Physical Planning for On-Chip Multiprocessor Networks and Switch Fabrics

Terry Tao Ye
Computer Systems Lab
Stanford University
taoye@stanford.edu

Giovanni De Micheli
Computer Systems Lab
Stanford University
nanni@stanford.edu

Abstract

On-chip implementation of multiprocessor systems requires the planarization of the interconnect network onto the silicon floorplan. Manual floorplanning approaches will become increasingly more difficult and ineffective as multiprocessor complexity increases. Compared with traditional ASIC architectures, multiprocessors have homogeneous processing elements and regular network topologies. Therefore, traditional ASIC floorplanning methodologies based on macro placement are not effective in this domain. In this paper, we propose an automated physical planning tool, called REGULAY*, that can generate floorplans for different topologies under different design constraints. Compared with traditional floorplanning approaches,* REGULAY *shows significant advantages in reducing the total interconnect wire-length while preserving the regularity and hierarchy of the network topology.*

1 Introduction

Multiprocessor Systems on Chips (MPSoCs) combine the advantages of parallel computing of multiprocessors with single chip integration of SoCs. MPSoCs are employed in embedded systems that require high performance data processing capabilities. Examples include *network processors* (NPs), *parallel multimedia processors* (PMPs) and other *application specific array processors* (ASAPs).

Advances in VLSI process technology allow designers to further increase system-level integration onto a single chip. Future MPSoCs are likely to consist of hundreds, or even thousands, of *processing elements* (PEs). These PEs will communicate with each other independently and concurrently [1]. Traditional shared-medium communication architectures (e.g., buses) will no longer be able to support the massive data traffic on this scale. Hence, future MPSoCs need to adopt a dedicated on-chip interconnect network that can provide reliable and scalable communication [2].

Designing the on-chip network will become a major task for future MPSoCs. A large fraction of the timing delay is spent on the signal propagation on the interconnect, and a significant amount of energy is also dissipated charging and discharging the load capacitance on the wires [3]. Therefore, an optimized interconnect network floorplan will be of great importance to MPSoC performance and energy consumption.

With the ever-increasing complexity of MPSoC integration, manual floorplanning of the processing elements and switches will become even more time consuming and inefficient. Automated methods are needed for large-scale MPSoC designs. Unlike traditional floorplanning that deals with the circuit macro block placement and wire routing [4], MPSoC floorplanning needs to solve the problems from a different perspective, as illustrated in Fig. 1. Namely:

1. *Folding and planarization* – MPSoC network topologies are multi-dimensional. MPSoC planar layout requires that PE blocks are tiled and abutted on the floorplan in a two-dimensional tile array [1]. The planarization process is also constrained by the pre-defined aspect ratio and row/column numbers of the tile array.
2. *Regularity and hierarchy* – MPSoC network topologies are often regular and hierarchical. The planarization of the network is not only a simple packing process: it has to preserve the regularity and hierarchy on the floorplan.

Proceedings of the Application-Specific Systems, Architectures, and Processors (ASAP'03)
ISBN0-7695-1992-X/03 $17.00 © 2003 IEEE

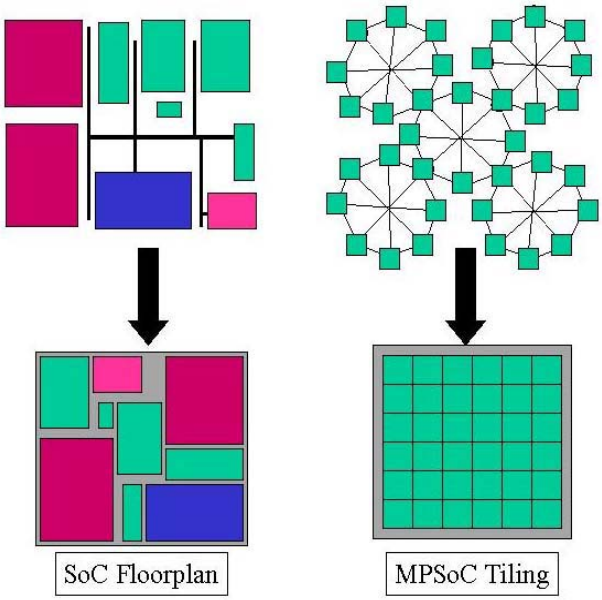

Figure 1. MPSoC Tiling is different from traditional floorplan

3. *Critical path and total wire-length* – Interconnect delays and power consumption are the two critical issues in MPSoC network designs. On one hand, inter-node communication latencies are dominated by the wire propagation delays. Therefore, the wire-length of the timing-critical links needs to be minimized. On the other hand, interconnect wires are the main contributors of the total system power consumption. Reducing the total wire-length helps reducing the power dissipated on the interconnect.

Prior network graph planarization approaches either targeted only some specific topologies, or they were not flexible enough to adapt to many of the floorplan constraints imposed by the silicon implementation [5] [6]. Therefore, those approaches are not suitable for an automated design flow.

In this paper, we describe a floorplanning tool called *REGULAY* that can automatically place regularly-configured MPSoC node processors as well as switch fabrics onto a user-defined tile floorplan. Given the MPSoC network topology and the physical dimension of the network nodes as inputs, along with the floorplan specification (locations of the I/O tiles, number of rows and columns of the tiles), *REGULAY* can create a floorplan that best satisfies different design constraints.

The paper is organized as follows: Section 2 will first describe some of the popular topologies used in MPSoC networks. Based on the characteristics of these networks, Section 3 generalizes and formulates the MPSoC floorplanning problem. Our proposed floorplanning method consists of two steps: regularity extraction (Section 4), and legalization (Section 5). A couple of different network topologies are tested by *REGULAY* in Section 6. The resulting floorplans are much more compact as compared with other general ASIC floorplanning tools.

2 MPSoC Network Topologies

Because of different performance requirements and cost metrics, many different multiprocessor network topologies are designed for specific applications. MPSoC networks can be categorized as *direct networks* and *indirect networks* [7]. In direct network MPSoCs, node processors are connected directly with each other by the network. Each node performs dataflow routing as well as arbitration. In indirect network MPSoCs, node processors are connected by one (or more) intermediate node switches. The switching nodes perform the routing and arbitration functions. Therefore, indirect networks are also often referred to as *multistage interconnect networks* (MIN). In this paper, to avoid confusion, we call the intermediate switching nodes in indirect networks *switch fabrics*, and simply refer to both node processors and node switches as "nodes".

Direct networks and indirect networks can have different topologies [7]. It is not the objective of this paper to discuss the functionalities and performance metrics of these different networks.

Proceedings of the Application-Specific Systems, Architectures, and Processors (ASAP'03)
ISBN0-7695-1992-X/03 $17.00 © 2003 IEEE

Rather, we are going to give only a brief description of some of the popular network topologies. We will use these topologies as examples to formulate the MPSoC floorplanning problems in later sections.

2.1 Direct Network Topologies

2.1.1 Orthogonal Topology

Nodes in orthogonal networks are connected in k-ary n-dimensional mesh (k-ary n-mesh) or k-ary n-dimensional torus (k-ary n-cube) formations, as shown in Fig. 2. Because of the simple connection and easy routing provided by adjacency, mesh and torus networks are widely used in parallel computing platforms [8]. Orthogonal networks are highly regular. Therefore, the interconnect length between nodes is expected to be uniform to ensure the performance uniformity of the node processors.

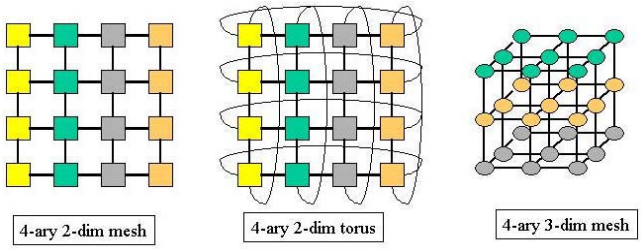

Figure 2. Mesh and torus networks

2.1.2 Cube-Connected-Cycles Topology

The *cube-connected-cycles* (CCC) topology is proposed as an alternative to orthogonal topologies to reduce the degree of each node [9], as shown in Fig. 3a. Each node has 3 degrees of connectivity as compared to 2^n degrees in mesh and torus networks. CCC networks have a hierarchical structure: the three nodes at each corner of the cube form a local ring.

2.1.3 Octagon Topology

The Octagon network (Fig. 3b) was proposed by [10] as an on-chip communication architecture for network processors. In this architecture, eight processors are connected by an octagonal ring and three diameters. The delays between any two node processors are no more than two hops (through one intermediate node) within the local ring. The Octagon network is scalable. If one node processor is used as the bridge node, more Octagon networks can be cascaded together, as shown in Fig. 3b.

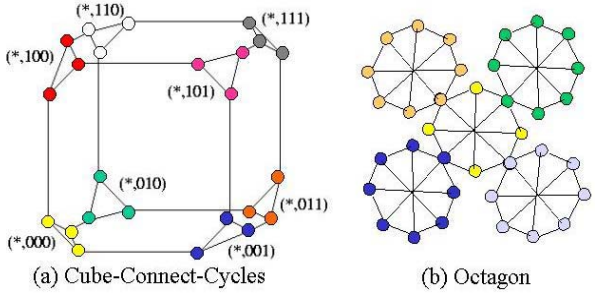

Figure 3. Octagon networks and cube-connected-cycles networks

2.2 Indirect Network Topologies

2.2.1 Butterfly Topology

The Butterfly network (Fig. 4) is an indirect network architecture. Inside the butterfly fabrics, each source-destination route uses a dedicated datapath. The delays between any two node proces-

sors are the same, and the delay is determined by the number of intermediate stages on the switch fabrics.

Butterfly topology has many different isomorphic variations, such as *Omega Network*, *Benes Networks*, etc. Because they have similar topologies, these networks can be tiled with similar floorplans.

2.2.2 Fat-tree Topology

Unlike the Butterfly network, a *fat-tree* network provides multiple datapaths from source node to destination node. As shown in Fig. 4, the fat-tree network can be regarded as an expanded n-ary tree network with multiple root nodes. The network delays are dependent on the depth of the tree. SPIN network [11] is one design example that uses 4-ary fat-tree topology for the MPSoC on-chip communication.

2.3 MPSoC Network Floorplan

Although quite different in their topologies, the above network examples show some important aspects in common: *regularity and hierarchy*. Regular and hierarchical topologies help to achieve the computation parallelism and performance uniformity. Therefore, preserving the regularity and hierarchy formations in the silicon floorplan is critical in MPSoC implementation.

Furthermore, as described in Section 1, on-chip interconnect delays and power consumption add additional requirements in MPSoC floorplan designs. To reduce wiring delays, MPSoC floorplans need to limit the wire-length of the critical links (links that are timing sensitive). To reduce the interconnect energy dissipation, the total network wire-length needs to be minimized.

3 Problem Formulation

In an MPSoC floorplan, each node processor or node switch is placed as a dedicated hard block tile. For example, in direct networks, as in the case of the Octagon network design, the node processors can be tiled in a two-dimensional array, e.g., a 6×6 array in Fig. 5. In indirect networks, as in the case of the Butterfly network, the tiling of the switch fabrics will be constrained by the locations of the node processors, as shown in Fig. 5.

Formally, we are given a source network S connecting a set of modules M, $M = \{m_i, i = 1, 2, 3, ..., p\}$, and a target two-dimensional tile array T with $col \times row$ tiles. Since modules cannot overlap, we assume $p \leq col \times row$. Each net in N connects two (or more) modules in M, and has a weighting factor. For example, net $n_{ij,...,k} \in N$ connects modules in $m_i, m_j, ..., m_k$ and has weight $w_{ij,...,k}$.

Different network topologies and application requirements set different constraints on MPSoC floorplanning problems. To be more specific, we summarize the constraints that are relevant for MPSoC floorplanning:

1) *Regularity constraints* – As shown in Section 2, MPSoC placement should preserve the regularity of the original network topology.

2) *Hierarchy constraints* – MPSoC networks may have hierarchical topologies (clusters), e.g., a cascaded Octagon network consists of multiple local rings. The placement should also preserve this hierarchical clusters.

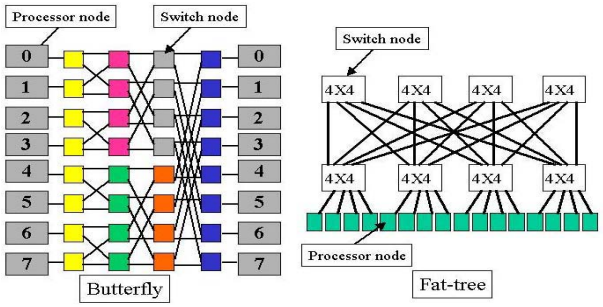

Figure 4. Butterfly and Fat-tree network switch fabrics

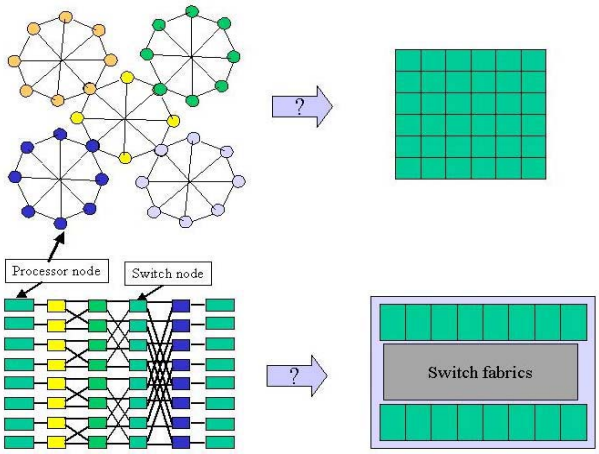

Figure 5. Constraints of floorplan tiling

3) *I/O constraints* – An MPSoC is implemented on a single chip. Some node processors (or node switches, in the case of switch fabrics) serve as I/O nodes; therefore, they need to be placed at the peripherals of the floorplan. An MPSoC floorplan needs to accommodate those nodes at their proper locations.

4) *Aspect-ratio constraints* – Chip die size is limited by the silicon area and aspect ratio. Therefore, node processor blocks and node switch blocks need to be packed into a two-dimensional array with predefined numbers of rows and columns.

5) *Critical-path constraints* – The links between some node processors may be the critical paths, e.g. the center ring in the cascaded Octagon network. Therefore, the nodes connected by the critical paths need to be placed closer to each other.

6) *Total net-length constraints* – Reducing the total net length will achieve shorter interconnect delays with lower power consumption.

The floorplanning problem is to determine a mapping from S to T, such that the constraints are met and the overall wiring length is minimal. Such a problem is computationally intractable, and has been the object of extensive investigation in the ASIC domain. We propose a two-step heuristic approach that takes into account the special properties of MPSoC topologies.

The proposed approach consists of two steps: 1) regularity extraction and determination of tentative locations, and 2) legalization. The first step generates the relative locations of the modules based on the regularity and hierarchical information extracted from the network topology. This is achieved by forming the quadratic matrix and using specific weights on the nets. If some modules have pre-fixed locations in T, these locations are used as placement constraints. The total weighted net length is used as objective function. The second step will pack the modules onto the floorplan constrained by the I/O locations and aspect ratio.

4 Regularity Extraction

We solve the regularity extraction and initial location problems by forming the positions of the modules into a quadratic objective function. The summation of the quadratic wire-length between all the nodes can be calculated through matrix operations that preserve the topological regularity information, i.e., if the nodes in the original topology are symmetrical, the corresponding elements in the matrix should be symmetrical as well. Furthermore, all subsequent matrix operations (e.g., transposition, vector multiplication, etc.) will preserve the regularity formation. Therefore, through optimizing this objective function, the total quadratic wire length is minimized, and the regularity information will be preserved in the optimization process.

4.1 Forming the Objective Function

Giving a set of modules M, $M = \{m_i, i = 1, 2, 3, ..., p\}$,with locations on $(x_1, y_1), (x_2, y_2), ..., (x_p, y_p)$, the total weighted square wire length objective function can be expressed as

$$\Phi(x,y) = \sum_{i,j=1}^{p} w_{ij}((x_i - x_j)^2 + (y_i - y_j)^2) = \mathbf{x^T Q x} + \mathbf{y^T Q y} \quad (1)$$

where $\mathbf{x}$ and $\mathbf{y}$ are the location vectors for the modules on X and Y dimensions. $\mathbf{Q}$ is the quadratic matrix for the weight factor w_{ij}. $\mathbf{Q}$ is generated in the following ways: 1) w_{ij} is 0 if there is no connection between modules m_i and m_j. 2) When modules m_i and m_j are connected, the value of w_{ij} should be the weighting factor of the net between m_i and m_j. 3) The diagonal elements w_{11}, w_{22}, w_{nn}, etc. of the quadratic matrix are the negative sums of all the elements on the same row.

$$w_{ij} = \begin{cases} 0 & i \neq j \text{ and no interconnect between } m_i \text{ and } m_j \\ \textit{weighting\_factor(i,j)} & i \neq j \text{ and } m_i, m_j \text{ are connected} \\ -\sum_{k=1,k\neq i}^{p} w_{ik} & i = j \text{ (The sum of } w_{ik} \text{ in the row i)} \end{cases}$$

As mentioned in Section 3, MPSoC floorplanning is sometimes constrained by pre-defined I/O locations. To address these two different scenarios (with and without I/O constraints), we develop two approaches, as described in the following sections.

4.2 Without I/O Constraints

Eq. 1 shows that the $\mathbf{x}$ and $\mathbf{y}$ location vectors are independent of each other; therefore, we can optimize the positions on the X and Y coordinates separately.

4.2.1 X-dimension Optimization

Since there is no I/O or boundary condition, we need to further normalize the objective function on the X-dimension by using the inner product of the $\mathbf{x}$ vector $\mathbf{x^T x}$. We adopt the quadratic optimization theorem [12] with a different formulation:

Theorem 1 *A quadratic $p \times p$ matrix Q has p different non-negative eigenvalues, $\lambda_1 < \lambda_2 <, ... < \lambda_p$. The corresponding ortho-normal eigenvectors are $e_1, e_2, ...e_p$, where e_1 is the eigenvector of eigenvalue λ_1, and e_2 is the eigenvector of eigenvalue λ_2, etc.*

For any vector $\mathbf{x}$, we have the following relationship:

$$\Phi(x) = \frac{\mathbf{x^T Q x}}{\mathbf{x^T x}} \geq \frac{\mathbf{E_1^T Q E_1}}{\mathbf{E_1^T E_1}} = \lambda_1 \quad (2)$$

where $E_1 = c\mathbf{e_1}$ and c is a constant.

Therefore, we need to use only the eigenvector e_1, which corresponds to the smallest eigenvalue λ_1, as the location vector for the X coordinate to achieve the minimum value of the X-dimension objective function.

4.2.2 Y-Dimension Optimization

The Y-dimension location vector has to be orthogonal to the location vector on the X-dimension, otherwise, the modules will be placed in a diagonal line on the floorplan. We have already used e_1 as the X-dimension location vector, so for the Y-dimension vector, we have $\mathbf{y}^T e_1 = 0$.

Again, we will adopt the theorem presented in [12]. The theorem is re-written as follows:

Theorem 2 *If $\mathbf{y}^T e_1 = 0$, the normalized objective function on the Y-dimension has the following relationship:*

$$\Phi(y) = \frac{\mathbf{y^T Q y}}{\mathbf{y^T y}} \geq \frac{\mathbf{E_2^T Q E_2}}{\mathbf{E_2^T E_2}} = \lambda_2 \quad (3)$$

Using the two theorems, we can see that when there are no I/O constraints, we will use the eigenvector e_1 and e_2, which correspond to the first two smallest eigenvalues of the quadratic function $\mathbf{Q}$ as the location vectors on the X and Y dimensions (the X and Y location vectors can be exchanged).

Fig. 6 shows the screen-shot of the initial node locations of the five-ring Octagon network without I/O constraints. The locations on the X-Y plane are obtained directly from the first two eigenvectors of $\mathbf{Q}$. From the locations of the nodes, we can see that not only is the regularity formation of the nodes preserved, the hierarchical clustering of the cascaded Octagon rings is shown as well.

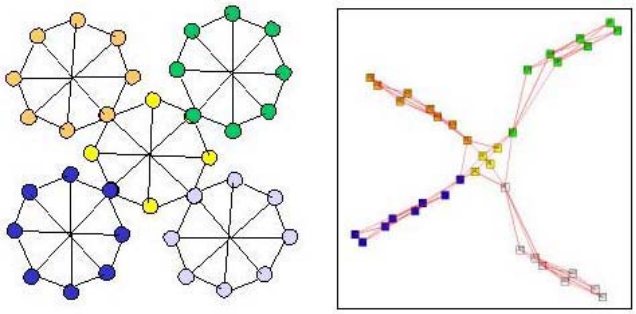

Figure 6. Initial eigenvector locations of 5-ring Octagon network without I/O constraints

Fig. 7 shows the initial locations of the cube-connect-cycles obtained from the eigenvectors of the quadratic matrix. Again, the formation of the nodes preserves the regularity as well as the hierarchy of the original topology.

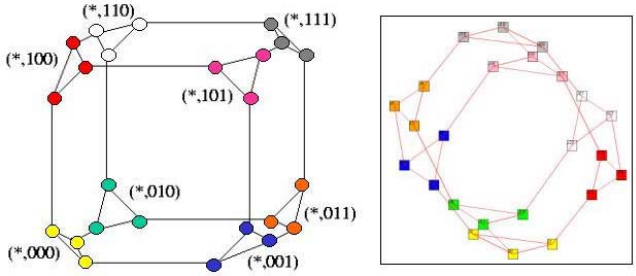

Figure 7. Initial eigenvector locations of cube-connected-cycles without I/O constraints

4.3 With I/O Constraints

If the positions of some modules are pre-fixed by the I/O constraints, we denote these modules as $M_f \subset M$, and their corresponding location vector is denoted as $\mathbf{x}_f \subset \mathbf{x}$. Similarly, the locations of all the movable modules are denoted as vector $\mathbf{x}_c \subset \mathbf{x}$. The objective function can then be re-written as:

$$\Phi(x) = (\mathbf{x_c} \quad \mathbf{x_f}) \begin{pmatrix} Q_{cc} & Q_{cf} \\ Q_{fc} & Q_{ff} \end{pmatrix} (\mathbf{x_c} \quad \mathbf{x_f})^{\mathbf{T}} \tag{4}$$

Solving the zeros of the derivative of the objective function, we have

$$\mathbf{Q_{cc}x_c} = -\mathbf{Q_{cf}x_f} \tag{5}$$

The vector solved from the equation $\mathbf{x_c}$ is used as the location vector on the X-dimension. Because the X-dimension and the Y-dimension are independent, we can perform the same operation on the Y-dimension with the corresponding constraints.

Fig. 8 shows the initial locations of the five-ring cascaded Octagon network with I/O constraints. The four bridge I/O nodes in the center Octagon ring are used as I/O nodes and placed at the four corners of the floorplan. The four I/O nodes are used as the fixed locations in the quadratic equation Eq. 4. Under the I/O constraints, the Octagon network shows a different formation than that without I/Os (Fig. 6). Nevertheless, the regularity as well as the hierarchical formation of the network is still preserved.

ISBN0-7695-1992-X/03 $17.00 © 2003 IEEE

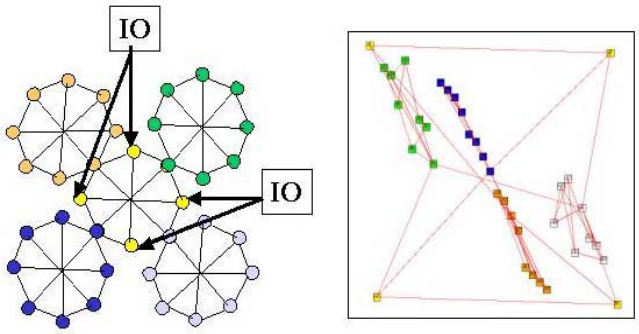

Figure 8. Initial locations of 5-ring Octagon network with I/Os on the corners

Fig. 9 shows the initial locations of the 2-ary 3-fly Butterfly switch fabrics. There are 8 node processors and 32 node switches in the network. The node processors are numbered from 0 to 7, as shown in Fig. 9. One node processor connects to two node switches, serving as input switch and output switch respectively. On the floorplan, we place the node processors 0, 1, 2, 3 on left side of the floorplan, while the node processors 4, 5, 6, 7 on the right side. This arrangement of node processors imposes I/O constraints on the Butterfly switch fabrics, because the switching nodes that serve as input and output have to be placed next to the corresponding node processors. The regularity formation of the switch fabrics is still preserved under these I/O constraints, as shown in the figure.

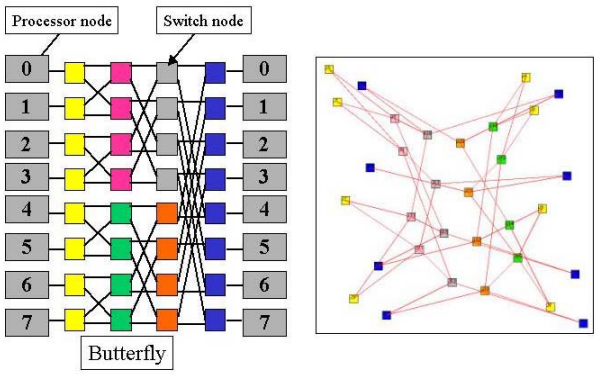

Figure 9. Initial locations of Butterfly network with I/O constraints

5 Legalization

The node positions solved from the quadratic objective function optimization are real-valued numbers. They cannot be used directly in the tiling placement, where the locations on rows and columns are quantized. Instead, we will use these values as relative locations and further legalize (quantize) the node positions.

The legalization procedure also consists of two steps: 1) sorting, where the nodes are ordered by the X and Y coordinates, and 2) packing, where the node blocks are assigned to the corresponding tiles (row and column positions).

In the sorting step, as shown in Fig. 10, the nodes are first sorted according to their X coordinates and evenly partitioned into several bins. The number of bins is equal to the number of columns. Then the nodes in each bin are further sorted according to their Y coordinates. After this step, the nodes are ordered in both the X and Y coordinates. The packing step will assign the nodes into the corresponding tiles in the $col \times row$ tile floorplan. The legalization procedure involves two linear sorting operations, which can be implemented with any existing sorting algorithms.

Fig. 11 shows the legalization results. Fig. 11a is the legalized floorplan from Fig. 6 and Fig. 11b is legalized from Fig. 8. Both floorplans preserve the regularity and hierarchy formations of the original topologies. Furthermore, the proposed floorplan also achieves a minimal total interconnect

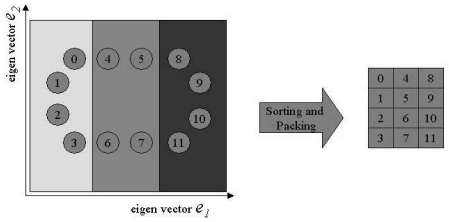

Figure 10. Legalization of the node locations by sorting and packing

wire-length compared with traditional floorplanning approaches. We will show this comparison in details through several experiments.

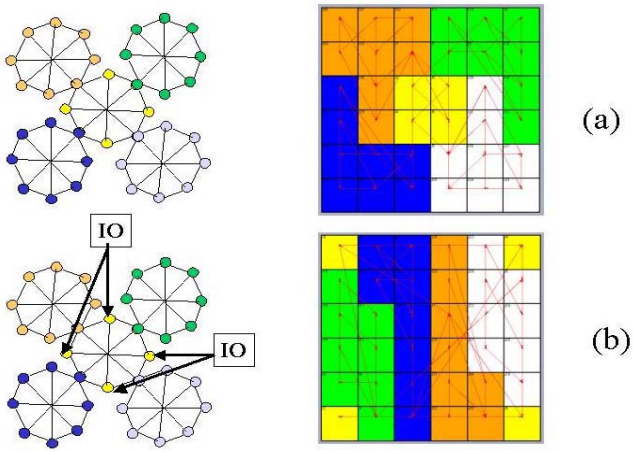

Figure 11. Legalized floorplan of Octagon networks with and without I/O constraints

6 Experiments

We have built a tool called *REGULAY* that implements the proposed floorplanning method. *REGULAY* is written in C++ with GUI written in Tcl/Tk. To the best of the authors' knowledge, there were no prior tools that target specifically on the MPSoC network floorplanning applications. Therefore, we compare the resulting floorplan and the total interconnect wire-length with the results obtained from ASIC floorplanning approaches. We choose UCLA MCM floorplanner [13] for this comparison. MCM is an open-source non-commercial tool that was originally designed to solve general ASIC floorplanning problems. Nevertheless, we perform this comparison to show that our method is particularly advantageous for MPSoC floorplans.

Fig. 12 shows the floorplan result of the cube-connected-cycles by *REGULAY*. There are total 24 nodes and 36 nets in the topology. For a better visualization of the regularity and hierarchy of the resulting floorplan, we assign different colors to different groups of nodes. There are no I/O constraints for the floorplan. From the floorplan formation, we can see that regularity information of the topology is well preserved by *REGULAY*.

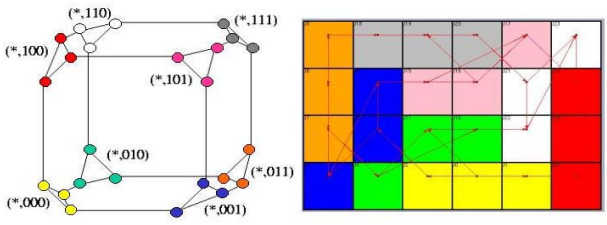

Figure 12. Floorplan of Cube-Connected-Cycles network

The 4-ary 3-mesh network floorplan result is shown in Fig. 13. There are 64 nodes and 144 interconnects in this network, and the floorplan is an 8×8 tile array. Both the original 4-ary 3-mesh

topology and the resulting floorplan are shown in the figure. Again, different groups of nodes are assigned different colors for a better visualization. As shown from the figure, *REGULAY* creates a satisfying results for this topology. All the nodes are placed into a regular and clustered formation on the floorplan. The locations of yellow nodes and blue nodes are symmetrical to each other, and the green nodes and white nodes are symmetrical too. This is because that yellow and blue nodes are "sandwiched" between green and white nodes in the original topology.

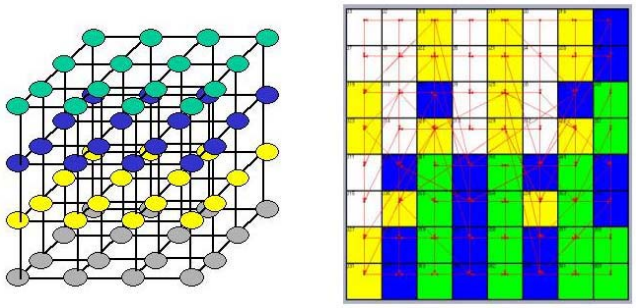

Figure 13. Floorplan of 4-ary 3-mesh network

Fig. 14 shows the floorplan of 4-ary 3-cube torus network. There are total 64 node processors and 192 nets in this network, and they are also mapped into the same 8×8 tile floorplan. For a clearer view of the original network topology, we do not show all the 192 nets in the figure. No I/O locations are constrained. Compared with the 4-ary 3-mesh network, the torus floorplan shows a different formation of regularity and clustering. As shown in this figure, the green and yellow nodes locations are symmetrical to each other, while the blue and white nodes are symmetrical too. This difference is caused by the "wrap around" nets added in the torus topology.

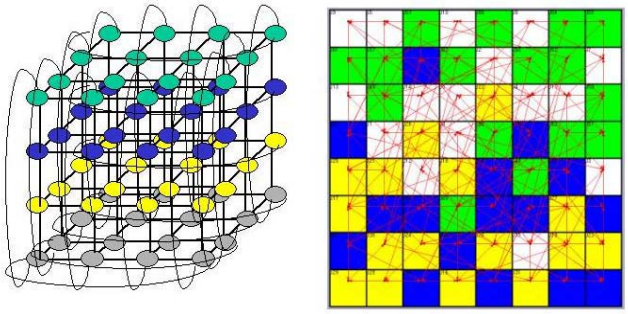

Figure 14. Floorplan of 4-ary 3-cube torus network

A 2-ary 3-fly Butterfly switch fabrics is tested as an example for indirect network. We use the same I/O constraints as described in Section 4.3, and the floorplan is legalized from the initial locations shown in Fig. 9. As illustrated in Fig. 15, under these I/O constraints, *REGULAY* creates a very dense arrangement of the switch fabrics, the regularity of the topology, as well as the locality of the I/O switches are well preserved.

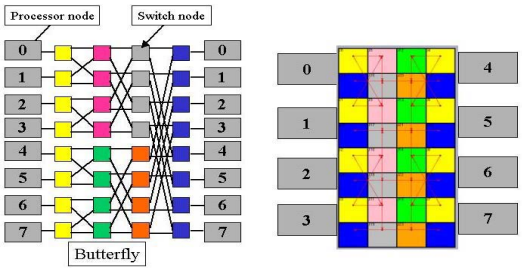

Figure 15. Floorplan comparison of constrained Butterfly network

Furthermore, we compare the total network wire-length and average net wire-length between *REGULAY* and UCLA MCM. Each PE in the network is $100\mu m \times 100\mu m$ in size. The wire-length

is the Manhattan distance between two connected PEs. The results are compared in Table 1. We further calculate the wire-length reduction (both total and average) achieved by *REGULAY* over UCLA MCM. As shown in the table, the wire-length created by *Regular* is 1.8× to 6.5× smaller in all the benchmarks. Particularly, *REGULAY* shows even greater advantages in those more complex networks: the 4-ary 3-mesh and torus network and the Octagon network achieve much higher wire-length reduction than those simpler networks.

Table 1. Wire-length comparison between REGULAY and UCLA MCM

	REGULAY		*UCLA MCM*		
	total wirelength	average wirelength	total wirelength	average wirelength	improvement
5ring Oct	12400	206	54000	900	4.4
CCC	6000	166	10800	300	1.8
4ary 3mesh	28800	200	115200	800	4.0
4ary 3torus	60800	422	393600	2733	6.5
2ary 3fly	9600	200	19200	400	2.0

7 Conclusion

In this paper, we proposed a physical floorplanning method for MPSoC on-chip network and switch fabrics, and introduced *REGULAY*, a network floorplanning tool that implements the proposed methodology. Experiments show that *REGULAY* can automatically create an optimal floorplan that preserves the regularity and hierarchy formation of the network topology, while achieving significantly reduced total wire-length compared to traditional floorplanning tools.

References

[1] Dally, William; Toles, Brian "Route Packets, Not Wires: On-Chip Interconnection Networks" *38th Design Automation Conference, 2001. Proceedings*

[2] Benini, Luca; De Micheli, Giovanni "Networks on Chips: A New SoC Paradigm" *IEEE Computers, January 2002*

[3] R. Ho, K. Mai, M. Horowitz, "The Future of wires," *Proceedings of the IEEE*, April 2001.

[4] Bryan Preas and Michael Lorenzetti "Physical Design Automation of VLSI Systems" *The Benjamin Cummings Publishing Company, 1988*

[5] Dehon, A; "Compact, Multilayer Layout for Butterfly Fat-Tree" *ACM Symposium on Parallel Algorithms and Architectures, 2000*

[6] Greenberg, R.I.; Leiserson, C.E.; "A Compact Layout for the Three-Dimensional Tree of Meshes" *Applied Math Letters, P171-176, 1988*

[7] Duato, J.; Yalamanchili, S.; Ni, L. "Interconnection Networks, an Engineering Approach" *IEEE Computer Society Press*, 1997

[8] Dally, W.J; "Performance Analysis of a k-ary n-cube Interconnect Networks" *IEEE Transactions on Computers, June, 1990*

[9] Preparata, F.P; Vuillemin, J. "The Cube-Connected Cycles: A Versatile Network for Parallel Computation" *Comm. of the ACM, May 1981*

[10] Karim, F.; Nguyen, A.; Dey, S. "On-chip Communication Architecture for OC-768 Network Processors" *38th Design Automation Conference, 2001. Proceedings*

[11] Gherrier, P.; Greiner, A "A Generic Architecture for On-Chip Packet-Switched Interconections" *DATE Conference, 2000. Proceedings*

[12] Lengauer, T; "Combinatorial Algorithms for Integrated Circuit Layout" *Wiley, John & Sons, Sep. 1990*

[13] Cong, J., et al.; "Relaxed Simulated Tempering for VLSI Floorplan Designs" *Proc. of ASP Design Automation Conference, 1999*

ISBN0-7695-1992-X/03 $17.00 © 2003 IEEE