must be depopulated less to meet the CLB constraint as well. What is needed is a way to depopulate only the routing-congested regions of a circuit so CLB count is inflated as little as possible. We believe such an approach is important for fitting large System-on-Chip designs onto modern FPGAs.

The primary application of our depopulation technique is to reduce the channel width requirements of a circuit so that it can be mapped to a channel-width constrained FPGA. Rather than depopulate the entire circuit, which would inflate area rather quickly, we suggest depopulating smaller regions (possibly entire IP blocks) that are interconnect-intensive.

The secondary application of our depopulation technique is for FPGA architects designing lower-cost FPGAs. When designing a new FPGA architecture, it is important to determine a fixed channel width for the entire family. Our approach allows adoption of a slightly smaller channel width (perhaps 10%) without a net area increase. This is done by mildly depopulating the hard-to-route benchmarks just enough to match the planned CLB array sizes prior to determining the family's final channel width.

The remainder of paper is organized as follows. Section 2 describes previous work in cluster depopulation. Our benchmark creation strategy is discussed in Section 3. Section 4 discusses the new clustering method. Results are presented in Section 5, followed by conclusions in Section 6.

The following terms will be used in this paper:

k	LUT size (number of inputs)
N_C	cluster size of *circuit* (BLE-limit, max. # *used* BLEs)
N_A	cluster size of *architecture* (# of BLEs per CLB)
I	Number of inputs to a cluster

2. PREVIOUS WORK

One of the earliest attempts to balance logic and routing elements to decrease area was performed by DeHon [6]. However, this analysis was performed for an FPGA with a binary tree interconnect structure. In this work, we adopt a mesh interconnect which is more representative of commercial FPGAs.

Tessier [12] showed that depopulation of clusters can result in reduced channel widths. The algorithm presented in [12] depopulates each cluster equally so there is a uniform distribution of empty BLEs across the chip (*e.g.*, N_C is fixed). Although this reduces channel width, it also depopulates regions of the circuit that are not heavily congested. This leads to unnecessary CLB inflation in these regions. We suggest using a different N_C value for each partition of the circuit. This N_C value may vary between partitions in the circuit such that routing-congested areas are depopulated more.

Singh [11] presented a clustering algorithm (iRAC) which is very effective at reducing channel width. iRAC reduces channel width by identifying low fan-out nets and completely absorbing them into a cluster. This reduces the total number of external nets, hence reducing the routed channel width. iRAC also limits the number of inputs to each CLB by using the Rent parameter of the underlying architecture, resulting in solutions that have some depopulation. We have found limiting inputs is ineffective, so we limit BLE usage. Our technique also differs from [11] by targeting specific channel-width constraints.

To illustrate that our technique can be used with multiple clustering algorithms, we use both T-VPack and iRAC. We have implemented a replica of iRAC using the seed selection and attraction functions in [11], but we did not implement the Rent-based input-limiting function. Despite this, our iRAC replica achieves within 2% of [11] in the number of external nets. All of our experiments were carried out using both T-VPack and the iRAC replica.

3. BENCHMARK CREATION

Current FPGAs can implement multi-million gate System-on-Chip (SoC) designs. Unfortunately, large SoC designs are not available for academic research. Available benchmarks are small and do not have obvious IP block partitions. However, FPGA researchers need large circuits to investigate new architectures and CAD algorithms. Hence, we created our own synthetic benchmark circuits to cope with this problem.

SoC designs consist of multiple IP blocks integrated together. The IP blocks can be widely varied in their function and purpose, and are often worked on by different designers. During development, each IP block might be individually placed and routed on an FPGA several times. As well, these different blocks may have different interconnect demands, just like those shown in Figure 2.

To mimic a large SoC, we create *meta-circuit* benchmarks by treating the largest 20 MCNC circuits as individual IP blocks of a common SoC. Each MCNC circuit is a unique, self-contained function with an appropriate I/O count, just like an IP block. Some of these MCNC circuits (*e.g.* **bigkey**) have many inputs and outputs, making them similar to "glue logic" that connects other IP blocks together. To avoid creating combinational loops, a flip-flop is added to the primary outputs of each MCNC circuit. Then, we stitched these together into the following three different meta-circuits:

- **Independent.** Each primary input and primary output of each IP block remains a primary input and primary output of the meta-circuit. There is no interaction between IP blocks.

- **Pipeline**. The IP blocks are placed in a random, sequential order, each representing stages in a pipeline. Additional (leftover) inputs/outputs between pipeline stages become primary inputs/outputs of the meta-circuit.

- **Clique**. The outputs of each individual IP block are uniformly distributed to the inputs of all other circuits in the meta-circuit. The connections are made to encourage as much inter-block communication as possible.

When stitching, precise output-to-input connections are assigned once randomly and held constant for all benchmarks created. Only connections with fan-out of 1 are formed. We connect only to the IO boundaries and not any internal nodes of the IP blocks.

[1] In Figure 2, some circuits are depopulated more than necessary.

Alternatively, we could have used synthetic-circuit generating techniques [7, 8, 13]. These techniques are good for cloning existing circuits: they typically work by top-down partitioning or bottom-up clustering of modules and adding nets between the modules while enforcing stochastic interconnect parameters. Unfortunately, we do not have any initial SoC designs to clone. Also, we believe "scaling up" a single IP block to mimic a large SoC design is not realistic: it ignores the natural efficiency of design partitioning and the well-defined IO boundaries of each IP block.

We could improve our random-assignment stitching by applying the published synthetic techniques at the top level. This was not done due to time constraints. Our primary concern was to create a large circuit with varying interconnect usage among the IP blocks to determine whether our depopulation approach is useful.

Experiments in this paper were carried out using all 3 meta-circuits. We believe that large SoC designs would contain a mixture of these 3 styles, but the single most-representative style would have more inter-block connections like the **Clique** circuit.

4. CLUSTERING METHOD

This section describes the new clustering approach. We enforce BLE-limits during clustering, profile each IP block's channel width needs for different depopulation levels, and choose the one with the fewest CLBs. Then we predict the overall area based on the assumption that the channel width of the meta-circuit will match the most-congested IP block. This prediction shows a large, flat area region where CLB count can be safely traded off for area.

4.1 Depopulation Strategies

We evaluated the effectiveness of two different CLB depopulation methods. The first method, similar to [12], is to strictly limit the number of BLEs that can be packed into a CLB (BLE-limit). The second method, similar to [11], is to strictly limit the number of inputs into a CLB (input-limit). Figure 3 shows the routed channel widths for circuit **clma** after implementing the two limits in both clustering algorithms. Other circuits produce similar results.

Figure 3 shows that the BLE-limit method exhibits a monotonically increasing relationship between the BLE-limit size and the routed channel width. Hence, BLE-limit can be effectively used to decrease routed channel widths. Surprisingly, the input-limit approach did not exhibit this same predictable relationship. This contradicts traditional thinking that reducing inputs is an effective way to reduce channel width. Hence, we adopted the BLE-limiting technique.

4.2 Channel-width Profile of IP Blocks

For FPGA designs that contain multiple IP blocks, we hypothesized that the channel width needed to route will be similar to the IP block with the highest channel width needs. That is, the other IP blocks do not temper the channel width needs of the hard-to-route IP block. Although this is just a first-order approximation that ignores the effects of inter-block communication, we have found it to be a good estimate of the final routed channel width. Hence, we first develop a channel-width profile of each IP block. Then, we select the depopulation level needed by each IP block to meet the overall channel-width constraint.

The 20 MCNC benchmark circuits were synthesized and technology mapped using FlowMap [5] and FlowPack. A LUT size of $k = 6$ was used for improved delay [10]. The circuits were clustered using both T-VPack [2] and our iRAC replica and then placed and routed using VPR v4.30 using length 4 wires [2]. We placed and routed the circuits for cluster sizes $N_C = 2$ to 16. The number of inputs used is $I = k(N_C + 1)/2$ [1].

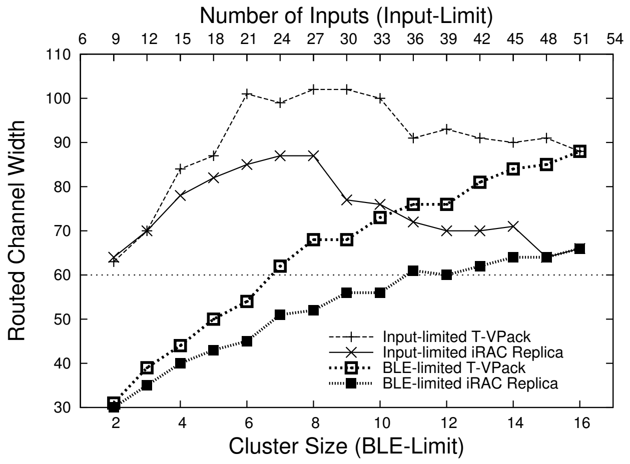

Figure 3: Input- and BLE-limits during clustering

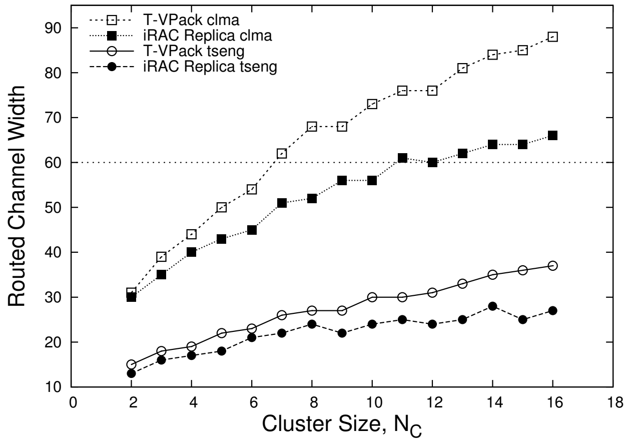

Figure 4: Circuit, algorithm, and N_C impact

Figure 4 shows the channel width needs of two circuits as the cluster size N_C is increased. If a channel-width constraint of 60 is imposed with T-VPack, a cluster size $N_C \leq 6$ is required to route **clma**. We say 6 is the *maximal cluster size* for **clma** at the given channel-width constraint. In contrast, the maximal cluster size for **tseng** is 16 for the same constraint.

Using an architecture with a CLB size of $N_A = 16$, we set 11 different channel-width constraints and determined these maximal cluster sizes for each IP block using both T-VPack and our iRAC replica. T-VPack results for some circuits are shown in Table 1. Channel-width constraints below 45 were not possible because some circuits could not be depopulated enough to route with such a small channel width. Channel-width constraints greater than 95 were not interesting because all CLBs were fully populated.

4.3 Non-uniform Clustering of Meta-circuit

The meta-circuit is formed by stitching together individual clustering solutions of each IP block. This allows us to depopulate only the routing-intensive ones. Although this stitching also precludes some inter-block optimizations, this should have little impact if the IP blocks are sufficiently large enough. Also, clustering individually preserves each IP block in a form that more closely resembles how each was developed and tested by separate designers prior to integration.

Circuit	Channel-width Constraint 45	50	55	60	65	70	75	80	85	90	95
clma	3	5	5	6	8	10	11	12	14	15	16
dsip	6	13	16	16	16	16	16	16	16	16	16
frisc	4	5	7	7	9	10	13	15	16	16	16
spla	5	6	8	11	13	16	16	16	16	16	16
tseng	16	16	16	16	16	16	16	16	16	16	16

Table 1: Maximal N_C cluster sizes from T-VPack

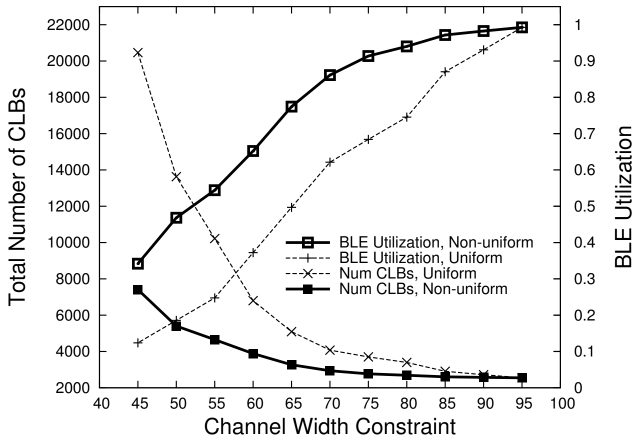

Figure 5: CLB count and BLE utilization with T-VPack

When clustering the IP blocks, we have two choices for the cluster size N_C with a given channel-width constraint:

- **Uniform (Minimum) Cluster Size**. Depopulate all of the IP blocks to the same value of N_C, the minimum of the maximal cluster sizes for all IP blocks. This is similar to [12] which also uses uniform depopulation of clusters.
- **Non-uniform (Maximal) Cluster Size**. Depopulate the IP blocks by different amounts, using the maximal cluster size for each one.

For each of the 11 constraints in Table 1, we generated a **Uniform** meta-circuit and **Non-uniform** meta-circuit. This was also repeated for the iRAC clustering solutions.

As discussed earlier, the **Uniform** meta-circuit will contain more CLBs than necessary and results in lower BLE utilization. Figure 5 shows the total CLBs and BLE utilization obtained from the meta-circuits produced from T-VPack clusterings. Although not shown, similar results were obtained using iRAC. It is clear from Figure 5 that **Non-uniform** clustering of the IP blocks significantly improves both BLE utilization and CLB count. Hence, we adopt this approach for the remainder of the paper.

4.4 Area Prediction

The change in CLB count and the channel-width constraint can be used to roughly predict the overall impact on area. Assuming that routing consumes approximately 70% of the total FPGA area, and logic consumes the remaining 30%, the predicted area can be expressed as a factor of the original area:

$$\text{Predicted Area Factor} = \left(\frac{W_{\text{depop}}}{W_{\text{full}}}\text{RF} + \text{LF}\right)\frac{\text{CLB}_{\text{depop}}}{\text{CLB}_{\text{full}}}$$

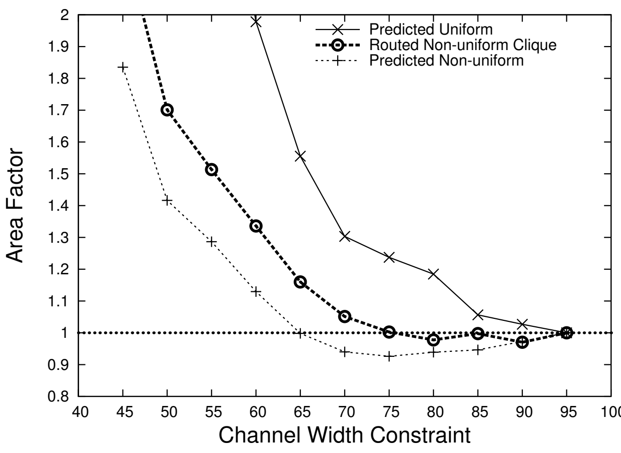

Figure 6: Predicted area with T-VPack

where:		
	W_{depop}	Channel width constraint (depopulated)
	W_{full}	Original channel width (fully populated)
	RF	Routing fraction (=0.7)
	LF	Logic fraction (=0.3)
	$\text{CLB}_{\text{depop}}$	Total number of CLBs (depopulated)
	CLB_{full}	Original number of CLBs (fully populated)

Figure 6 shows the predicted area factor for the T-VPack **Non-uniform** and **Uniform** meta-circuits. For example, it is predicted that a channel-width constraint of 75 for **Non-uniform** lowers area by 5% compared to the original fully-populated version with a channel width of 95. From this figure, we see how **Uniform** clustering is ineffective: it quickly leads to an area increase. In contrast, the **Non-uniform** approach area is relatively flat from 65–95. This suggests we can aggressively apply channel-width constraints with little impact on overall area.

Also shown in Figure 6 is the actual area overhead obtained using VPR after routing the **Non-uniform Clique** meta-circuit. The actual area is higher than the predicted area, but it still shows a substantially flat area response for channel widths of 75–95.

5. RESULTS

The circuit stitching program was used to create meta-circuits for each of the channel-width constraints listed in Table 1. The target architecture has a LUT size of $k = 6$, cluster size of $N_A = 16$, and $I = 51$. We also carried out the same experiment with $N_A = 10$, $I = 33$ and observed similar results. These N_A values were chosen to match cluster sizes of Altera's Stratix II [10] and Cyclone II.

In total, 66 large netlists were created and placed using VPR (11 channel width constraints, 3 meta-circuits, using both T-VPack and the iRAC replica). We expect that large SoC designs will be floorplanned prior to the final placement process, but VPR does not support floorplanning. Instead, it starts with a random placement of all CLBs and uses simulated-annealing to find a minimum-cost placement. Interestingly, VPR was able to generate solutions that appear to be floorplanned. This reduced our need to impose an artificial floorplan on the design *a priori*. Figure 7 shows a VPR screen shot after placement. We have edited this figure to highlight the IP block locations and to show the amount of depopulation.

Next, the placed netlists were routed using VPR. While routing, the channel width was continuously reduced until the circuit became unroutable. This produced the final maximum routed channel

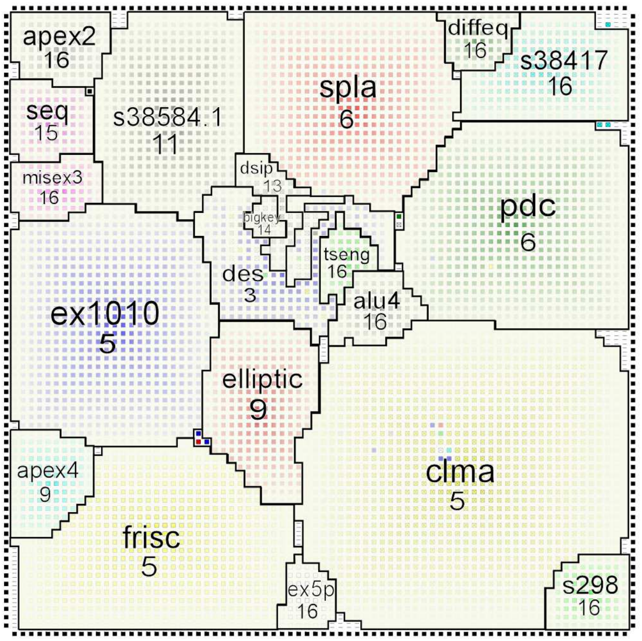

Figure 7: VPR placement of Non-uniform Clique with T-VPack

width.[2] The maximum channel width results for T-VPack and the iRAC Replica versions of **Clique** are shown in Figure 8.

The results show that the maximum routed channel width was *slightly higher* than what was imposed by the channel-width constraint. This is expected, as the channel-width constraint ignores the effects of inter-block communication and I/O padframe access. The most important result is that the routed channel width decreases as the channel-width constraint is reduced.

Figure 8 shows that channel width decreases of almost 50% are possible with **Clique**. Although this decrease comes with a large overall area penalty (2.3x from Figure 6), it may be the **only viable solution** to users of channel-width constrained FPGAs.

The same experiments were repeated for **Independent** and **Pipeline**. Figure 9 shows the maximum routed channel width of **Pipeline** produced from the iRAC replica. Here, we note that the maximum channel width *gets worse* as channel-width constraints are imposed. Similar results were obtained with T-VPack, and also with **Independent**. This is not a failing of our clustering approach, but a limitation of the interactions between the benchmark, VPR placement, and the underlying FPGA architecture. Specifically, we found that the IO-intensive IP blocks were strongly attracted to the IO padframe during placement and stretched into highly rectangular shapes. This caused severe localized congestion in the routing channels nearest to the padframe. This can probably be solved, so we calculated the *average* channel width of all routing channels and show this in Figure 9. The average channel width still tracks the channel-width constraint, suggesting that our approach is viable.

Table 2 shows a summary of the best channel width decreases that were obtained for each benchmark circuit. Small decreases of 0–20% are possible with only 5% increase in area. Large decreases of nearly 50% are possible with up to 3.3 times overall area. Again, although this is a high area cost, it may be the **only viable solution** in a real FPGA device where hard channel-width constraints are imposed.

[2]It is a *maximum* because the routing solution uses at most this many routing tracks in each of the channels.

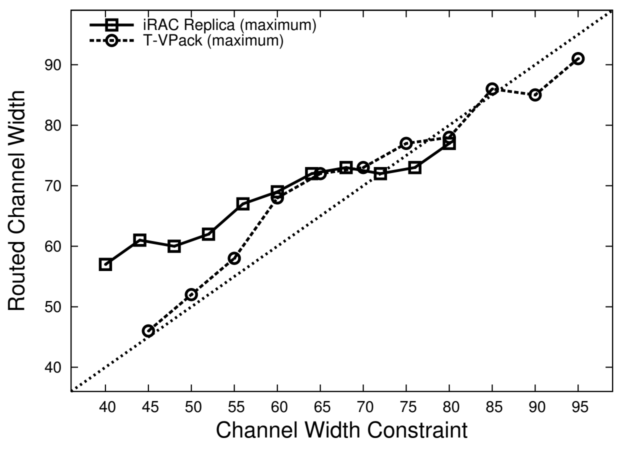

Figure 8: Channel width of Non-uniform Clique

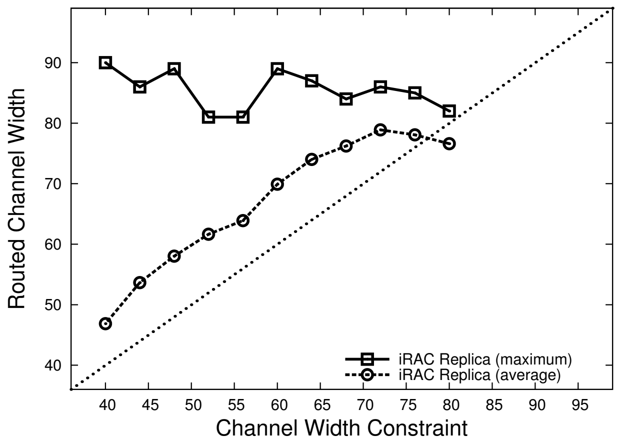

Figure 9: Channel width of Non-uniform Pipeline

In some cases, only small decreases in maximum channel width were achievable. As explained earlier, this is because some IP blocks introduce heavy congestion at the periphery due to high IO-pad needs. Instead, Table 2 also shows the average channel width required. The average channel width can be reduced by up to 50% using our depopulation technique. For this average width to become a real savings, the placement tool and architecture must be tuned to avoid the hotspots that are imposed by our benchmark.

Figure 10 shows the critical-path delay results. Critical-path delay should gradually increase as more depopulation is applied. In general, delay does seem to follow this trend, but it does tend to jump around. This "delay noise" appears to result from instability in the placement. As depopulation is applied, the VPR placement engine keeps IP blocks together, but sometimes their location in the floorplan is shifted significantly relative to other IP blocks. This caused the critical path to sometimes relocate from within an IP block (which gradually degrades as depopulation is applied) to connections between IP blocks (which introduces large delay jumps). Imposing a pre-defined floorplan may help reduce this "noise" in large designs.

Figure 10 also shows the average wirelength per net. Average wirelength per net increased as more depopulation is applied. This is expected because an increase in CLB count must also increase average distance traveled. Also, depopulating will cause connec-

		Channel Width Decreases			
	Clustering	(<1.05x area)		(2.3x–3.3x area)	
Circuit	Tool	MaxW	AvgW	MaxW	AvgW
Clique	T-VPack	20%	14%	49%	47%
	iRAC Rep.	5%	6%	26%	38%
Indep-endent	T-VPack	10%	9%	18%	44%
	iRAC Rep.	2%	3%	14%	46%
Pipeline	T-VPack	11%	9%	24%	44%
	iRAC Rep.	0%	1%	1%	39%

Table 2: Best reductions in channel width

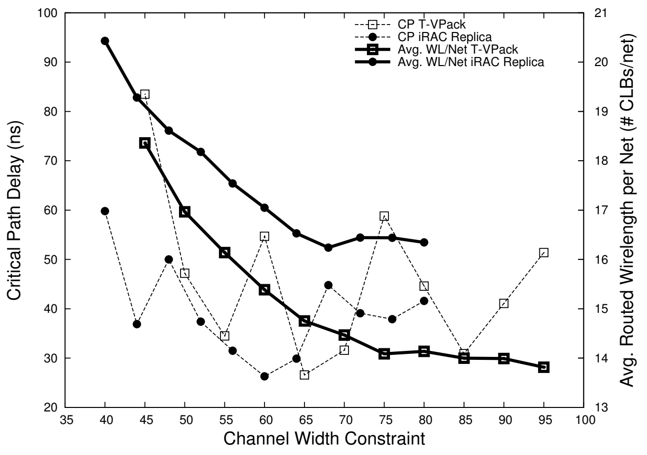

Figure 10: Critical-path delay for Clique

tions that were previously internal to a CLB (hence, ignored) to become external nets with a measurable distance. This slightly tempers the increase in average wirelength.

Notice that iRAC replica produces a higher *average* wirelength than T-VPack. However, the *total* wirelength was lower and the critical-path delay results were similar.

6. CONCLUSIONS

In this paper we have proposed a system level technique for mapping large system-on-chip (SoC) designs to channel-width constrained FPGAs. In particular, the method helps fit hard-to-route circuits into FPGAs that have narrow channel widths at the expense of using more CLBs. Since larger devices with more CLBs are usually available, this is a practical trade-off.

We have observed that depopulating CLBs (*i.e.*, not filling them to capacity) is a very effective way to reduce channel width needs of a circuit. Limiting the number of BLEs in a CLB that are used is much more effective than limiting the number of inputs to a CLB.

It is important to apply **Non-uniform** depopulation when clustering. Otherwise, area increases very rapidly and limits the usefulness of the approach. This was demonstrated using a simple area model.

We have demonstrated that it is sufficient to selectively depopulate parts of a large circuit that would otherwise have routing congestion. We depopulate the most routing-intensive IP blocks until the routing demands of those blocks are comparable to the demands of the other blocks. This way, channel width can be reduced by 0–20% with less than 5% increase in area. By continuing to depopulate more IP blocks, we can continue to decrease channel width by about 50% at the expense of more CLBs. Overall, this uses up to 3.3 times the original area, but it may be the **only viable solution** in a real FPGA device where hard channel-width constraints are imposed. By purchasing an FPGA device with higher logic capacity, designs which are otherwise unroutable can be made routable.

Future work is being expanded on three fronts. First, we are applying this technique to a commercial Bluetooth SoC. Second, we are attempting to automate this approach into the CAD flow. Third, we are attempting to address the VPR placement problem that causes the channels near the padframe to become overly congested.

7. ACKNOWLEDGMENTS

Thanks to Edmund Lee for his assistance. We are grateful for the use of WestGrid[3] computing resources.

8. REFERENCES

[1] E. Ahmed and J. Rose. The effect of LUT and cluster size on deep-submicron FPGA performance and density. In *Int'l Symp. on FPGAs*, pages 3–12, 2000.

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

[3] E. Bozorgzadeh, S. Ogrenci-Memik, et al. Routability-driven packing: Metrics and algorithms for cluster-based FPGAs. *Journal of Circuits, Systems, and Computers*, 13(1):77–100, February 2004.

[4] Collaborative Benchmarking Laboratory. *LGSynth93 suite*. North Carolina State University.

[5] J. Cong and Y. Ding. FlowMap: An optimal technology mapping algorithm for delay optimization in lookup-table based FPGA designs. *IEEE Trans. on Computer-Aided Design*, pages 1–12, January 1994.

[6] A. DeHon. Balancing interconnect and computation in a reconfigurable computing array. In *Int'l Symp. on FPGAs*, pages 69–78, 1999.

[7] M. Hutton, J. Rose, and D. Corneil. Automatic generation of synthetic sequential benchmark circuits. *IEEE Trans. on Computer-Aided Design*, 21(8), August 2002.

[8] P. Kundarewich and J. Rose. Synthetic circuit generation using clustering and iteration. *IEEE Trans. on Computer-Aided Design*, 23(6), June 2004.

[9] P. Leventis, M. Chan, et al. Cyclone: A low-cost, high-performance FPGA. In *IEEE Custom Integrated Circuits Conference*, San Jose, CA, September 2003.

[10] D. Lewis, E. Ahmed, et al. The Stratix II logic and routing architecture. In *Int'l Symp. on FPGAs*, Monterey, CA, February 2005.

[11] A. Singh and M. Marek-Sadowska. Efficient circuit clustering for area and power reduction in FPGAs. In *Int'l Symp. on FPGAs*, pages 59–66, 2002.

[12] R. Tessier and H. Giza. Balancing logic utilization and area efficiency in FPGAs. In *Int'l Workshop on Field Programmable Logic and Applications*, 2000.

[13] P. Verplaetse, D. Stroobandt, and J. van Campenhout. Synthetic benchmark circuits for timing-driven physical design applications. In *Int'l Conference on VLSI*, pages 31–37, Las Vegas, NV, June 2002.

[3]WestGrid is funded in part by the Canada Foundation for Innovation, Alberta Innovation and Science, BC Advanced Education, and the participating research institutions. WestGrid equipment is provided by IBM, Hewlett Packard and SGI.