Evaluation of Hierarchical FPGA partitioning methodologies based on architecture Rent Parameter

Zied Marrakchi, Hayder Mrabet and Habib Mehrez
Dept ASIM-LIP6
Université Paris 6, Pierre et Marie Curie
4, Place Jussieu, 75252 Paris, France
{zied.marrakchi, hayder.mrabet, habib.mehrez}@lip6.fr

ABSTRACT

The complexity of circuits to implement on FPGA has necessitated to explore hierarchical interconnect architectures. A large body of work shows that a good partitioning hierarchy, as measured by the associated Rent parameter, will correspond to an area-efficient layout. We define the architecture Rent parameter of a netlist to be the lower bound on the Rent parameter of any partitioning hierarchy of the netlist. Experimental results show that a combination between a multilevel bottom-up clustering and a top-down refinement generates partitioning hierarchies whose Rent parameters are lower than those of other methods.

1. INTRODUCTION AND PREVIOUS WORK

Mesh is the most studied and most used industrial topology. This style has been the subject of considerable published work by Rose et al. [2]. The mesh structure consists of an array of logic blocks with I pins in each side, which can be linked together thanks to uniform horizontal and vertical programmable routing channels [2].

There is little published research on Hierarchical FPGA (HFPGA) architecture. In a HFPGA , logic blocks and routing resources are organized into levels. A $level_i$ cluster has W_i IO (Input/Output) wire tracks and contains a set of k $level_{i-1}$ clusters connected with $level_i$ switch box.
A common way to compare both architectures is the switching requirement using Rent's Rule.

$$IO = cN^p$$

This empirical relationship links the number of IO with the N gates of a design. In our case of HFPGA, N corresponds to the number of logic blocks linked together. c is a constant factor that corresponds to the IO size of the logic block, and p defines the growth rate.
In his work, DeHon [4] has showed that the k-HFPGA has more efficient switching that grows as $O(1)$ if $p < 0.5$. Once $p > 0.5$ the mesh architecture becomes more interesting. Mesh switching and wiring area grows as $O(N^{p-0.5})$ while the k-HFPGA switching resources diverge and grows as $O(N^{2p-1})$.
Thus the Rent parameter is an accurate indicator of the wiring and switching requirements for a given partitioning hierarchy. In particular given a choice between two partitioning tools, the one with lower Rent parameter will require less switching and wirelength and correspond to a denser final layout. Thus, one aspect of our present work compares various partitioning methods with the goal to identify the partitioning strategy leading to the optimal hierarchy. This affords a new methodology for comparing the utility of multilevel partitioning techniques.

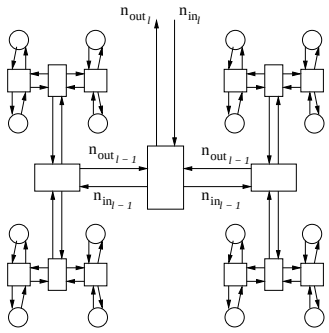

Figure 1: hierarchical interconnect

2. PROPOSED ARCHITECTURE

To make size estimate of the architecture more concrete, let us consider a specific structure build according to Rent's rule. We build a fully hierarchical interconnect with inter-level signaling bandwidth growing according to Rent's Rule. To simplify analysis, we consider only unidirectional signal wires.

The gates are recursively partitioned into k equally sized sets at each level of the hierarchy. The principal interconnect occurs at each node of convergence in the hierarchy (see figure 1). At a level l in the hierarchy, each node has a fan-in from below of $k * n_{out_{l-1}}$ signals and a fan-in from above of n_{in_l}. Similarly, it has a fan-out of $n * n_{in_{l-1}}$ towrd the leaves and n_{out_l} towards the root. At each level l, we have n_{LUT_l} LUTs, n_{in_l} external inputs and n_{out_l} external outputs. We have $n+1$ distinct output directions from each node of convergence in the interconnect: n for the n leaves, plus one for the root. Allowing full connectivity within each tree node, each of the n leaves picks its n_{in} inputs from the $(n-1)*n_{out}$ outputs from its siblings and from the n_{in} inputs from the parent node. The n_{out} outputs of this node are selected from the $n * n_{out}$ outputs from all n subtrees converging at this point. Figure 2 shows this basic arrangement for $n = 2$. As shown in the previous section, the Rent parameter is an accurate indicator of the wiring and switching requirements for a given partitioning hierarchy. Thus, we introduce the notion of an architecture Rent parameter. The architecture

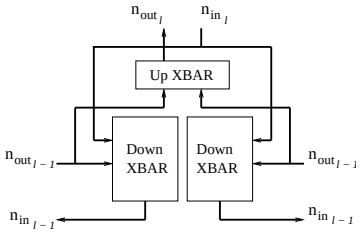

Figure 2: switching node in 2-arity Hierarchical Interconnect

Rent parameter is the minimum possible Rent parameter of the architecture allowing all routability achievement of a given netlist. So to determine the Rent parameter of a circuit we run the multilevel partitioning method, then in each level we determine the maximum number of inputs and outputs in all clusters of the same level. Finally We calculate the Rent parameter for each level according to the hierachical combining: $N_{LUT_l} = k^l$ and $w_l = C((k)^l)^p$
C is the number of LUT inputs/outputs and w_l is the number of $cluster_l$ inputs/outputs.

3. PARTITIONING METHODOLOGIES

3.1 Top-down partitioning

Top-down approaches partition a given netlist into smaller subclusters. This technique is based on the global connectivity informations and leads to a good partitioning solution. Otherwise it is a long time consumer specially with large netlist graphs. To deal with this problem, the conceptors of hMetis [6] propose to run the partition on a coarsened graph. Thus in a first step the hypergraph is coarsened successively and it is partitioned into k-parts. Then this k-way partition is successively refined as it is successively projected back into the original hypergraph. This tool was used in [8] to construct clusters in the case of the clustered mesh FPGA. In this work we have extended the application of this tool to construct a multilevel hierarchy. Our method is a top down one, so first we construct clusters of the top level and after that each cluster is partitioned into subclusters. This is done until the bottom of the hierarchy was reached. The objectif function in hMetis k-way partitioning consists on minimising the cut in the partition (min-cut).

3.2 Bottom-up partitioning

The size of the smallest multilevel hierarchical FPGA is penalised by the coarse granularity of the architecture. This means that in most cases the number of logic blocks slots in the architecture is greater than the number of instances in the netlist to implement. This can have a good effect on congestion alleviating. In fact, DeHon [3] showed that for hierarchical FPGAs, 100% logic utilisation is not necessary benefical for overall device area minimisation. His results indicate that a careful partitioning of designs and depopulation of logic clusters can result in better HFPGA resources utilisation. In the following we present two different techniques to distribute instances over clusters. In both cases we use the same objectif function proposed in [1] to compute the attraction of each block to a cluster.

3.2.1 BLEs-limit strategy

Tessier [7] showed that depopulation of clusters can result in reduced channels width in the case of mesh architecture. The algorithm presented depopulates each cluster equally so there is a uniform distribution of empty BLEs across the chip. In this approach, an attempt is made to spread the logic evenly across all clusters in the device while limiting the number of inputs that logically drive each cluster to be less than the number of pins physically available. In this clustering algorithm, the number of LUTs to be held in each cluster (N_{high}, N_{low}) is first determined. These numbers reflect the overall LUT utilisation of the device and differ by only one LUT. Following this step the number of device clusters that hold each quantity of LUTs (C_{high} and C_{low}, respectively) is determined. Clustering is then performed for two types of clusters.

3.2.2 Pins-limit strategy

Singh [1] presented a clustering algorithm (iRAC) which is very effective at reducing channel width in the case of mesh architecture. iRAC also limits the number of inputs to each CLB by using the Rent parameter, resulting in solutions that have some depopulation. The aim of this technique is to alleviate routing congestion by absorbing as many nets into clusters as possible, and depopulating clusters according to Rent's rule in order to achieve spatial uniformity in the clustered netlist.
We notice that pins constraints enforcing is inefficient when it is applied in high levels. This is due essentially to the bottom-up and the greedy aspect to construct clusters of the technique. To deal with such problem, we propose to create clusters in high levels without pins constraints enforcing. As presented in figure 3, once the multilevel clustering is achieved, we run a multilevel top-down refinement.

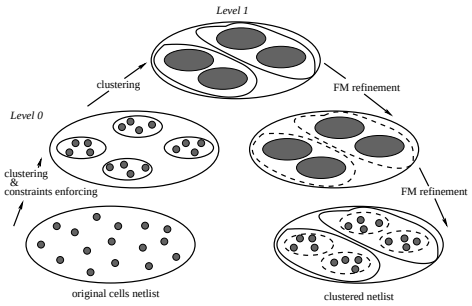

Figure 3: Multilevel clustering & refinement

3.3 Multilevel refinement

After the clustering phase (with both strategies), we obtain k clusters in each level. We can consider that the k clusters in a level present an initial solution to a k-way partitioning problem. During the refinement phase, cells will be moved between clusters (parts) to optimise an objective function without violating the constraints imposed by the cluster size. In a level, cells are not allowed to move between all clusters, because this can decrease the quality of the solution obtained in the higher level. To prevent such bad effect, cells can only move between neighboring clusters. We call neighboring clusters, all clusters in a level belonging to the same supercluster. Thus in every level, neighboring

clusters will be isolated and form a subgraph. In figure 3 those subgraphs are presented by the continued lines and partition by the dashed ones. A cell is allowed only to move across dashed lines. The objective function is local to each subgraph and corresponds to the maximum number of pins of all clusters (parts) belonging to the same subgraph. An FM algorithm [5] will be applied on a subgraph to optimise the local objective function. The complexity of our k-way refinement is reduced since we apply it for successively for each subgraph (in each subgraph there are small number of parts: Arity of the architecture).

4. EXPERIMENTAL RESULTS

4.1 Partitioning methodologies comparison

Within the preceding approaches, the main classes of partitioning are the top-down and bottom-up. The following algorithms were used in our experiments:

- TD: Top-Down partitioning based on hMetis (mincut).
- BU-B : Bottom-Up clustering with BLE-limit.
- BU-P : Bottom-Up clustering with Pins-limit.
- BU-B-R: A BU-B followed by the refinement phase.
- BU-P-R: A BU-P followed by the refinement phase.

The experiments were performed as follows. Each partitioning algorithm was used to construct a partitioning hiearachy for the circuit via recursive partitioning of the circuit and its subpartitions. We have used some of the MCNC benchmark circuits with different logic sizes. At each partitioning step, we noted the number of external pins for each subpartition as explained in section 2. In order to correlate the experimental data to Rent's rule, we re-express the relationship $w_l = C((k)^l)^p$ as $logw_l = logC + p.log(k)^l$ (l is the corresponding level). As shown in table 1, for each level we have an associated Rent parameter.
For the bottom-up clustering approaches we notice that the pins-limit strategy BU-P is more efficient than the BLEs-limit strategy BU-B. This seems evident since the aim of this strategy is to reduce the number of external pins of each constructed cluster.
We notice that the TD method is efficient to reduce the Rent in high levels. This approach has the disadvantage that the objectif function is not suitable to the reduction of the Rent's parameter. In fact the Min-cut, tries to reduce communication between all clusters with no regard to the signaling bandwidth of a particular cluster. The BU-P approach is more efficient to reduce the Rent's parameter in low levels. Otherwise with this technique, we obtain a bad solution when we construct the highest levels. This is due to the inefficiency of pins-limit in this stage. As shown in table 1 the solution can be improved if we run a top-down refinement phase. With the BU-P-R we obtained a good Rent parameter in all levels (lowest and highest ones).

4.2 Architectures comparison

As shown in table 1 the average architecture Rent's parameter is greater than 0.5. As demonstrated in section 1, this favorises the mesh architecture in term of wires and switch reduction. To confirm this with experimental results, we have claculated the number of needed switches in each resulting architecture.
The mesh architecture is composed of clusters and has an uniform routing network with single-length segments and a subset switch box. Each cluster contains 8 4-LUT. The number of inputs in each cluster is 18 and the number of outputs is 8. Both inputs and outputs are fully connected. The IO pads are fully connected too. We use t-vpack to construct clusters and the channel minimising VPR 4.3 to place and route the obtained netlists. We vary the IO_{ratio} to achieve the optimal array size. VPR chooses the optimal size as well as the optimal channel width to place and route each benchmark.
From table 2, we notice that the average number of needed switches in the hierarchical HFPGA is about 3 times greater than in the mesh architecture. This is due essentially to coarse granularity of the architecture and the full populated crossbar.

5. CONCLUSION

We have introduced the notion of an architecture Rent prameter defined as the lowest Rent parameter achievable by any partitioning method. Our results indicate that a combination between a multilevel bottom-up clustering and a top-down refinement generates partitioning hierarchies whose Rent parameters are lower than those of other methods. The aim of the work was to find the best partitioning method to obtain the smallest HFPGA area. Despite our effort we found that with a fully-populated crossbar, hierarchical FPGA can not be denser than mesh FPGAs. Thus to make HFPGA more competitive, we must use depopulated routing interconnect. The question is how to depopulate the hierachical interconnect and keep a good routability. This is the aim of our future work.

6. REFERENCES

[1] M. M.-S. A. Singh. Efficient circuit clustering for area and power reduction in fpgas. *Proc. FPGA'02*, February 2002.

[2] V. Betz, A. Marquardt, and J. Rose. Architecture and cad for deep-submicron fpgas. *Kluer Academic Publishers*, January 1999.

[3] A. DeHon. balancing interconnect and computation in a reconfigurable array (or why you don't really want 100% lut utilisation). *Proc. FPGA'99*, 1999.

[4] A. DeHon. Unifing mesh and tree-based programmable interconnect. *IEEE Transactions on VLSI Systems*, IEEE Transactions on VLSI Systems(12):10, October 2004.

[5] C. M. Fiduccia and R. M. Mattheyeses. A linear-time heuristic for improving network partitions. *Proc. DAC*, pages 175–181, 1982.

[6] G.Karypis and V.Kumar. Multilevel k-way hypergraph partitioning. *Design automation conference*, 1999.

[7] R.Tessier and H.Giza. Balancing logic utilisation and area efficiency in fpgas. *Int'l workshop on field programmable logic and applications*, 2000.

[8] H. Z.Marrakchi and H.Mehrez. Hierachical fpga clustering to improve routability. *Conference on Ph.D Research in MicroElectronics and Electronics, PRIME*, 2005.

Benchmark				TD		BU-P		BU-B		BU-P-R		BU-B-R	
Name	LUTs	Arch	level	Rent	T(s)	Rent	T(s)	Rent	T(s)	Rent	T(s)	Rent	T(s)
C1908	224	8x8x8	0	0.73	0.6	0.690	0.4	0.773	0.3	0.690	1.2	0.773	1.2
			1	0.616		0.563		0.597		0.511		0.585	
			Avr	0.675		0.626		0.685		0.600		0.679	
ttt2	241	8x8x8	0	0.615	1.4	0.528	0.3	0.641	0.2	0.528	1.3	0.641	1.3
			1	0.517		0.405		0.474		0.396		0.430	
			Avr	0.566		0.466		0.558		0.462		0.536	
C3540	509	8x8x8	0	0.845	4	0.773	1.76	0.773	1	0.773	2.5	0.773	1.8
			1	0.717		0.700		0.700		0.681		0.694	
			Avr	0.781		0.737		0.737		0.727		0.734	
seq	1750	8x8x8x8	0	0.828	21.3	0.773	20	0.845	12	0.773	27	0.845	20.3
			1	0.686		0.727		0.743		0.651		0.666	
			2	0.605		0.667		0.651		0.626		0.650	
			Avr	0.707		0.722		0.746		0.683		0.720	
apex2	1878	8x8x8x8	0	0.877	23	0.712	20	0.877	15	0.712	37	0.877	26
			1	0.722		0.773		0.770		0.722		0.753	
			2	0.602		0.674		0.684		0.626		0.630	
			Avr	0.734		0.720		0.777		0.687		0.754	
s298	1931	8x8x8x8	0	0.828	20	0.733	19	0.828	13	0.733	50	0.645	31
			1	0.517		0.627		0.631		0.597		0.828	
			2	0.341		0.588		0.579		0.480		0.597	
			Avr	0.562		0.650		0.679		0.603		0.509	
frisc	3556	8x8x8x8	0	0.877	56.5	0.733	54	0.810	38	0.733	75	0.810	48
			1	0.669		0.732		0.725		0.681		0.624	
			2	0.605		0.674		0.690		0.626		0.626	
			Avr	0.717		0.713		0.742		0.680		0.687	
elliptic	3604	8x8x8x8	0	0.828	56	0.712	52	0.828	35	0.712	80	0.828	45
			1	0.624		0.702		0.741		0.641		0.641	
			2	0.626		0.625		0.669		0.626		0.633	
			Avr	0.692		0.680		0.746		0.660		0.701	
spla	3690	8x8x8x8	0	0.861	61.4	0.733	59	0.845	40	0.733	84	0.773	65
			1	0.745		0.790		0.783		0.747		0.764	
			2	0.629		0.667		0.709		0.627		0.667	
			Avr	0.745		0.730		0.779		0.702		0.735	
average	**1931**			**0.687**	**27.13**	**0.672**	**23**	**0.717**	**17.1**	**0.645**	**39.7**	**0.688**	**26.6**

Table 1: Rent parameter results

Benchmark		Mesh				HFPGA		
Name	LUTs	CLBs	W	IO ratio	Switches x10^3	Arch	Occup-ation%	Switches x10^3
C1908	224	28	18	3	27.9	8x8x8	43.7	64.7
ttt2	241	31	10	2	15.4	8x8x8	47	43.3
C3540	509	65	21	2	68.4	8x8x8	99.4	317.2
seq	1750	242	41	2	410	8x8x8x8	42	1627.9
apex2	1878	263	40	1	445.4	8x8x8x8	45.9	1854.4
s298	1931	305	30	1	374.4	8x8x8x8	47	699.2
frisc	3556	737	54	2	1633.5	8x8x8x8	86	3643.8
elliptic	3604	527	46	3	950.7	8x8x8x8	87.9	3343.7
spla	3690	750	56	2	1956	8x8x8x8	90	4269.8
average	**1931**	**327.55**	**35**	**2**	**653.5**		**65.43**	**1762.6**

Table 2: Switches comparison between mesh FPGA architecture and HFPGA