Optimization of Placement of Dynamic Network-on-chip Cores Using Simulated Annealing

Branislav Hredzak, Oliver Diessel*
School of Electrical Engineering and Telecommunications
* School of Computer Science and Engineering
University of New South Wales
Sydney NSW 2052, Australia
E-mail: b.hredzak@unsw.edu.au, o.diessel@unsw.edu.au

***Abstract*-We derive an objective function which instead of mapping/placing application task graphs in a compact manner onto reconfigurable devices, dilates the mappings as much as the available latencies on critical connections allow. The objective function is then optimized using simulated annealing. The main advantage of the dilated placement of the task graphs is that the unused resources between an application's configured components can be used to provide additional flexibility when the configuration needs to change. We present results of applying the dilated placement to one synthetic case and one real case. The presented results show successful and meaningful graph dilation.**

I. INTRODUCTION

A. Reconfigurable Systems and Networks-on-chip

Reconfigurable systems implemented on field-programmable gate arrays (FPGAs) provide designers with a means by which performance-critical components can be implemented in a more flexible, robust, reusable and useful manner than if they are implemented as application-specific integrated circuit (ASIC) devices.

A significant advantage of reconfigurable systems is that new functional blocks or components can be added and/or old ones can be removed or exchanged, for a range of benefits. Changes may be initiated by a user who requires new functionality to be added, for example, a specific filter to a surveillance camera network [1]; or more filtering may be required to process a signal affected by noise [2]; a different modulation scheme may need to be employed [3]; or some (hardware) components may have failed, necessitating reallocation of functional tasks to the available resources [4]. As suggested, some of these changes may be planned for at design time, but it is desirable to allow for others to occur in an unforeseen manner as repairs to bugs become available, when a new component is developed, when new protocols are invoked, or when a user changes the requirements or demands new functionality.

In recent years, attention has been drawn towards so-called networks-on-chip as a high-performing and flexible interconnection methodology for both static and reconfigurable systems [5-8]. A network-on-chip (NoC) consists of routers that are interconnected via links that convey packetized messages between the routers. Networks are more scalable and flexible than buses and point-to-point connections. Therefore, NoC technology appears to be well suited to providing the communication infrastructure of future dynamically reconfigurable systems-on-chip.

To date, few results ([9-11]) attempt to address the time-varying needs of *dynamic* and *open* application sets. In [9], a regular mesh of routers is proposed. Variously sized cores are placed where they fit and non-deterministic, adaptive routing algorithms are proposed for communicating between the cores, but communication requirements cannot be guaranteed. In [10], a tile-based approach is used to implement a customized network for the configured components. Unfortunately, reconfiguring the tiles disrupts the traffic on links within the tiles. Reconfiguration of the network incurs significant overheads. In [11] an architecture that combines packet-switching and physical circuit-switching is proposed. Reconfiguring the circuit switches can change the network topology, while packet switching can be used to share the circuit-switched connections when the flexibility is needed.

B. Dilated Placement of Dynamic Network-on-chip Cores

Conventional approaches assume that field-programmable gate array (FPGA) resources are in short supply. We explore the potential of utilizing the considerable FPGA resources that are expected to be available in the mid-term future. Hence, instead of mapping an application to the smallest compact region possible, we propose to dilate the mapping as much as the available latency (slack) on critical network connections allows. In a dilated mapping, the free space comprising unused configurable regions, associated network routers and links between an application's configured components can be used to provide additional flexibility when the configuration needs to be changed. For example, when a core is to be added to a dilated design, there are potentially many more placement sites that provide good connectivity to components that have already been configured. Alternatively, it will be less disruptive, and is likely to incur fewer overheads, to move within a small neighbourhood (jog) some of the cores that have already been placed in order to make room for a new one. More importantly, it will be easier to allocate additional routing paths or insert express channels into free regions when additional bandwidth is required or latency needs to be reduced.

In this paper, we derive an objective function that will enable initial, quasi-optimal, dilated placement of application

978-1-61284-972-0/11/$26.00 ©2011 IEEE

task graphs into a given network topology. The objective function addresses the following critical issues: (1) how to spread out the placement of cores without exceeding latency requirements; (2) how to place cores so as to maximize the potential to provide additional bandwidth or reduce latency between cores when required; and (3) how to place cores so as to allow fast and easy relocation, addition or removal of cores.

II. Development of Dilated Mapping Technique

A. Definitions

Definition 1: We define a *communication task graph (CTG)* as a special case of an application characterization graph, as given in [12]. A communication task graph *CTG = CTG(T, C)* is a directed graph, where each vertex $t_i \in T$ represents an IP core or a task that has been mapped to a core and each directed arc $c_{i,j} \in C$ summarizes the total communication or interconnection between vertices t_i and t_j.

While cores for reconfigurable computing applications should be tagged with application-specific information such as size or area, clock frequency, period of activity, and latency, we ignore these details in this paper. However, our work allows the placement of a core to be locked to a particular position to use location-dependent resources.

Each arc $c_{i,j}$ is tagged with application-specific information including the sum of the bandwidth required by all signals from t_i to t_j and the maximum latency that can be tolerated for the connection.

A communication task graph may be specified by the number of cores in the graph and a listing of the connection characteristics as provided in Table I and illustrated in Fig. 1.

Definition 2: We define a *network topology* as a directed graph *NT = NT(R, L)* comprising a set of *routers R* and *links L* connecting them. Each $r_i \in R$ is specified by its *xy* coordinates on an FPGA floorplan and each link $l_{i,j} \in L$ connects a given source router r_i with a unique destination router r_j. In this study, each link $l_{i,j}$ has a maximum bandwidth capacity and a latency associated with it. All links are assumed to have the same maximum bandwidth and the same, constant link latency. The routers are assumed to be capable of switching the traffic when all incoming links are operating at maximum bandwidth. We assume the latency of a router is a constant, irrespective of the traffic volume it switches. Routers are also assumed to have a pair of special *local links* that allow a single core to be directly connected with the router. This pair of links allows the core to inject traffic (data) into the network and to receive data from other cores connected to the network. We define the latency of a network *hop* to comprise the sum of the latency of a single network link and the latency of the router the link inputs to. We neglect the latency, incurred at a core, of injecting data into the network and transferring it from the router to the core. The Manhattan distance between cores and the latency per hop thus characterize the latency between cores.

In this work, we consider the problem of mapping given CTGs to two-dimensional (2D) mesh network topologies. A 2D mesh network has routers located at all *xy* positions in [1..*max_x*, 1..*max_y*] and all routers are connected via bi-directional links to their immediate neighbours to the north, south, east and west, where these neighbours are present.

Definition 3: Mapping a CTG to a network topology involves finding an optimal placement (router positions) for the CTG cores and determining legal, minimal cost routes through network links for the inter-core connections. We have considered XY routing, which is known to be deadlock-free on a mesh [13].

Maximizing performance usually forces the mapping to be as compact as possible. For example, the CTG of Table I might be mapped to a 4 x 4 router mesh as in Fig. 2. In this case, the maximum link bandwidth must be at least 40 MB/s for the mapping to be possible with XY routing. In our work, we are primarily interested in obtaining mappings that facilitate future dynamic modification. Our strategy for achieving such mappings is to dilate the embedding of the cores in the network topology so as to maximize the opportunities (via unused neighbouring routers) for inserting cores into the CTG.

The dilation of the mapping is constrained by the latency bounds on connections and the utilization of link bandwidth capacity.

TABLE I
A SIMPLE 4-CORE CTG EXAMPLE

Connection	Bandwidth (MB/s)	Latency (ns)
$c_{1,2}$	20	10
$c_{2,3}$	30	20
$c_{3,4}$	40	20
$c_{4,1}$	10	10

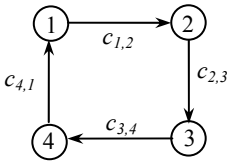

Fig. 1. Example of a simple 4-core CTG.

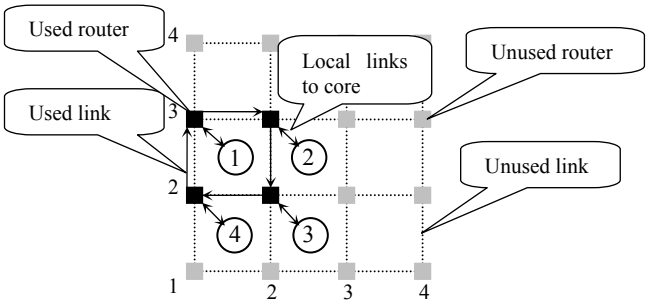

Fig. 2. Mapping of CTG from Table I to 4 x 4 mesh. Router positions are depicted as grey squares at link intersections. Link positions are dotted. Used routers and links are darkened. Positions are numbered along x (horizontal) and y (vertical) axes.

As an example, the CTG of Table I can be mapped in a dilated manner to a 4 x 4 router mesh with a maximum link bandwidth of at least 40 MB/s and a hop latency of 10 ns as in Fig. 3. The difference with the previous mapping of Fig. 2 is that connections $c_{2,3}$ and $c_{3,4}$ have now been mapped to routes that require two hops instead of one as permitted by their given latency constraints. The advantage of doing so is that if a 5th core needs to be added to the CTG of Table I , as listed in Table II, then the mapping of Fig. 2 would have to be modified in order to accommodate the latency requirements for the 5th core. This can incur significant reconfiguration overheads. However, in this case, the 5th core can easily be inserted into the dilated mapping of Fig. 3 at router position (2, 2) as illustrated in Fig. 4. In the absence of a priori knowledge of future CTG requirements, as is the case in open systems, we feel generalized dilation is the best strategy for minimizing the potential cost of modifying the mapping. The dilated mapping problem we have outlined above in some sense inverts the conventional mapping problem that aims to minimize the cost of the mapping by minimizing the sum of the products of the embedded connection lengths and the required connection bandwidths. In our case, the unused spaces (unallocated FPGA resources) between cores, and thus the connection lengths, are to be maximized subject to the latency constraints. We conjecture this problem is harder to optimize than the usual mapping problem since there is more freedom to place the cores at a distance. The approach we have therefore taken to obtain a timely solution to the problem is to apply a simulated annealing heuristic, which is known to produce good results for the conventional mapping problem.

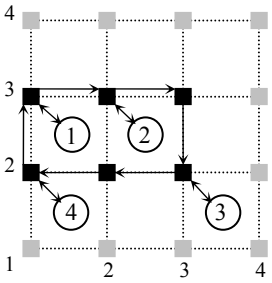

Fig. 3. Dilated mapping of CTG from Table I to 4 x 4 mesh.

TABLE II
A 5TH CORE IS ADDED TO THE CTG EXAMPLE OF TABLE I .

Connection	Bandwidth (MB/s)	Latency (ns)
$c_{1,2}$	20	10
$c_{2,3}$	30	20
$c_{3,4}$	40	20
$c_{4,1}$	10	10
$c_{4,5}$	10	10
$c_{5,2}$	20	10

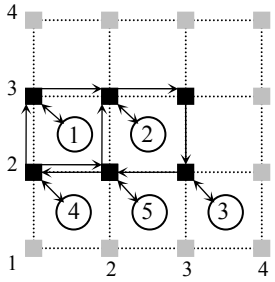

Fig. 4. Addition of 5th core to mapping of Fig. 3.

B. Simulated Annealing Framework

Simulated annealing (SA) [14], has been applied to many combinatorial optimization problems including partitioning, placement and routing, to name a few. The method requires four ingredients for it to be applicable: a concise description of a configuration of the system; a random generator of rearrangements of the elements in a configuration; a quantitative objective function containing the tradeoffs to be made; and an annealing schedule of the temperatures and length of times for which the system is to be evolved.

For the dilated mapping problem, the mapping of CTG cores to mesh routers represents a system configuration. It should be clear, that the placement of cores at specific routers together with our choice of XY routing algorithm determines which routes are used for the inter-core connections.

Rearrangements are generated by randomly choosing two routers, at least one of which should be occupied by a core, and exchanging the cores connected to the chosen routers. The derivation of the quantitative objective function we used will be described in the following subsection.

C. Objective Function

The objective function (1) used in our experiments balances a term promoting the compact placement of communicating cores with another term that favours the dilated placement of cores,

$$cost = \alpha * compaction + (1-\alpha) * dilation\ . \qquad (1)$$

The balance of the compaction and dilation terms can be adjusted by modifying the weight given to the multiplicative constant, α. The goal of the annealing process is to minimize the cost as given by (1).

The compaction term of (1) can be expanded as in (2),

$$compaction = \sum_{connections,\ c} bandwidth_c * hops_c \qquad (2)$$

in which the product of the required bandwidth for each connection between the cores, $bandwidth_c$, and the number of hops (Manhattan distance) over which the connection is formed, $hops_c$, is summed over all connections in the graph. This approach is typically used to minimize the distance between the most heavily communicating cores and thus

commonly leads to a low power/low latency/high clock frequency solution.

The dilation term of (1) is expanded as in (3),

$$dilation = \beta * slack + \gamma * proximity + \delta * utilization\,, \qquad (3)$$

whereby the amount of *slack* on connections, the *proximity* of cores and the *utilization* of link capacity are minimized. These factors can be traded off by adjusting the multiplicative constants, β, γ and δ.

The overall *slack* in (3) of the current placement of the cores is calculated as a sum of slacks for all connections with a given required latency,

$$slack = \sum_{connections,c} slack_c = \sum_{connections,c} (latency_c - hops_c * latency_h)\,, \qquad (4)$$

where the slack of a connection, $slack_c$, is given as a difference in the required latency of a connection, $latency_c$, and the product of the number of hops for the connection, $hops_c$, and the hop latency, $latency_h$.

The proximity term in (3) uniformly distributes (dilates) those core pairs *(u,v)* that are mutually non-communicating or for which no required latency is specified. It is calculated as (5)

$$proximity = \sum_{\substack{horizontal \\ core\ pairs\ (u,v)}} \left(hor\_dist_{u,v} - spacing_{uniform}\right)^2 + + \sum_{\substack{vertical \\ core\ pairs\ (u,v)}} \left(ver\_dist_{u,v} - spacing_{uniform}\right)^2 \qquad (5)$$

where $hor\_dist_{u,v}$ is a horizontal mutual distance between a horizontal pair of cores, $ver\_dist_{u,v}$ is a vertical mutual distance between a vertical pair of cores and $spacing_{uniform}$ is the optimal mutual spacing between a pair of cores in horizontal and vertical directions if they were uniformly distributed across the mesh. The term $spacing_{uniform}$ in (5) is calculated as (6)

$$spacing_{uniform} = ceil\left(\frac{mesh_{dim}}{ceil\sqrt{cores_{number}}}\right), \qquad (6)$$

where $mesh_{dim}$ is the dimension of the square mesh, $cores_{number}$ is the total number of cores to be mapped on the mesh and *ceil* rounds the expression to the nearest integer towards infinity.

Placements that minimize the link utilization are preferred in order to provide spare capacity for cores to be added to the communication graph over time. This goal is captured by (7)

$$utilization = \sum_{shared\ links,\ s} connections_s * bandwidth_s\,, \qquad (7)$$

where $connections_s$ is the number of connections sharing each used link and $bandwidth_s$ is the sum of the bandwidths of the connections sharing the link, i.e., the total bandwidth of the link. Rather than apply (7) to the CTG embedding in the network, which penalizes longer (dilated) shared paths more than shorter ones, we construct a graph of the embedded CTG connections, thereby inserting pseudo-vertices into the CTG where signal paths meet at shared routers. Equation (7) is then applied to this graph of the embedding with each shared connection being counted just once.

Constraints (8) and (9)

$$bandwidth_{\max} - bandwidth_s \geq 0, \forall\ shared\ links,\ s \qquad (8)$$

$$latency_c - hops_c * latency_h \geq 0, \forall\ connections,\ c \qquad (9)$$

also need to be observed. Constraint (8) requires that the total bandwidth allocated to a network link, $bandwidth_s$, must be no greater than the maximum link bandwidth capacity $bandwidth_{max}$. Constraint (9) ensures that no connection exceeds the required latency of the connection.

III. RESULTS

This section presents simulation results to allow our dilated mapping technique to be evaluated for one synthetic case, Case I, and one real application case, Case II.

As described in the previous section, the balance of the compaction and dilation terms can be adjusted by modifying the weight given to the multiplicative constant α in (1). In the results presented here, α was changed in a discrete manner, i.e., initially $\alpha = 1$, corresponding to compaction only. Once compaction converged, α was changed to $\alpha = 0$, which corresponded to dilation only. In this way, the initial random allocation was first compacted and then dilated. By running the compaction stage first, the SA algorithm initially obtains a compacted solution in which the maximum link bandwidths are unlikely to be exceeded. This intermediate solution also locates the communicating cores as closely as possible, resulting in the maximum possible slack between communicating cores. When the dilation phase is started from the compacted solution, the probability of starting with a solution that does not adhere to the given constraints is smaller and the convergence of the SA algorithm is significantly improved.

Experimentation with a number of graphs indicated that $\beta = 1$, $\gamma = 0.2$ and $\delta = 0.04$ resulted in a reasonable balance in the contributions to (3) from the slack, the proximity and the utilization. If all contributions are of equal importance then the values of the multiplicative constants β, γ and δ are chosen intuitively so that the magnitudes of the terms $\beta * slack$, $\gamma * proximity$, and $\delta * utilization$ are approximately equal during the optimization.

In our experiments, the SA was started with an initial, random placement of the cores. Following standard SA practice, the current placement was perturbed at the given temperature and the change in configuration was accepted if the perturbation resulted in a lower cost configuration, or with exponentially decreasing likelihood if it yielded an increase in cost [14]. In both the compaction and dilation phases we decreased the temperature when at least 50 successive rejected perturbations preceded a successful one. The algorithm was terminated when $T_{min} = 0$ was reached, or when 10,000 successive perturbations were rejected.

A. Case I

In Case I, listed in Table III, we dilated 8 cores into a 9 x 9 mesh. Fig. 5 shows the initial placement of the 8 cores. Fig. 6 shows the detail of the compacted solution and Fig. 7 shows the dilated solution. The maximum allowed link bandwidth was set to 20. The dilated placement has zero slack for all connections, the utilization is minimized, and the distances between non-communicating cores were optimized. Comparing placements shown in Fig. 6 and Fig. 7 it can be appreciated that new cores can be inserted more readily into the dilated placement in order to satisfy the latency requirements of new cores while at the same time not exceeding any given link bandwidth constraints.

B. Case II

In Case II, we dilated a high performance, power constrained 4G wireless modem application described in [15, 16]. The application has 34 nodes (cores). In our results, cores 1 to 16 correspond to nodes M0 to M15 and cores 17 to 34 correspond to nodes S0 to S17 in [16], and our single hop latency is assumed to be 50 ns. A summary of bandwidth and latency requirements can be found in [16]. We dilated the 34 cores into a 16 x 16 mesh. The initial placement is not shown here due to space constraints. Fig. 8 shows the detail of the compacted solution and Fig. 9 shows the dilated solution. As in Case 1, it can be appreciated that new cores can be inserted more readily into the dilated placement in order to satisfy the potential latency requirements of new cores while not exceeding link bandwidth limits.

TABLE III
INTERCONNECTION MATRIX FOR CASE I.
(Connections listed as *Required Bandwidth/Connection Latency in Hops*; cells representing null connections left blank).

Core	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
t_1		10/2			10/4			
t_2	10/2		10/2			10/4		
t_3		10/2		10/2			10/4	
t_4			10/2					10/4
t_5	10/4					10/2		
t_6		10/4			10/2		10/2	
t_7			10/4			10/2		10/2
t_8				10/4			10/2	

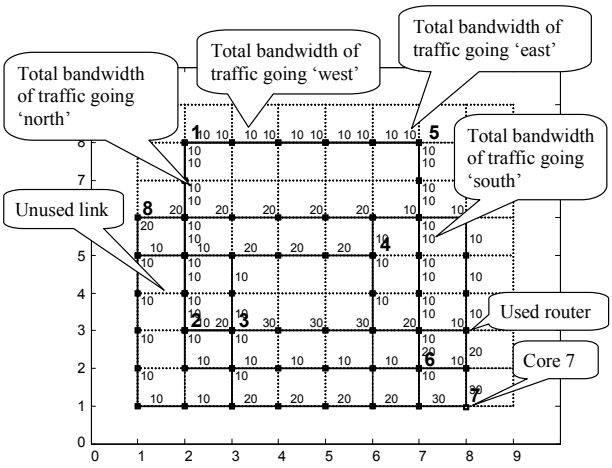

Fig. 5. Initial (random) placement of Case I into a 9 x 9 mesh.

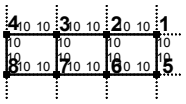

Fig. 6. Placement of Case I after compaction.

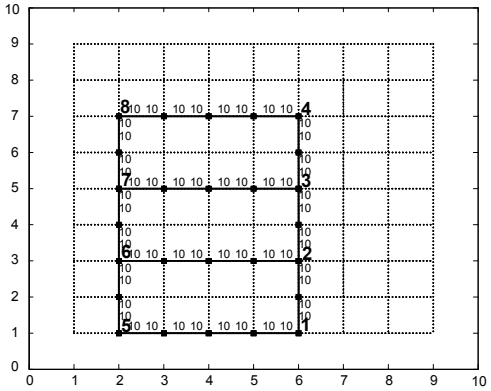

Fig. 7. Placement of Case I into a 9 x 9 mesh after dilation.

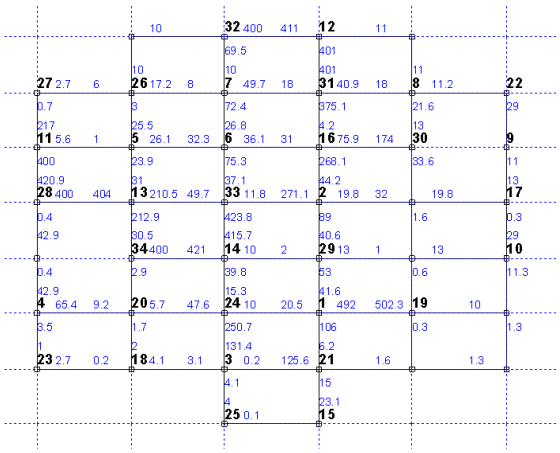

Fig. 8. Placement of Case II after compaction.

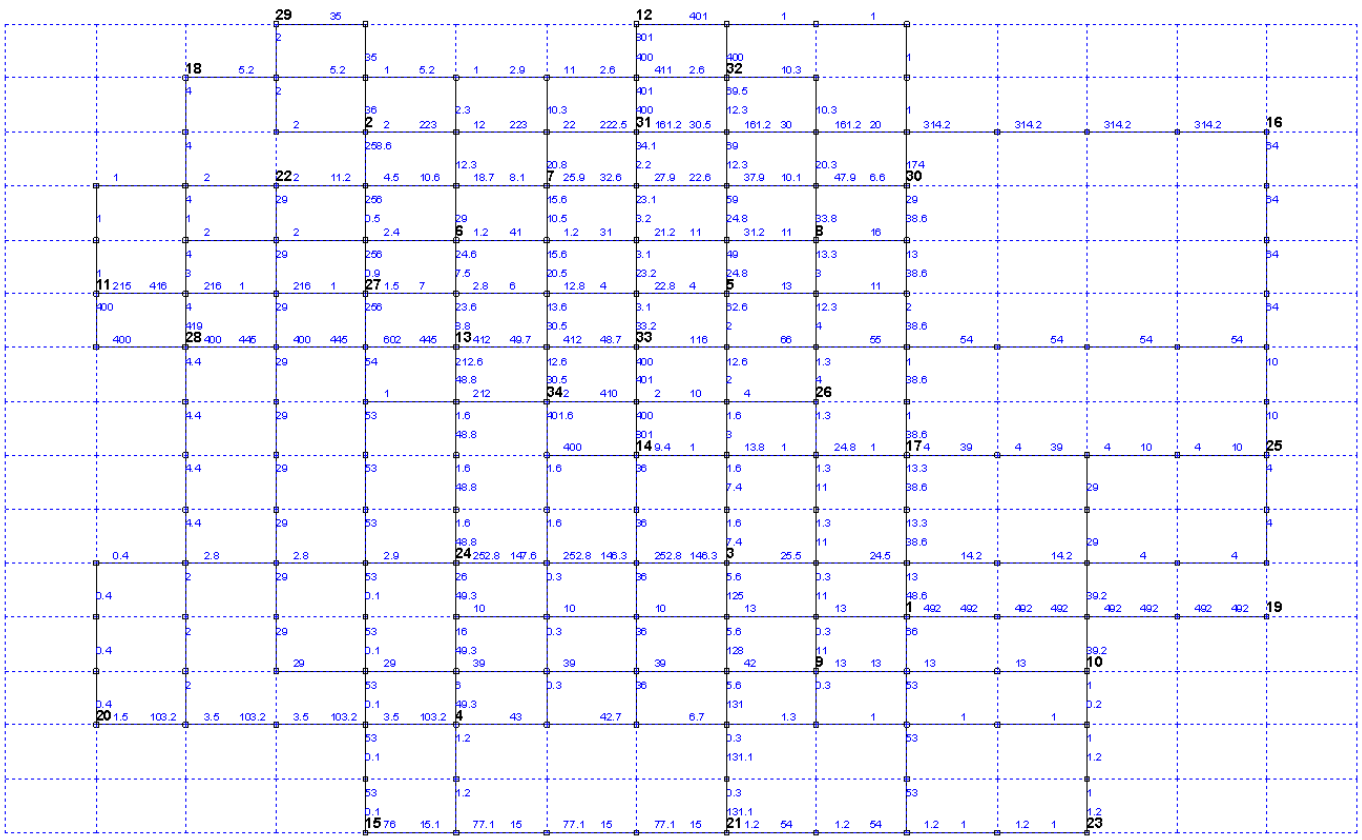

Fig. 9. Placement of Case II into a 16 x 16 mesh after dilation.

IV. CONCLUSION

In this paper we detailed our motivation for exploring the benefits of dilating communication task graphs, derived a simulated annealing approach to dilating the initial placement of such graphs, and presented results of applying the algorithm to one synthetic case and one real case. The method results in successful and meaningful graph dilation. We plan to examine the effect of varying the parameters of the simulated annealing and to explore approaches to maintaining a dilated placement at runtime.

REFERENCES

[1] V. Tadigotla and S. Commuri, "Dynamic image filter selection using partially reconfigurable FPGAs for imaging operations," presented at the Proceedings of the 5th WSEAS International Conference on Circuits, Systems, Electronics, Control and Signal Processing, Dallas, Texas, 2006.

[2] R. Tessier, *et al.*, "A reconfigurable, power-efficient adaptive Viterbi decoder," *IEEE Transactions on VLSI Systems,* vol. 13, pp. 484-488, 2005.

[3] G. J. M. Smit, *et al.*, "Dynamic Reconfiguration in Mobile Systems," presented at the Proceedings of the Reconfigurable Computing Is Going Mainstream, 12th International Conference on Field-Programmable Logic and Applications, 2002.

[4] P. Zipf, "Applying dynamic reconfiguration for fault tolerance in fine-grained logic arrays," *IEEE Trans. Very Large Scale Integr. Syst.,* vol. 16, pp. 134-143, 2008.

[5] D. Micheli and L. Benini, "Networks on Chips: Technology and Tools," ed: Morgan Kaufmann, 2006.

[6] A. Jantsch and H. Tenhunen, *Networks-on-Chip*: Kluwer, 2003.

[7] W. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," presented at the Proc. Des. Autom., 2001.

[8] M. B. Taylor, *et al.*, "Evaluation of the Raw Microprocessor: An Exposed-Wire-Delay Architecture for ILP and Streams," presented at the International Symposium on Computer Architecture (ISCA), 2004.

[9] C. Bobda, *et al.*, "A Dynamic NoC Approach for Communication in Reconfigurable Devices " presented at the International Conference on Field Programmable Logic and Applications, 2004.

[10] T. Pionteck, *et al.*, "Applying Partial Reconfiguration to Networks-On-Chips," presented at the International Conference on Field Programmable Logic and Applications (FPL), 2006.

[11] M. B. Stensgaard and J. Spars, "ReNoC: A Network-on-Chip Architecture with Reconfigurable Topology," presented at the International Symposium on Networks-on-Chip (NOCS), 2008.

[12] R. Marculescu, *et al.*, "Outstanding research problems in NoC design: system, microarchitecture, and circuit perspectives," *Trans. Comp.-Aided Des. Integ. Cir. Sys.,* vol. 28, pp. 3-21, 2009.

[13] L. M. Ni and P. K. McKinley, "A Survey of Wormhole Routing Techniques in Direct Networks," *Computer,* vol. 26, pp. 62-76, 1993.

[14] S. Kirkpatrick, *et al.*, "Optimization by Simulated Annealing," *Science,* vol. 220, pp. 671-680, May 13, 1983 1983.

[15] R. Beraha, *et al.*, "The design of a latency constrained, power optimized NoC for a 4G Soc," presented at the ACM/IEEE International Symposium on Networks-on-Chip 2009.

[16] I. h. Walter, *et al.*, "The design of a latency constrained, power optimized NoC for a 4G Soc," Technion-Israel Institute of Technology CCIT Report #724, 2009.