Congestion-Driven Regional Re-clustering for Low-Cost FPGAs

Darius Chiu, Guy G.F. Lemieux, Steve Wilton

Electrical and Computer Engineering, University of British Columbia
British Columbia, Canada
dariusc@ece.ubc.ca
lemieux@ece.ubc.ca
stevew@ece.ubc.ca

***Abstract*—FPGA device area is dominated by a limited amount of interconnect. CAD tools must meet a hard channel-width constraint for a circuit to be successfully mapped to a device. Previous work has shown that if a design cannot be mapped to a device due to insufficient interconnect availability, it is possible to identify regions of high interconnect demand and spread out the logic in this area into surrounding regions. This is done by re-packing logic in the affected regions into an increased number of CLBs. This increases the effective amount of interconnect in these high-demand areas. This methodology has been shown to significantly reduce channel width, at the expense of CLB count and runtime.**

In this paper, we extend this previous algorithm in two ways: we present novel region selection techniques to optimize the selection of which regions should be depopulated, and we introduce a local channel-width demand model which can be used to more accurately determine the amount of white space insertion at each iteration. Together, these techniques lead to significant run-time improvements and reduce the area of the resulting FPGA implementations. We were able to improve runtime by a factor of up to 5.5 times while reducing area by up to 20% when compared to previous methods.

I. Introduction

FPGA devices have limited routing and logic capacity. For a similar number of logic elements (LEs), it is common for manufacturers to offer routing-rich devices at a high cost, or much less expensive devices with less routing. It is then much more economical to try to meet the channel-width constraints of the latter devices. In the situation where routing resources are the limiting factor, CAD tools must be able meet a hard channel-width constraint. Since local interconnect usage varies spatially with placement, spreading LEs among an increased number of configurable logic blocks (CLBs) will allow access to more aggregate routing resources, thus spreading areas of peak routing demand. Effectively, this creates a tradeoff between peak channel-width demand and logic usage.

While effective, this tradeoff between logic and routing must be performed for congested areas. That is, it is not economical to globally fill each CLB with just a single LE to reduce interconnect demand as much as possible. Instead, tools addressing this problem must take into consideration that routing resource demands will vary between regions of the placement. CLBs in different regions must be non-uniformly clustered to meet local interconnect demands without excess CLB use. This is especially relevant in large System-on-Chip designs, where several tightly connected IP blocks are connected together to form a larger circuit. A select few IP blocks may have internal interconnect requirements that exceed channel-width constraints, causing the entire circuit to be unroutable. Yet, reducing congestion specifically for these few IP blocks will allow the entire circuit to be routable without unnecessarily inflating area by reclustering other regions.

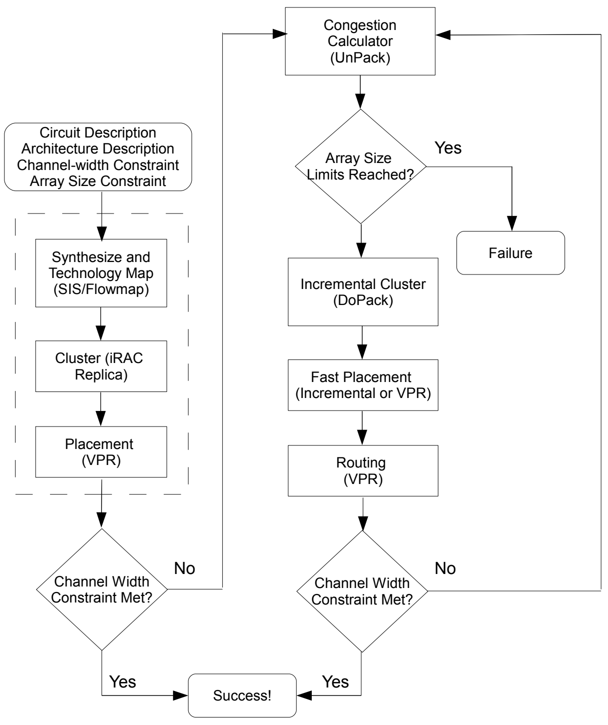
Un/DoPack [1] presented a fully automated CAD flow which algorithmically lowers the minimum routable channel-width (MRCW) of a circuit by iteratively inserting whitespace, in the form of empty LEs, into the design where local interconnect requirements exceed channel-width constraints. Congested regions are identified, CLBs in these regions are fully unpacked into constituent LEs, and repacked into clusters with smaller cluster-sizes. The process of re-clustering LEs into smaller clusters is called depopulation. This framework is the basis for the work presented in this paper.

While Un/DoPack demonstrated the effectiveness of depopulating local regions to meet hard channel-width constraints, this work introduces several congestion-driven methods which include local congestion information to improve the depopulation decision-making process. In particular, we improve the basic Un/DoPack framework by better choosing which areas of the design to depopulate. We use a local channel-width demand model to calculate the amount of whitespace insertion required and also integrate a congestion-driven placement technique. Our goal is to reduce runtime and area of Un/DoPack.

The remainder of this paper is organized as follows. Section 2 discusses background and related work. Section 3 outlines the techniques used in this paper to improve to the Un/DoPack flow. Section 4 presents the methodology used in this work. Section 5 discusses our results and section 6 presents the conclusion and future work.

II. Background and Related Work

We assume an island-style architecture containing an array of logic blocks separated by horizontal and vertical rows of routing tracks. The logic capacity of this architecture is defined as the number of configurable logic blocks (CLBs), each containing a fixed number of logic elements (LEs). Each LE



Congestion Calculator (UnPack)
Circuit Description Architecture Description Channel-width Constraint Array Size Constraint
Array Size Limits Reached?
Yes
Failure
Synthesize and Technology Map (SIS/Flowmap)
Incremental Cluster (DoPack)
Cluster (iRAC Replica)
Fast Placement (Incremental or VPR)
Placement (VPR)
Routing (VPR)
Channel Width Constraint Met?
No
Channel Width Constraint Met?
No
Yes
Success!
Yes

Fig. 1. Complete Un/DoPack CAD Flow [1]

consists of a lookup table and flip-flop. The *minimum routable channel-width* (MRCW) is defined as the minimum number of routing tracks needed to successfully map a circuit to the target architecture. The maximum MRCW is defined as the minimum number of routing tracks needed to successfully map a circuit to the target architecture without any depopulation. The interconnect demand near a CLB is captured with a *congestion label*. The congestion label of a CLB is the maximum number of signals routed from both the X or Y directions.

While clustering tools exist which maximize logic utilization by fully packing CLBs, various authors [2][3] have shown that the best performance may result from a balance between logic utilization and interconnect demand. Work by [1][4][5] showed that overall area could be reduced by packing CLBs to less than 100% capacity. Since FPGA area is dominated by interconnect, reducing the overall interconnect requirements by balancing local routing demand with logic utilization can produce a net decrease in FPGA area.

In [3], a clustering tool is presented which performs depopulation uniformly across the design. However, this also leads to depopulation of uncongested regions. Un/DoPack [1] performs depopulation for local regions across the design. Un/DoPack reduces interconnect in localized regions of the FPGA by iteratively reclustering regional LEs into an increased number of CLBs. Whitespace insertion limited to a local region is especially important in the case of circuits with large interconnect variation, such as system-on-chip circuits. These circuits may consist of many different subcircuits, each having significantly different interconnect demands.

While Un/DoPack was shown to be very effective in reducing the channel-width, runtime and area expansion can be reduced through the use of better congestion-driven region selection and cluster-size calculation techniques. The basic Un/DoPack flow is described in detail in the next section.

A. Un/DoPack CAD Flow

The Un/DoPack Flow, shown in Figure 1, iteratively reduces the MRCW of a circuit by spreading local areas of high congestion until the MRCW of the circuit meets a user specified channel-width constraint. The user provides the following inputs to Un/DoPack: a circuit description, architecture description, target channel-width constraint, and an array size constraint.

Initially, a traditional CAD flow using SIS/Flowmap [6] and VPR [7] is run. Any packing and placement algorithm can be used; we use our iRAC replica and VPR. If it meets the specified target channel-width constraint on the first pass, it is done. This is shown inside the dashed outline in Figure 1. If the channel-width constraint cannot be met, the iterative portion of the Un/DoPack flow is invoked, which consists of the steps described below.

The first step of the iterative portion, the UnPack step, determines which regions to depopulate. A region should be depopulated if the local interconnect demands of the region exceed the specified channel-width constraints. Previous work presented two depopulation schemes: a single and multiple region depopulation scheme. These schemes select regions of a fixed radius, and calculate a new cluster-size. The new cluster-size is calculated such that enough whitespace is introduced to expand the number of CLBs in the region by a pre-determined amount. Regions are selected based on the highest congestion label in the congestion map first, closest to the center of the array. The single region version of Un/DoPack adds an amount of CLBs equal to the number of CLBs in a row and column of the array. The multiregion version of Un/DoPack inserted whitespace proportional to the peak congestion and an empirically determined scaling factor.

The second step repacks the LEs in the selected regions. The clusters are packed less than 100% full using new cluster-sizes determined by the UnPack step. LEs from each region are reclustered with other LEs from the same region. Any clustering algorithm can be used; for this work we will be using a replica of iRAC [4] as our clustering algorithm.

The final step is to perform placement and routing. Un/DoPack uses the incremental placer, RePlace [8], which preserves the placement stability in each iteration. Because Un/DoPack is creating additional CLBs at each iteration, these will need to be placed somewhere in the array. RePlace creates an initial placement where CLBs are placed in approximately the same regions as in the previous placement. A low temperature anneal is then performed to optimize the placement. Since Un/DoPack is modifying only small localized regions of the FPGA, the incremental placement has the combined effect of significantly reducing runtime, when compared to a full VPR placement. Incremental placement also preserves relative placement for CLBs in uncongested regions. After the routing step, if the circuit remains unroutable at the specified channel-width constraint, Un/DoPack will iteratively depopulate the remaining congested areas using the UnPack and DoPack steps. The complete Un/DoPack CAD flow is illustrated in Figure 1.

B. Baseline Un/DoPack

The baseline version of Un/DoPack, depopulates a large, single region of the device. In each iteration, the number of CLBs created is equal to the number of CLBs in a row and column of the array.

C. Fine-Grained Un/DoPack

As one of the main observations in [9], authors noted that adding very small amounts of whitespace at each iteration produced superior area results in spite of a large increase in runtime. They referred to this as the Fine-Grained approach. At each iteration, a single region is selected. The number of CLBs in the region is determined using Equation 1.

$$new\_region\_CLBs = \frac{num\_region\_LEs}{num\_CLBs\_in\_region + \sqrt{num\_CLBs\_in\_region)} * 2 + 1} \quad (1)$$

D. Multiregion Un/DoPack

The multiple region approach presented in [1], referred to as Multiregion Un/DoPack, improved runtime by depopulating multiple regions simultaneously. In this approach, there is no restriction on how many CLBs are created in each iteration. Each region in the Multiregion depopulation scheme grows proportionally to the peak congestion in each region and an empirically determined scaling factor. This method produces much better runtime results than Fine-Grained Un/DoPack, but area inflation is increased.

III. Multiple Region Depopulation with congestion-driven Metrics

This section describes the improvements to Un/DoPack. We introduce congestion-driven techniques that utilize local congestion metrics, to reduce runtime and area inflation.

A. Region Selection

We first introduce techniques to optimize the region selection process. Whereas [9] depopulates regions centered on CLBs with the largest congestion labels which are closest to the center of the array, we introduce the following procedure which attempts to select the most congested region to depopulate.

The first congestion region is selected by finding the CLB with the highest congestion label, closest to the center of the array. This forms an initial x,y center location for a rectangular windowed region of size $W \times H$ which will be marked for depopulation. In this work, these window sizes are chosen to match those of previous work [1] [9]. Next, the window center is adjusted slightly (up to $\pm\frac{W}{2}$ or $\pm\frac{H}{2}$) by using force directed shifting. The congestion label for each CLB in the window and the position relative to the region center is used to create vectors which lead away from the center of the region. The sum of these vectors produces a net direction in which to move the window. A search in this direction is then used to determine the window location that encompasses the largest average congestion. The furthest that the window can shift is a distance of up to $\pm\frac{W}{2}$ or $\pm\frac{H}{2}$ away from the original window center.

Subsequent regions are marked in this way until all CLBs with a congestion label larger than the target channel-width constraint are marked. The force-directed move ensures the depopulation window region is repositioned so the CLB label peak value is still captured, but it will also capture as many other CLBs as possible that need depopulation. A list of all such regions is then sorted such that the regions with the highest average congestion are depopulated first. CLBs from overlapping regions will belong to the region which is depopulated first. At each depopulation step, we determine the number of CLBs which will be added. The regions are depopulated in sorted order until all congested regions are marked for depopulation.

B. Whitespace Insertion

We experimented with using a channel-width demand model to improve the accuracy of the whitespace insertion. We will term this scheme the Congestion-Model Multiregion (CMR) Un/DoPack. Of the interconnect models available in previous work, the most applicable to this work consist of those models which predict wire length and channel-width demands[10][11]. Recently, authors in [11] extended available models to consider the routing inflexibility inherent in FPGAs. The advantage of this model is that we can use it to predict outcomes in interconnect demand based on decisions made during clustering. Since this model assumes its application to an entire FPGA, we will be making some simple assumptions to apply it to a local region.

1) Modeling Regional Interconnect Demand: While most channel-width demand models predict the interconnect demand at a global level, we are interested in determining, on a region-by-region basis, how much whitespace insertion is necessary for each congested region to become routable. Thus we create the following simple model of local interconnect demand to allow the application of a global model.

$$total\_region\_demand = region\_internal\_demand + region\_external\_demand \quad (2)$$

Interconnect demand for a region of CLBs can be generally categorized in two types: internal interconnect, which is the interconnect needed to route to any CLB located inside a region, and external interconnect, which consists of routing that connects CLBs outside the region, but pass through the region without connecting to any CLBs inside the region.

While depopulation directly affects the internal interconnect demand through whitespace insertion, we cannot directly reduce interconnect demand from external routing by whitespace insertion into a region. Instead we must account for its effects by identifying the contribution to the channel-width demand inside a region caused by these external nets. This is shown in Equation 2. Our goal is to reduce total_region_demand to meet our channel-width constraint.

We apply a congestion estimation model, Wirelength-per-Area [12], to the internal and external nets separately. This gives an indication of the relative amount of interconnect demand from each type, in each region.

For simplicity, we assume that subsequent iterations will produce relatively the same amount of external-net congestion in the next iteration. Although this is a simplification, interconnect demand from external nets are typically less than interconnect from internal nets. This is intuitive since local routing should mostly originate in local logic. We leave the influence of inter-region effects on congestion to future work. The channel-width estimation model using Equation 3 [11], is the used to determine the amount of whitespace to insert for the next reclustering step.

2) Modeling Internal Demand: The interconnect model presented in [11] is shown in Equation 3.

$$\begin{aligned} W &= W_{abs\_min} \\ &+ \frac{1}{\beta}\left(\frac{W_{abs\_min}}{F_s}\right)\left(\frac{W_{abs\_min}}{F_{Cin}}\right)^{\alpha_{in}}\left(\frac{W_{abs\_min}}{F_{Cin}}\right)^{\alpha_{out}} \\ &+ \frac{\lambda(L-1)}{4}\left(1+\frac{1}{F_{Cin}^{\alpha in}}\right) \end{aligned} \quad (3)$$

where

$$W_{abs\_min} = p\frac{\lambda \bar{R}}{2} = \rho\lambda \quad (4)$$

In Equations 3 and 4, W is the channel width, λ is the average number of used inputs per CLB, $\overline{R}$ is the average point-to-point distance for two terminal nets, F_s is switch block flexibility, F_{Cin} and F_{Cout} denote connection block flexibilities for inputs and outputs, and L is wire segment length. α_{in}, α_{out}, and p are empirically determined constants. We use the values for constants presented in [11]. As in [11], we assume that $\overline{R}$ remains constant across different cluster sizes.

Applied to the CLBs in a local region, we can use this model to predict the internal channel-width demand for a region. We wish to determine the amount of whitespace insertion required for a region to become routable. We note that the channel-width required depends on two factors: properties of the architecture, shown as the constant terms in Equation 3, and properties of the region, shown as ρ in Equation 4.

After routing, we measure the peak channel-width value and average number of used inputs per CLB to solve for the ρ in Equation 4 for each region. We then substitute our target internal channel-width value and solve for λ. The resulting value for λ allows us to set a target for the clustering step. In our implementation, the clustering tool iteratively reclusters a region with a progressively lower cluster-size until the average number of used inputs constraint is met. Since the clustering step runtime is very small, the increase to overall runtime is negligible. In addition, we retain the flexibility of the clustering tool to use different clustering methods.

C. Congestion-Aware Placement

A second goal of this work is to show that a congestion-aware placement tool improves the quality of the Un/DoPack flow. While several simulated annealing based congestion-aware placement tools exist [13][14][15], work presented in [13] follows a philosophy complimentary to Un/DoPack by optimizing placement to reduce local congestion.

$$Cost = coeff * \sum_{i=1}^{N_{nets}} q[i](bb_x(i) + bb_y(i)) \quad (5)$$

The placement approach in [13], uses a congestion estimation map to create a coefficient. This coefficient is multiplied with the bounding box cost function in the VPR placement tool to penalize swaps which lead to congested placements. The modified cost function is shown in Equation 5. For each net i, the horizontal and vertical span are added and multiplied with the q(i) factor which compensates for the fanout of the net. This is summed over all nets and multiplied with the coefficient calculated from the congestion map. The coefficient is calculated using Equation 6.

$$coeff = \left(\frac{\Sigma_{i,j}U_{i,j}^2}{nx \cdot ny} / \left(\frac{\Sigma_{i,j}U_{i,j}^2}{nx \cdot ny}\right)^2\right) \quad (6)$$

In Equation 6, $U_{i,j}$ is a CLB label in the congestion estimation map used. The congestion estimation map in [12] uses an approach referred to as Bounding Box Overlap. The congestion estimation map indicates how many bounding boxes overlap each CLB. Since we are able to use any method to generate a congestion estimation map, we also took the opportunity to explore a slight variation to the Bounding Box Overlap heuristic. In [12], authors explained that a related heuristic, Wirelength per Area, produces a congestion map which better indicates relative local amounts of interconnect demand when compared to Bounding Box Overlap. We created congestion maps using the Wirelength per Area method and applied this to the congestion-aware placement tool.

Table I illustrates the effect of congestion-aware placement on the maximum MRCW of our benchmark suite. While much slower in runtime (Table II), results show that the number of routing tracks needed is reduced consistently.

In Table I and Table II, results were obtained by performing a typical VPR CAD flow using various placement methods. VPR Default uses the default VPR placement algorithm which combines timing-driven and wirelength-driven placement. VPR BB utilizes VPR wirelength-driven placement only. BB Overlap utilizes the Bounding Box Overlap congestion map cost function, as mentioned previously. Similarly, Wirelength per Area placement utilizes the Wirelength per Area congestion cost function in the placer. For each entry in Table I and Table II, a place and route is performed for each benchmark circuit. Routing is performed with the binary search and verify_binary_search option enabled. This allows VPR to determine the minimum channel width required to successfully route each circuit using each of the mentioned placement methods.

The runtime and maximum MRCW results are compared in Table I and Table II. We note that the Wirelength per Area method indeed performs slightly better in reducing the maximum MRCW, although both produce consistently good results. Our experiments combine our Congestion-Model Multiregion version of Un/DoPack with the congestion-aware placement, using the Wirelength per Area method. We will term this scheme CMR + CAP. The congestion-aware placement is integrated with the incremental placement tool.

TABLE I
MAX MRCW COMPARISON OF PLACEMENT SCHEMES

Circuit	VPR Default (tracks)	VPR BB (tracks)	BB Overlap (tracks)	Wirelength per Area (tracks)
stdev0	96	95	92	93
stdev002	96	93	94	86
stdev004	101	97	98	92
stdev006	89	89	90	86
stdev008	119	116	115	106
stdev010	153	150	152	139
stdev012	145	145	141	138

TABLE II
RUNTIME COMPARISON OF PLACEMENT SCHEMES

Circuit	VPR Default (seconds)	VPR BB (seconds)	BB Overlap (seconds)	Wirelength per Area (seconds)
stdev0	2406	783	10063	10918
stdev002	2207	882	10226	10639
stdev004	2170	831	8792	9694
stdev006	1704	692	8794	9803
stdev008	1810	776	9602	10204
stdev010	2279	1037	10394	11942
stdev012	1875	1063	12126	13808

IV. Methodology

This section discusses the experimental framework to evaluate our algorithmic improvements.

The results in this work are compared with baseline Un/DoPack flow presented in [8]. Baseline Un/DoPack has the following characteristics:

- Congestion calculator with single region depopulation
- Clustering algorithm which is a replica of iRAC [4]
- Incremental Placer presented in [8], using default VPR placement (timing and wirelength-driven placement)
- VPR flags: pres_fac_mult 1.3, max_router_iterations 100
- FPGA architecture with LUT size k = 6, cluster-size N = 16, inputs per cluster I = 51, and a wire length of L = 4

The experiments were conducted on a single core of a Xeon 2.6 GHz processor with 2.5GB of RAM. The maximum MRCW of each circuit, was determined from VPR with the binary search option set. The verify_binary_search option in VPR was used to ensure that the lowest routable channel width was measured. All versions of Un/DoPack were run on our servers including the following Un/DoPack schemes: Baseline, Fine-Grained, and Congestion Model Multiregion. All VPR simulations used an overuse penalty factor growth factor, pres_fac_mult, of 1.3 and the maximum number of router iterations, max_router_iterations, set at 100. The channel-width constraints are the same as those presented in [9] using the benchmark circuits described below.

We used the benchmark circuits from [1]; architectural parameters were chosen to match [1]. Each of the five benchmark circuits have the following characteristics: 40013 LUTs, 241 inputs, 120 outputs, and approximately 52000 nets. These circuits are designed to help examine the performance of CAD tools for large, system-on-chip designs which are composed of subcircuits with varying amounts of interconnect demand. The benchmark circuits created in [1] were generated by using the synthetic benchmark generator GNL. GNL allows the user to specify the overall Rent parameter of the circuit, and also the Rent parameter of individual subcircuits. The average Rent parameter is 0.65 for each benchmark circuit, but the Rent exponents for each of the subcircuits are chosen to set a desired standard deviation in the Rent exponent. In this work, these subcircuits are created to match the characteristics of MCNC benchmark circuits [16]. The result is a set of benchmark circuits where some have uniform local interconnect demand across the circuit, while others have regions of high local interconnect demand.

A separate set of experiments were performed to examine the effect of combining a congestion-aware placement tool with the Congestion-Model Multiregion Un/DoPack approach. All parameters for this set of experiments are identical to those mentioned above, with the exception of the inclusion of the congestion-aware placement cost function in the incremental placer.

V. Results

We compared the runtime and area performance of Baseline Un/DoPack to the following strategies: Multiregion Un/DoPack, Fine-Grained Un/DoPack, and the Congestion-Model Multiregion Un/DoPack. We also performed experiments which combined the Congestion-Model Multiregion Un/DoPack approach with a congestion-aware placement tool.

A. Previous Methods

Our results show that of all the methods tested, the Fine-Grained approach produces the best results in terms of area growth; the increase in area is reduced by 30%, when compared to Baseline Un/DoPack. However, this had a large runtime penalty; due to the small number of CLBs inserted at every iteration, many iterations are needed, especially for lower target channel-width constraints. The runtime increases accordingly. Multiregion Un/DoPack, attempted to reduce runtime by depopulating multiple regions simultaneously, while inserting whitespace proportional to the peak congestion value of a region. As expected, Multiregion Un/DoPack was able to outperform the runtime of Baseline Un/DoPack, improving runtime by up to 6x. Area performance was also improved in general. Total area was improved by up to 17.5% over Baseline Un/DoPack. However, unlike Fine-Grained Un/DoPack, which consistently improved area results, the Multregion version also produced worse area results, inflating area up to 32% more than Baseline Un/DoPack.

B. Multiple Region Depopulation with Congestion-Model Driven Whitespace Insertion

Figures 2 and 3 show examples of typical results. Shown is the area and runtime of other methods compared to Baseline Un/DoPack. Area is measured as the total transistor area of the logic and routing of the CLBs used. This captures the effect of successfully reducing the channel width as well as the effect of using more CLBs after depopulating. Area is calculated in the same way as VPR [7], where the layout area of an individual transistor is expressed in units of minimum-width transistor areas. Runtime figures presented are normalized the the maximum MRCW for the circuit. A value of 0.6 on the x-axis means that the final routed channel width is reduced by 40%.

Results for other circuits were similar to circuit stdev004. In situations where the maximum MRCW is large (close to 1.0), the performance of all approaches will be similar. This is because only a small number of regions need to be depopulated for the circuit to become routable. Therefore at these large channel width constraints, the number of iterations required for each approach is similar, and so the runtime is also similar. As the maximum MRCW is lowered, Baseline and Fine-Grained Un/DoPack perform less depopulation at each iteration than the Multiregion or Congestion-Model Multiregion approaches. This allows our approach to have a significant speedup at lower maximum MRCWs. In addition, our Congestion-Model Multiregion approach reduces the over-depopulation of regions and area performance is improved even at low target channel widths.

Overall, results show that with CMR Un/DoPack, runtime is reduced significantly while area performance is also favorable. Our runtime results are comparable to Multiregion Un/DoPack and we are able to outperform the runtime of the Multiregion approach in some instances. More importantly, the area performance of the Congestion Model Multiregion version are better than Multiregion Un/DoPack on a consistent basis. Area performance, was worse than Baseline Un/DoPack in a very few instances, up to 6.3%. Using this approach, we are able to obtain up to 5.5x speedup over baseline Un/DoPack and an area reduction of up to 20% over baseline Un/DoPack.

Figure 4 illustrates the area increase and runtime increase of different schemes when compared to Baseline Un/DoPack. Each data-point compares the runtime and area of a particular scheme for a circuit and channel width constraint, normalized to Baseline Un/DoPack. Data points for all five benchmark circuits at all channel-width constraints [9] are shown in Figure 4. We note that CMR Un/DoPack improves both runtime and area simultaneously, unlike Fine-Grained and

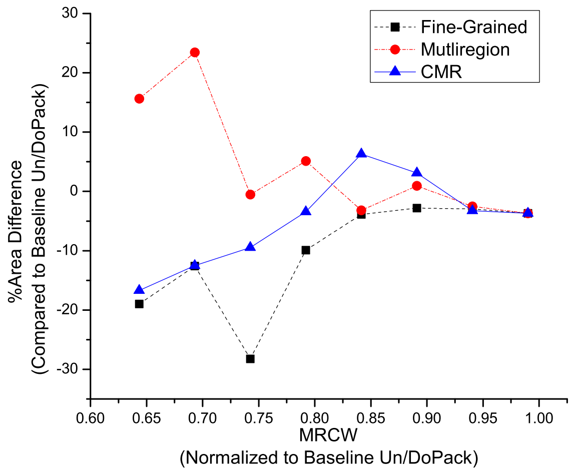


Fig. 2. Stdev004 - Area Comparison with Baseline Un/DoPack - circuit stdev004

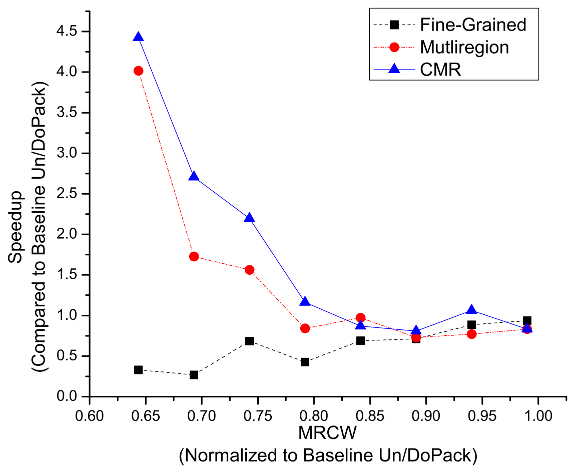


Fig. 3. Runtime Comparison with Baseline Un/DoPack - circuit stdev004

Multiregion Un/DoPack. Fine-Grained Un/DoPack achieves the best area at the expense of runtime while Multiregion Un/DoPack improves runtime at the expense of area. In some situations, CMR Un/DoPack may perform worse than Baseline Un/DoPack in runtime and area. This is because at larger channel-width constraints, the number of Un/DoPack iterations and the amount of whitespace insertion needed is similar for different schemes. Therefore, noise in the results can account for the performance difference in these cases.

C. Congestion-Aware Placer

Results show that the inclusion of a congestion-driven placement tool can further improve the Un/DoPack approach. Figures 5 to 7 illustrate that combining CMR Un/DoPack with a congestion-aware placer, consistently improves area. The CMR Un/DoPack curves shown in Figures 5 to 7 are

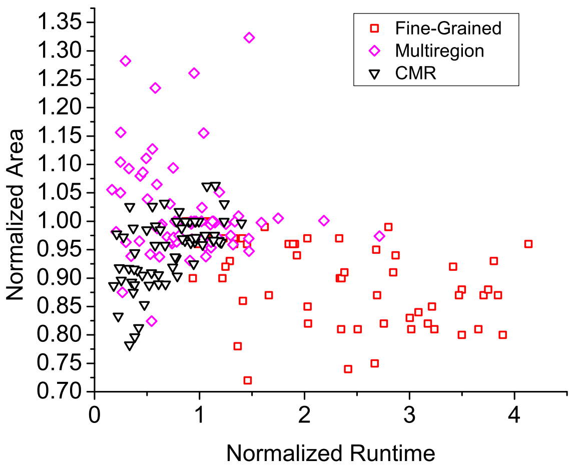


Fig. 4. Area and Runtime Tradeoff Comparison Across all Benchmark Circuits and Channel-Width Constraints

the same as that shown in Figures 2 and 3. Although the congestion-aware placement tool cannot by itself reduce the channel-width more than is possible with the Un/DoPack flow, when congestion-aware placement is combined with the CMR Un/DoPack approach, we are able to further obtain area improvement. This improvement can be significant, up to 25%, as shown in Figure 6. This illustrates that the inclusion of local congestion-driven techniques can further improve the Un/DoPack approach. In this work, the chosen congestion-aware placement method increased runtime significantly. We leave the implementation of a faster congestion-aware placement approach for future work.

VI. Conclusion and Future Work

In this paper we show that we are able to effectively improve the performance of the Un/DoPack flow in area and runtime, by effectively utilizing congestion information. This work presented methods to better select congested regions on an FPGA and calculate the amount of depopulation required at each iteration. Our Congestion-Model Multiregion approach is shown to improve runtime by a factor of 5.5 times and reduce area by up to 20%, compared to Baseline Un/DoPack. In our work, we reduced channel width up to 55%. We also showed that this CAD flow is complementary to using a congestion-aware placement method. The addition of a congestion-aware placement tool had positive effects on the overall area performance of Un/DoPack.

This work shows that an approach which utilizes spatial congestion information can be effective at determining how to depopulate. This work can be extended by considering the effects in subsequent iterations such as the influence of neighboring regions. Further consideration of other congestion-aware techniques should yield additional benefits.

Acknowledgment

The authors would like to acknowledge Altera Corporation and the Natural Sciences and Engineering Research Council of Canada (NSERC) for funding this work. They would also like to thank David Grant and Usman Ahmed for their assistance and feedback.

References

[1] 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 *IEEE/ACM International Conference on Computer-Aided Design*. San Jose, California: ACM, 2006, pp. 680–687.

[2] A. DeHon, “Balancing interconnect and computation in a reconfigurable computing array (or, why you don't really want 100% LUT utilization),” in *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*. Monterey, California, United States: ACM, 1999, pp. 69–78.

[3] R. Tessier and H. Giza, “Balancing logic utilization and area efficiency in FPGAs,” in *International Conference on Field-Programmable Logic and Applications*. Springer-Verlag, 2000, pp. 535–544.

[4] A. Singh and M. Marek-Sadowska, “Efficient circuit clustering for area and power reduction in FPGAs,” in *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. Monterey, California, USA: ACM, 2002, pp. 59–66.

[5] E. Bozorgzadeh, S. Ogrenci-Memik, and M. Sarrafzadeh, “RPack: routability-driven packing for cluster-based FPGAs,” in *Asia and South Pacific Design Automation Conference*. Yokohama, Japan: ACM, 2001, pp. 629–634.

[6] J. Cong and Y. Ding, “FlowMap: an optimal technology mapping algorithm for delay optimization in lookup-table based FPGA designs,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, no. 1, pp. 1–12, 1994.

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

[8] D. Leong and G. Lemieux, “RePlace: an incremental placement algorithm for field-programmable gate arrays,” in *International Conference on Field Programmable Logic and Applications*, 2009.

[9] M. Tom, “Channel width reduction techniques for System-on-Chip circuits in field-programmable gate arrays,” Master's thesis, University of British Columbia, April 2006.

[10] A. Gamal, “Two-dimensional stochastic model for interconnections in master slice integrated circuits,” *IEEE Transactions on Circuits and Systems*, vol. 28, no. 2, pp. 127–138, 1981.

[11] W. M. Fang and J. Rose, “Modeling routing demand for early-stage FPGA architecture development,” in *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*. Monterey, California, USA: ACM, 2008, pp. 139–148.

[12] D. Yeager, D. Chiu, and G. Lemieux, “Congestion estimation and localization in FPGAs: a visual tool for interconnect prediction,” in *International Workshop on System Level Interconnect Prediction*. Austin, Texas, USA: ACM, 2007, pp. 33–40.

[13] Y. Zhuo, H. Li, and S. Mohanty, “A congestion driven placement algorithm for FPGA synthesis,” in *International Conference on Field Programmable Logic and Applications*, 2006, pp. 1–4.

[14] G. Parthasarathy, M. Marek-Sadowska, A. Mukherjee, and A. Singh, “Interconnect complexity-aware FPGA placement using Rent's rule,” in *International Workshop on System-Level Interconnect Prediction*. Sonoma, California, United States: ACM, 2001, pp. 115–121.

[15] U. Brenner and A. Rohe, “An effective congestion driven placement framework,” in *International Symposium on Physical Design*. San Diego, CA, USA: ACM, 2002, pp. 6–11.

[16] MCNC, “LGSynth93 benchmark suite,” Microelectronics Centre of North Carolina, Tech. Rep., 1993.

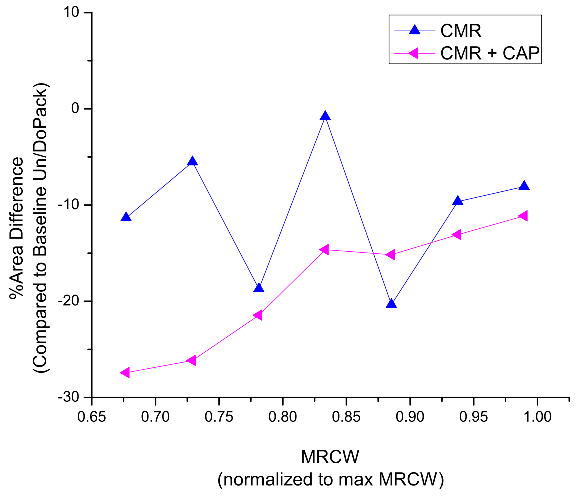


Fig. 5. Comparison of CMR Un/DoPack and CMR + CAP - Circuit Stdev0

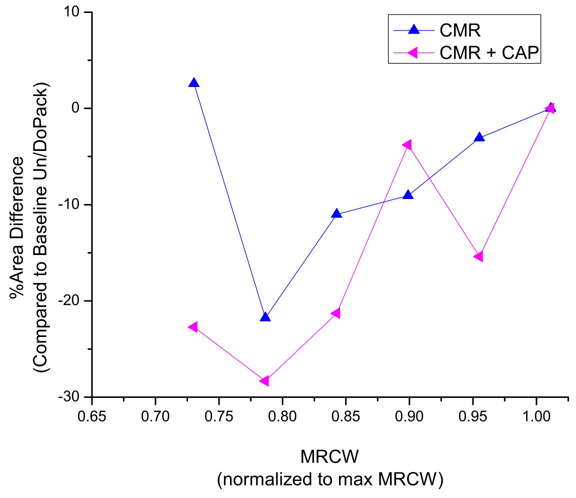


Fig. 6. Comparison of CMR Un/DoPack and CMR + CAP - Circuit Stdev006

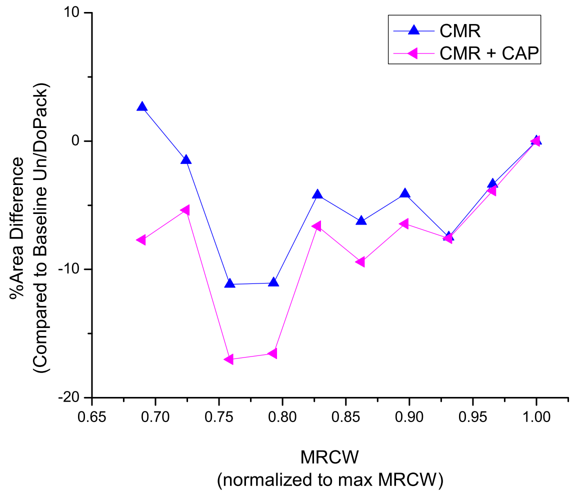


Fig. 7. Comparison of CMR Un/DoPack and CMR + CAP - Circuit Stdev012