ANT3D: Simultaneous Partitioning and Placement for 3-D FPGAs based on Ant Colony Optimization

Panayiotis Danassis, *Student Member, IEEE*, Kostas Siozios, *Member, IEEE*, and Dimitrios Soudris, *Member, IEEE*

***Abstract*—Three-dimensional technologies offer great promise in providing improvements in the overall circuit performance. This letter introduces a novel netlist partitioning and placement algorithm, named ANT3D, targeting 3-D reconfigurable architectures, based on ant colony optimization (ACO). Experimental results show the effectiveness of ANT3D algorithm as we achieve performance enhancement by 10% on average, compared to state-of-the-art tools, while using significantly fewer through-silicon vias (TSVs). Finally, by taking benefit from the inherent parallelism found in ACO algorithms, it is feasible to notable reduce the execution run-time of our algorithm.**

***Index Terms*—Partitioning, placement, swarm intelligence, three-dimensional FPGA.**

I. Introduction

AS THE semiconductor industry struggles to maintain its momentum down the path of Moore's Law, it is becoming clear that in addition to scaling line widths and chip sizes downwards, performance-enhancing technologies, such as through-silicon vias (TSVs), will be necessary to evolve the fast-expanding data-driven world. By stacking and interconnecting semiconductor layers vertically (3-D integration), as opposed to keep shrinking the line widths, chip designers have the potential to go around the limitations of geometric scaling; enable a significant increase in performance and reduction in power consumption through shorter signal paths; and achieve true cost reduction through the use of proven fabrication techniques [11].

The benefits of using 3-D integration in logic chips are especially great when designing field-programmable gate arrays (FPGAs), since these architectures suffer from data communication problems; the delay and power consumption of the interconnection network are the main performance bottlenecks [11]. The interest for designing 3-D FPGAs has already been addressed by the reconfigurable industry. Typical examples are the 3-D FPGAs provided by Tezzaron, as well as the 2.5-D Xilinx Virtex-7 and the UltraScale devices. Furthermore Xilinx recently announced the 16 nm UltraScale+; FPGAs which incorporate 3-D technology and exhibit a $2\times$ to $5\times$ performance-per-watt advantage over comparable systems designed using Xilinx's 28 nm devices.

Although 3-D integration seems promising, significant challenges associated with efficient circuit design have hampered its adoption and further development. A major aspect of 3-D FPGA architecture research is the development of computer-aided design (CAD) tools for mapping applications to FPGAs [1], [2]. It is well-established that the quality of a high-density-multilayer-FPGA-based implementation is largely determined by the effectiveness of the accompanying CAD tools [11].

Previous studies determined that the performance metrics of applications mapped onto 3-D FPGAs are tightly coupled to the availability of TSVs per device layer. Motivated by this challenge we developed a novel algorithm, which we propose throughout this letter, for netlist partitioning and placement onto 3-D FPGAs. The introduced algorithm relies on ant colony optimization (ACO) which exhibits inherent parallelism and thus is able to sufficiently reduce execution run-time. Instead of similar state-of-the-art approaches which focus mostly on timing improvement [1], [2], the solutions retrieved with the ANT3D algorithm provide higher routing flexibility in order to maximize performance metrics. Specifically we have achieved to reduce the number of utilized TSVs 10% (on average), which in turn leads to higher maximum operating frequency by 10% on average. Additionally the introduced coarse-grained OpenMP-based parallel implementation of the proposed algorithm leads to mentionable speedup of the execution run-time by exploiting the underlying multicore architectures.

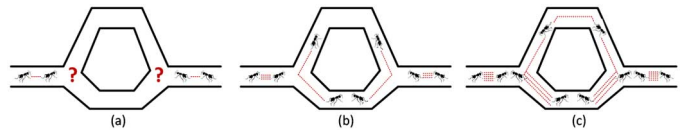

Fig. 1. How ants find the optimum solution. (a) Ants arrive at a decision point. (b) They choose probabilistically paths marked with strong pheromone values. Initially all the paths have the same probability to be chosen. (c) Since ants move at constant speed, the ants that chose the shorter path reach the destination faster than those who chose the longer path. Thus, pheromone is accumulated at a higher rate on the shorter path.

II. ANT3D: Simultaneous Netlist Partitioning and Placement Algorithm

This section introduces the proposed algorithm for addressing the netlist partitioning and placement problems in FPGAs. The proposed algorithm relies on the ACO engine and simulates the foraging behavior of certain ant species which use a chemical compound called pheromone, in order to mark favorable paths for the subsequent ants to follow [6], as it is depicted in Fig. 1. The introduced implementation incorporates concepts from both the MAX-MIN ant system (MMAS) and the ACS.

Manuscript received September 28, 2015; accepted January 27, 2016. Date of publication February 11, 2016; date of current version May 25, 2016. This work was supported in part by the TEACHER project funded by DAAD (2014). This manuscript was recommended for publication by M. Balakrishnan.

The authors are with the School of Electrical and Computer Engineering, National Technical University of Athens 15780, Greece (e-mail: panosd@microlab.ntua.gr; ksiop@microlab.ntua.gr; dsoudris@microlab.ntua.gr).

Color versions of one or more of the figures in this letter are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/LES.2016.2527925

To be precise, given a hypergraph $G(V, E)$ where each vertex $v \in V$ and hyperedge $e \in E$ have weights $s(v)$ and $w(e)$ respectively, the problem we address throughout this letter is to divide the set V into k balanced subsets, where k is the number of 3-D FPGA layers, such that the total weight of hyperedges being cut (i.e., spanning different subsets) is minimized. Then these vertices are assigned to exact locations per layer, so that there is no overlap among the vertexes while improving timing critical networks.

Algorithm 1 gives the pseudo-code for the proposed ANT3D algorithm. In contrast to similar heuristic approaches (i.e., simulated annealing, directed force, etc) where the solution's quality is gradually improved, the ANT3D algorithm generates a new solution from scratch per iteration, until the termination criterion is met[1]. In particular, during each iteration the $ant[m]$ places netlist's functionality i (i.e., vertex v_i) to the logic block j on layer k according to the probability (p^m_{ijk}) given by (1). For faster convergence, this probability takes into consideration both the *heuristic information* (η_{ijk}) and the *pheromone trails* (τ_{ijk}) [6]

$$p^m_{ijk} = \frac{[\tau_{ijk}]^\alpha \times [\eta_{ijk}]^\beta}{\sum_{(y,z)\in N^m} [\tau_{iyz}]^\alpha \times [\eta_{iyz}]^\beta} \quad (1)$$

where N^m are the available (i.e., nonutilized) logic blocks regarding the solution computed by $ant[m]$, and α, β are tunable parameters that define the importance of heuristic information (η_{ijk}) and pheromone trails (τ_{ijk}) respectively. Based on our exploration we have found that we can achieve mentionable speedup if only a portion of the solutions is retrieved according to Eq. (1). Specifically, we use (1) with probability $(1 - q_0)$, while with probability q an ant assigns the netlist's functionality i to the optimum location found so far J at layer K, mentioned as (J, K), according to (2). Regarding our implementation, the q_0 was set to 0.95 since this value rapidly decreases the convergence time without compromising the quality of the final placement. In other words, an ant makes the best possible move as indicated by the accumulated pheromone trails and the heuristic information with probability q_0, while with probability $(1 - q_0)$ the pheromone biases the ants towards unexplored promising regions

$$(J, K) = \underset{(y,z)\in N^m}{\arg\max} \left\{ [\tau_{iyz}]^\alpha \times [\eta_{iyz}]^\beta \right\}. \quad (2)$$

The quality of a solution computed by $ant[m]$ is evaluated based on the cost function described by (3), which takes into account the critical path delay ($Cost_{\text{Timing}}$) and the total wirelength ($Cost_{\text{Wiring}}$)

$$Cost = \lambda \times Cost_{\text{Wiring}} + (1 - \lambda) \times Cost_{\text{Timing}} \quad (3)$$

$$Cost_{\text{Wiring}} = \sum_{i=1}^{\#\text{nets}} q(i) \times [bb_x(i) + bb_y(i) + bb_z(i)] \quad (4)$$

[1]Our algorithm terminates its execution when the improvement at placement's quality is less than 2% for the last 5% of the total number of the algorithm's iterations.

where $bb_x(i), bb_y(i)$ and $bb_z(i)$ are the dimensions of the bounding cube across the x, y and z axes respectively regarding the net i, while $q(i)$ is a factor that compensates for the underestimation of the required wire by the above model [5]. Although more advanced wire-length models exist, such as the rectilinear minimum spanning tree (RMST) and the rectilinear steiner minimal tree (RSMT), they exhibit limited performance [8] compared to the employed half perimeter wire-length (HPWL) in regard of the combined netlist partitioning and placement problem studied throughout this letter. As to the timing analysis we employ the cost functions initially proposed at [7]. By tackling the netlist partitioning and placement problems simultaneously, it is more likely to avoid being trapped at local minima, as it is validated in Section III.

Algorithm 1 Pseudo-code of our ACO-based ANT3D.

1:**procedure** ANT3D(*netlist* file)
2:Initialize *Pheromone* matrix
3:Compute *StaticHeuristic* matrix
4:**while** !terminationCondition() **do**
5:**for** *ant [m]* in the *colony* **do**
6:Compute *Dynamic Heuristic* matrix
7:$q \leftarrow rand[0, 1]$
8:**if** $(q \geq q_0)$ **then**
9:Construct solution (according Eq. (1))
10:**else**
11:Construct solution (according Eq. (2))
12:**end if**
13:Evaluate solution's quality (according Eq. (3))
14:*Local pheromone* update (according Eq. (5))
15:**end For**
16:*Global pheromone* update (according Eq. (7))
17:**end while**
18:**return** *solution* file
19:**end procedure**

Depending on the quality of the solution, $ant[m]$ deposits different amounts of pheromone at the utilized resources to guide subsequent ants. Additionally our algorithm incorporates the concept of pheromone evaporation to avoid the trapping to local minima. The following subsections provide additional details about the implementation of these tasks.

A. Heuristic Information and Pheromone Trails

The heuristic information guides the ant's selections at the early stages of the algorithm, before the accumulated pheromone starts to take effect. For additional flexibility we employ two types of heuristic information, namely *static* and *dynamic*. Specifically, the static heuristic, which is a priori defined, tends to assign to spatially close locations blocks with increased number of connections [13]. On the other hand, the dynamic heuristic is more aggressive and aims to minimize the Manhattan distance for the blocks belonging to the same netlist's path (hyperedge).

Succeeding the early stages, the ants have deposited enough pheromone to the optimum trails to guide the subsequent ants. The initialization of those trails is firmly connected to the efficiency of ANT3D algorithm, as they highly influence the performance metrics and the speed of convergence [6]. With respect

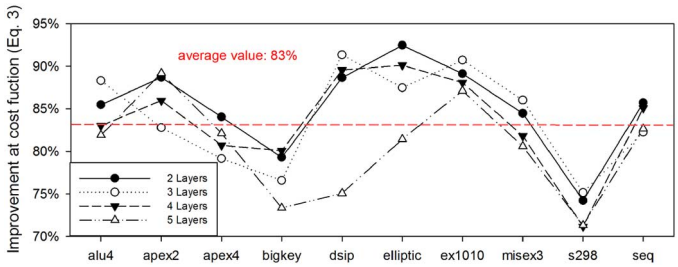

Fig. 2. Variation of cost function during the algorithm's execution.

to the physical implementation problem studied throughout this letter, the pheromone is initialized according to $\tau_0 = 1/\rho C^*$, where C^* denotes the cost of the best solution we forecast to achieve (i.e., $C^* = \min\{Cost\}$) and ρ denotes the evaporation rate. In order to determine the C^* value, we performed a detailed exploration on various benchmarks and different 3-D FPGAs. The results from this exploration are summarized in Fig. 2 where the vertical axis gives in a normalized manner the improvement of ANT3D's cost function [see (3)] as compared to its initial value (C^{init}). Based on that analysis we have found that $C^* = 0.83 \times C^{\text{init}}$ on average. Finally ρ is set to 0.5, as this value improves the time of convergence.

B. Pheromone Update

Another crucial task for the ANT3D algorithm is the way that pheromone trails are updated in order to guide the colony's ants towards promising solutions. Specifically the algorithm increases the pheromone trails associated with good solutions, while it decreases, using the evaporation mechanism, the trails associated with solutions that do not improve the cost function metric. For this purpose, two types of pheromone update rules are employed: a global and a local one.

Global Pheromone Trail Update: The global pheromone update rule, performed once per iteration, evaporates pheromone from all the paths and deposits new pheromone to those that are associated with promising solutions.

For the deposition of new pheromone both the $best\_so\_far$ and the $iteration\_best$ solutions are used, in an alternate way, and the relative frequency with which we apply either rule has an influence on how focused our search is. Specifically when the pheromone is updated according to the $best\_so\_far$ solution the algorithm focuses mostly on solutions similar to the global best, whereas using the $iteration\_best$ solution guarantees that a greater number of trails receive pheromone and consequently a wider exploration of search space is feasible. Thus carefully selecting the optimum combination between them is necessary to achieve fast convergence and high quality results. Experimental results show that the optimum performance is obtained when more emphasis is given to the $best\_so\_far$ solution. Eq. (5) gives the pheromone update function for the ANT3D algorithm

$$\tau'_{ijk} = \begin{cases} \tau_{\min}, & \text{if } \tau'_{ijk} < \tau_{\min} \\ \tau_{\max}, & \text{if } \tau'_{ijk} > \tau_{\max} \\ (1-\rho) \times \tau_{ijk} + \frac{1}{C^{\text{best}}}, & \text{if } ant^{\text{best}} \text{ assign } i \to (j,k) \\ (1-\rho) \times \tau_{ijk}, & \text{otherwise} \end{cases} \quad (5)$$

where τ' is the updated pheromone value, C^{best} refers to the cost of ant^{best}'s solution, while the exponent $best$ corresponds either to the $best\_so\_far$ or the $iteration\_best$ solution, depending on the employed rule. Lower ($\tau_{\min}$) and upper ($\tau_{\min}$) trail limits are defined as follows:

$$\tau_{\min} = \frac{\tau_{\max}}{const} \quad \text{and} \quad \tau_{\max} = \tau_0 = \frac{1}{\rho \times C^*} \quad (6)$$

Note that the lower and upper limits found in Eq. (5) are necessary to to avoid trapping in local minima.

Local Pheromone Trail Update: In combination with the global pheromone rule, we have also adopted the concept of the local pheromone update. Specifically the local pheromone update rule is performed when the $ant[m]$ computes a concrete solution (i.e., netlist's partitioning and placement) in order to discourage the upcoming ants of the same colony from computing a similar solution, which leads to an expansion of the exploration space. The local pheromone update is performed according to (7), where $\xi = 0.5$ refers to the local pheromone evaporation rate

$$\tau'_{ijk} = \begin{cases} \tau_{\min} & \text{if } \tau'_{ijk} < \tau_{\min} \\ (1-\xi) \cdot \tau_{ijk} & \text{if } ant[m] \text{ assign } i \to (j,k) \end{cases}. \quad (7)$$

C. Parallel Implementation

Since ants move concurrently and asynchronously, ACO algorithms are inherently parallelizable both in the data and population domains [6]. Although the mechanism of local pheromone update might lead to communication overhead, we can overcome this by implementing a coarse-grained approach with a rather rare information exchange. Specifically in our implementation we employ the same number of ants (compared to those employed at the single-core, single-thread execution), which move in parallel and exchange information only at the end of every iteration through the global pheromone update mechanism. The parallel flavor of our introduced algorithm has been implemented with the OpenMP API. Thus by assigning these ants to multiple cores it is feasible to achieve mentionable speedup.

III. Experimental Results

This section provides a number of quantitative results that highlight the effectiveness of the introduced algorithm in comparison to the state-of-the-art algorithms, namely the hMetis netlist partitioning [12] and the TPR placement [2]. The comparison was performed with circuits from the 20 biggest MCNC benchmarks. The 3-D FPGAs consist of up to five layers, each of which follows the homogeneous island-style architecture. For our experimentation the targeted 3-D devices exhibit average resource (i.e., logic) utilization ranging between 92%–97%. The targeted device per benchmark was identical to both the introduced (ANT3D) and the existing (hMetis and TPR) algorithms. The technology parameters for each layer correspond to 65 nm CMOS technology, while the electrical equivalent characteristics for TSVs are extracted from state-of-the-art solutions published in relevant literature [9]. Finally the physical parameters quantified on the rest of this section are retrieved after successful

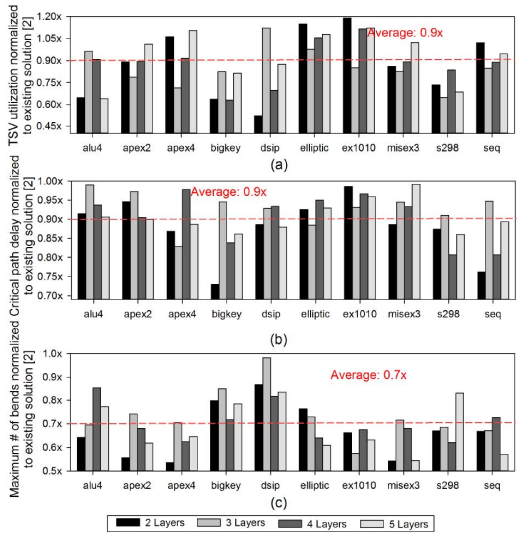

Fig. 3. Evaluation in terms of: (a) critical path delay; (b) utilization of TSVs; and (c) reduction of routing bends.

netlist routing with the Pathfinder negotiated congestion/delay algorithm [10]. Unfortunately we cannot provide comparisons against the rest of the flows found in relevant literature for 3-D FPGAs as these tools are not publicly available.

The efficiency of the introduced framework relies on a better manipulation of the available resources (TSVs), compared to the existing tools, which provides spatial locality to resources assigned to adjacent layers. To visualize this, Fig. 3(a) plots the gain in terms of utilized TSVs required for successful netlist routing. These results indicate that the ANT3D algorithm achieves to perform benchmark implementation with 10% fewer TSVs on average, as compared to the existing state-of-the-art tools [2], [12].

The shorter routing paths found in 3-D architectures impose lower resistance (R) and capacitance (C) values, which in turn lead to higher operation frequency, as it is summarized in Fig. 3(b). Specifically we have managed to reduce the critical path delay by 10% due to the better manipulation of wire-length and timing metrics (instead of retrieving a min-cut solution similar to existing state-of-the-art hMetis tool [12]). Note that the timing analysis for the alternative flows is performed using the Elmore delay model [10], in view of its high fidelity with respect to the actual circuit delay [4].

The increased flexibility of ANT3D algorithm also leads to routability improvement. In order to evaluate this metric, Fig. 3(c) gives the number of bends that are required for successful netlist routing. As we might conclude by this figure, the solutions retrieved from proposed algorithm impose 0.7× fewer bends on average, as compared to the existing TPR tool. In addition this gain is expected to improve the run-time for solving the netlist routing problem.

Regarding the parallel flavour of our algorithm, we achieved over 70% of the maximum theoretical speedup, as derived from the Amdahl's Law [3], as depicted in Fig. 4. Specifically this analysis refers to different number of ants per colony, while the number of threads span between 1–16. Note that for this experimentation our CPU is the Intel Xeon E5-2650 (8-cores/16-threads).

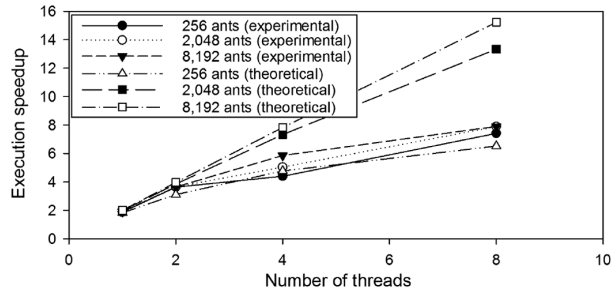

Fig. 4. Average speedup (among layers) regarding different number of ants.

IV. Conclusion

A novel algorithm for solving the netlist partitioning and placement problems simultaneously, was introduced. Experimental results regarding different benchmarks and target 3-D FPGAs validate the superiority of our approach as we achieve performance enhancement by 10% on average, as compared to relevant state-of-the-art algorithms. Furthermore our solution improves the design's routability since the subsequent netlist routing requires fewer TSVs (10% on average), while it imposes significant smaller number of bends (on average 30%).

References

[1] C. Ababei, Y. Feng, B. Goplen, H. Mogal, T. Zhang, K. Bazargan, and S. Sapatnekar, "Placement and routing in 3d integrated circuits," *IEEE Design Test Comput.*, vol. 22, no. 6, pp. 520–531, Nov. 2005.

[2] C. Ababei, H. Mogal, and K. Bazargan, "Three-dimensional place and route for fpgas," *IEEE Trans. CAD Integr. Circuits Syst.*, vol. 25, no. 6, pp. 1132–1140, Jun. 2006.

[3] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," in *Proc. Spring Joint Comput. Conf.*, New York, NY, USA, Apr. 18–20, 1967., pp. 483–485, AFIPS'67(Spring), ACM.

[4] K. Boese, A. Kahng, B. McCoy, and G. Robins, "Fidelity and near-optimality of elmore-based routing constructions," in *Proc. IEEE Int. Conf. Comput. Design: VLSI in Comput. Process.*, Oct. 1993, pp. 81–84.

[5] C.-L. Eric Cheng, "Risa: Accurate and efficient placement routability modeling," in *Proc. Int. Conf. Comput.-Aided Design*, 1994, pp. 690–695, ICCAD.

[6] M. Dorigo and T. Stützle, *Ant colony optimization*. Scituate, MA, USA: Bradford Company, 2004.

[7] A. Hahn Pereira and V. Betz, "Cad and routing architecture for interposer-based multi-fpga systems," in *Proc. the Int. Symp. on FPGA*, New York, NY, USA, 2014, pp. 75–84.

[8] A. B. Kahng, J. Lienig, I. L. Markov, and J. Hu, *VLSI Physical design: From graph partitioning to timing closure*, 1st Ed. ed. : Springer Publishing Company, Incorporated, 2011.

[9] G. Katti, M. Stucchi, K. de Meyer, and W. Dehaene, "Electrical modeling and characterization of through silicon via for three-dimensional ics," *IEEE Trans. Electron Devices*, vol. 57, no. 1, pp. 256–262, Jan. 2010.

[10] L. McMurchie and C. Ebeling, "Pathfinder: A negotiation-based performance-driven router for fpgas," in *FPGA, Symp. on Proc. the 3rd Int. ACM*, 1995, pp. 111–117.

[11] V. F. Pavlidis and E. G. Friedman, *Three-dimensional Integrated Circuit Design.*. , USA: Morgan Kaufmann Publishers Inc, 2009.

[12] N. Selvakkumaran and G. Karypis, "Multiobjective hypergraph-partitioning algorithms for cut and maximum subdomain-degree minimization," *IEEE Trans. CAD Integr. Circuits Syst.*, vol. 25, no. 3, pp. 504–517, Mar. 2006.

[13] W. Xu, K. Xu, and X. Xu, "A novel placement algorithm for symmetrical FPGA," in *Proc. 7th Int. Conf. ASIC*, Oct. 2007., pp. 1281–1284.