# On-chip implementation of multiprocessor networks and switch fabrics

Terry Tao Ye

R/D Center for Logistics and Supply Chain Management (LSCM),
Hong Kong
E-mail: taoye@cslmail.stanford.edu

Giovanni De Micheli*

EPFL, Switzerland
E-mail: giovanni.demicheli@epfl.ch
*Corresponding author

**Abstract:** On-chip implementation of multiprocessor systems needs to planarise the interconnect networks onto the silicon floorplan. Compared with traditional ASIC/SoC architectures, Multiprocessor Systems on Chips (MPSoC) node processors are homogeneous, and MPSoC network topologies are regular. Therefore, traditional ASIC floorplanning methodologies that perform macro placement are not suitable for MPSoC designs. We propose an automated MPSoC physical planning methodology. *REGULAY* can generate an optimal floorplan for different topologies under different design constraints. Compared with traditional floorplanning approaches, *REGULAY* shows significant advantages in reducing the total interconnect wirelength while preserving the regularity and hierarchy of the network topology.

**Keywords:** NoCs; network on chips; MPSoCs; structures interconnect; network interface; switch; router; link; physical design; floorplan; network topology.

**Reference** to this paper should be made as follows: Ye, T.T. and De Micheli, G. (2008) 'On-chip implementation of multiprocessor networks and switch fabrics', *Int. J. Embedded Systems*, Vol. 3, No. 4, pp.209–218.

**Biographical notes:** Terry Tao Ye received his PhD Degree from the Electrical Engineering Department of Stanford University in 2003. His research area is on high performance embedded system design, which can be applied to next generation high performance, low power consumption wireless communication systems, cellular systems and PDA systems. He received his BSEE Degree from Tsinghua University, Beijing, in 1993. Starting from January 2007, he is the Director of Research and Development at Hong Kong R/D Center for Logistics and Supply Chain Management (LSCM). LSCM was established by the Hong Kong government in 2006. The center is commissioned to conduct the research, development and deployment of RFID technology into the logistic and supply chain infrastructure in Hong Kong and the greater China region.

Giovanni De Micheli is Professor and Director of the EE Institute and the Integrated Systems Center at EPFL Lausanne. He is also the President of the Scientific Committee of CSEM in Neuchatel. He was previously Full Professor of EE at Stanford University. His research activities span logic synethesis, high-level languages and design, low-power design, HW/SW co-design and network-on-chip design. He is also interested in design with hybrid technologies, such as silicon nanowires arrays and biosensors. He is a fellow of ACM and IEEE and recipient of the IEEE Piore TFA in 2003.

## 1 Introduction

Future VLSI technology will enable hundreds, or even thousands, of Processing Elements (PEs) being integrated on the same chip. At the same time, Systems-on-Chip (SoCs) applications also demand higher data-processing capability that can perform parallel and multi-threading tasks. Multiprocessor Systems on Chips (MPSoCs) combine the advantages of computation-parallelism of multiprocessors with single chip integration of SoCs. Thus MPSoCs are widely employed in today's (and also tomorrow's) Network Processors (NPs), Parallel Multimedia Processors (PMPs) and many Application Specific Array Processors (ASAPs).

In MPSoCs, on-chip multiprocessors communicate with each other independently and concurrently (Dally and Towles, 2001). Traditional SoC shared-medium communication architectures (e.g., buses) will

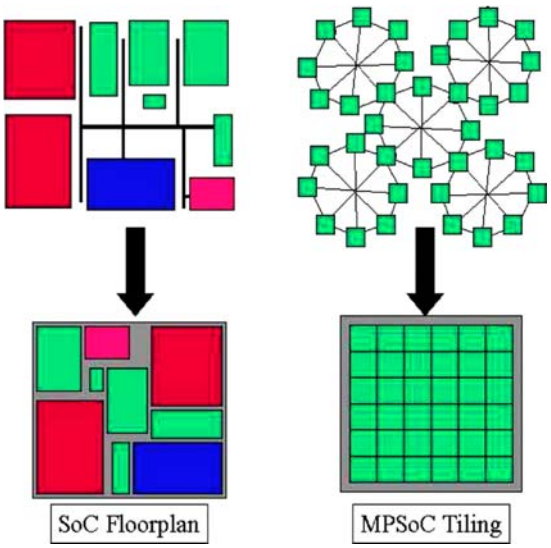
no longer be able to support the massive data traffic at this scale. In many cluster-level multiprocessor systems, the processor nodes are connected by an interconnection network. Similarly, MPSoCs also need to adopt a dedicated on-chip interconnection network that can provide reliable and scalable communication (Benini and De Micheli, 2002).

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 (Ho et al., 2001). Therefore, an optimised interconnect network floorplan will be of great importance to MPSoC performance and energy consumption.

Among many MPSoC architectures, homogeneous MPSoCs. i.e., on-chip multiprocessor systems with regular fabrics, are of particular interest because of their regular structures and parallel data computation capability. Similar to parallel computer clusters, homogeneous MPSoCs may also adopt different network topologies and switch fabrics and implement them onto the chip floorplan. In this paper, we will address the implementation issues of homogeneous MPSoC interconnection networks.

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 (Preas and Lorenzetti, 1988), MPSoC floorplanning needs to solve the problems from a different perspective, as illustrated in Figure 1. Namely:

**Figure 1** MPSoC tiling is different from traditional floorplan (see online version for colours)




- *Folding and planarisation.* 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 (Dally and Towles, 2001). The planarisation process is also constrained by the pre-defined aspect ratio and row/column numbers of the tile array.
- *Regularity and hierarchy.* MPSoC network topologies are often regular and hierarchical. The planarisation of the network is not only a simple packing process: it has to preserve the regularity and hierarchy on the floorplan.
- *Critical path and total wirelength.* Interconnect delays and power consumption are the two critical issues in MPSoC network design. On the one hand, inter-node communication latencies are dominated by the wire propagation delays. Therefore, the wirelength of the timing-critical links needs to be minimised. On the other hand, interconnect wires are the main contributors of the total system power consumption. Reducing the total wirelength helps reducing the power dissipated on the interconnect.

Prior network graph planarisation 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 (Dehon, 2000; Greenberg and Leiserson, 1998). Therefore, those approaches are not suitable for an automated design flow.

In this paper, we propose a floorplanning method and a 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 organised 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 generalises and formulates the MPSoC floorplanning problem. Our proposed floorplanning method consists of two steps: regularity extraction (Section 4), and legalisation (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 categorised as *direct networks* and *indirect networks* Duato et al. (1997). 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). Although some direct networks and indirect networks may be equivalent in functionality, e.g., if each node processor has one dedicated node switch, this node switch can either be embedded inside the node processor, or be constructed outside. Nevertheless, direct and indirect topologies have different impact on network physical implementation. 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'.

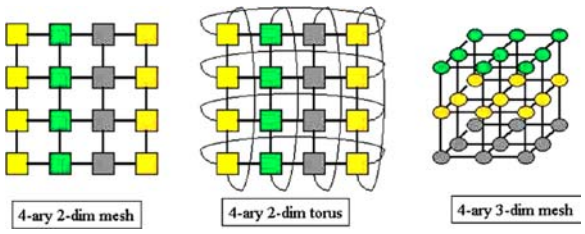
Direct networks and indirect networks can have different topologies (Duato et al., 1997). It is not the objective of this paper to discuss the functionalities and performance metrics of these different networks. Rather, we are going to give only a brief description of some of the popular network topologies. We will use these topologies as examples to formulate the MPSoC on-chip network 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 Figure 2. Because of the simple connection and easy routing provided by adjacency, mesh and torus networks are widely used in parallel computing platforms (Dally, 1990). 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.

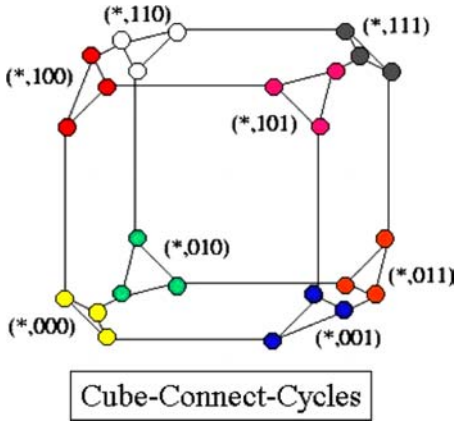
**Figure 2** Mesh and torus networks (see online version for colours)




### 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 (Preparata and Vuillemin, 1981), as shown in Figure 3(a). 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.

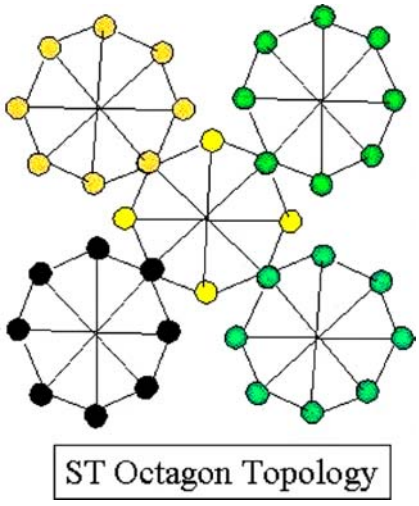
**Figure 3** Cube-Connected-Cycles networks (see online version for colours)




### 2.1.3 Octagon topology

The Octagon network is another example of direct network topologies (Figure 4). It was proposed by Karim et al. (2001) 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 stages (through one intermediate node) within the local ring. The Octagon network is scalable. If one node processor is used as the bridge node, more Octagons can be cascaded together, as shown in Figure 4.

**Figure 4** The octagon networks (see online version for colours)

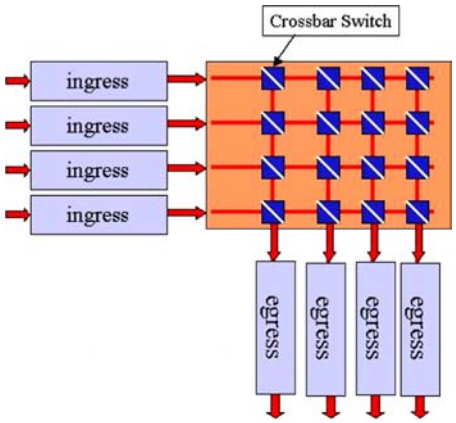



## 2.2 Indirect network topologies

### 2.2.1 Crossbar switch fabrics

An $N \times N$ crossbar network connects $N$ input ports with $N$ output ports. Any of the $N$ input ports can be connected to any of the $N$ output ports by a node switch on the corresponding crosspoint (Figure 5).

**Figure 5** Crossbar switch fabrics (see online version for colours)

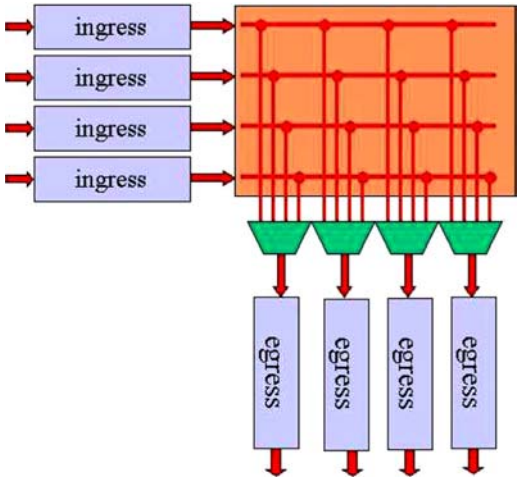



### 2.2.2 Fully-connected network

An $N \times N$ fully-connected network uses MUXes to aggregate every input to the output (Figure 6). Each MUX is controlled by the arbiter that determines which input should be directed to the output.

Similar to the crossbar network (fully connected switch network is also often referred as crossbar), in fully connected switch network, each source-destination connection has its dedicated data path.

**Figure 6** Fully-connected switch fabrics (see online version for colours)

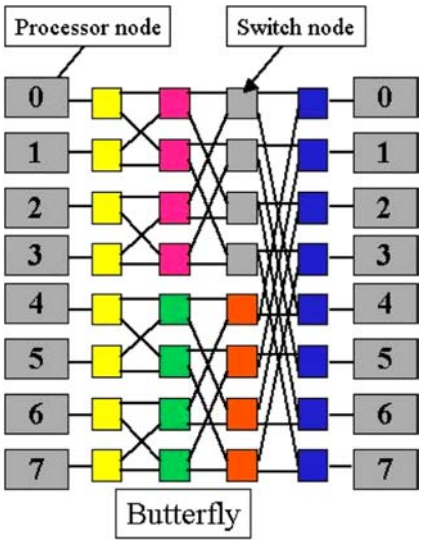



### 2.2.3 Butterfly topology

The Butterfly network (Figure 7) is an indirect network architecture. Inside the butterfly fabrics, each source-destination route uses a dedicated datapath. The delays between any two node processors 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.

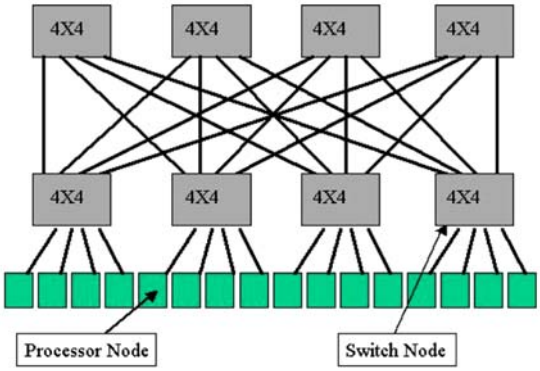
**Figure 7** The butterfly switch fabrics (see online version for colours)




### 2.2.4 Fat-tree topology

Unlike the Butterfly network, a *fat-tree* network provides multiple datapaths from source node to destination node. As shown in Figure 8, 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 (Adriahantenaina et al., 2003) is one design example that uses 4-ary fat-tree topology for the MPSoC on-chip communication.

**Figure 8** The fat-tree networks (see online version for colours)




### 2.3 MPSoC network floorplan

Although quite different in their topologies, many MPSoC networks have some important aspects in common: *regularity and hierarchy*. Regular and hierarchical topologies help to distribute the network traffic and balance the workload of node processors. Therefore, preserving the regularity and hierarchy formations in the silicon floorplan is critical in MPSoC implementation.

Furthermore, on-chip interconnect delays and power consumption add additional requirements in MPSoC floorplan design. To reduce wiring delays, MPSoC floorplans need to limit the wirelength of the critical links

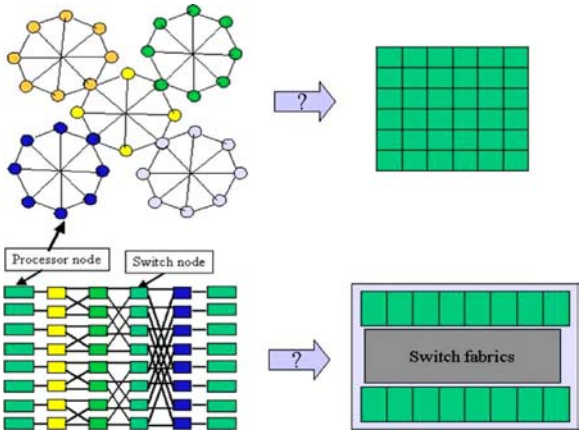
(links that are timing sensitive). To reduce the interconnect energy dissipation, the total network wirelength needs to be minimised.

## 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 $6 \times 6$ array in Figure 9. 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 Figure 9.

Formally, we are given a source network $S$ connecting a set of modules $M$, $M = \{m_i, i = 1, 2, 3, \ldots, p\}$, 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 $n_{ij,\ldots,k} \in N$ connects modules in $m_i, m_j, \ldots, m_k$ and has weight $w_{ij,\ldots,k}$.

**Figure 9** Constraints of floorplan tiling (see online version for colours)




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

- *Regularity constraints.* As shown in Section 2, MPSoC placement should preserve the regularity of the original network topology.
- *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.
- *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.
- *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.
- *Critical-path constraints.* The links between some node processors may be the critical paths, e.g., the centre ring in the cascaded Octagon network. Therefore, the nodes connected by the critical paths need to be placed closer to each other.
- *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:

- regularity extraction and determination of tentative locations
- legalisation.

The first step generates the relative locations of the modules based on the regularity and hierarchical information extracted from the network topology. 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 represent the network topology as a connectivity matrix, where each element off-diagonal of the matrix corresponds to an edge of the topology graph. We use the total square wirelength among the nodes as the objective function. The minimisation of the objective function can be calculated through a series of matrix operations.

The matrix representation preserves the topological regularity information, i.e., if the nodes in the original topology are symmetrical, the corresponding elements in the matrix are symmetrical as well. Furthermore, all subsequent matrix operations (e.g., transposition, vector multiplication, etc.) will preserve regularity. Therefore, by minimising the total square wirelength with this model, the regularity information is preserved in the optimisation process.

### 4.1 Forming the objective function

We generalise this problem by assigning weights on the nets, thus allowing us to privilege the proximity/distance of some node pairs. Thus the weighted total square wirelength objective function can be formed in the following way.

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

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

where $\mathbf{x} \in \mathbf{R}^p$ and $\mathbf{y} \in \mathbf{R}^p$ are the location vectors for the modules on X and Y dimensions. $\mathbf{Q} \in \mathbf{R}^{p \times p}$ is the matrix that represents the weighted connectivity of the topology graph, where the weight factors $\{w_{ij}, i = 1, 2, \ldots, p, j = 1, 2, \ldots, p\}$ are the matrix elements. $\mathbf{Q}$ is generated in the following way:

- $w_{ij}$ is 0 if there is no connection between modules $m_i$ and $m_j$.
- When modules $m_i$ and $m_j$ are connected, the value of $w_{ij}$ is the weighting factor of the net between $m_i$ and $m_j$.
- The diagonal elements $\{w_{ii}, i = 1, 2, \ldots, p\}$, etc., of the matrix are the opposite of the sums of all off-diagonal elements on the same row.

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

When the network topology graph is connected, it can be proved that the matrix $\mathbf{Q} \in \mathbf{R}^{p \times p}$ constructed from this graph has the following properties (Hall, 1970):

- $\mathbf{Q}$ is positive positive semi-definite
- is of rank $p - 1$.

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 Floorplan without I/O constraints

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

#### 4.2.1 X-dimension optimisation

Since there is no I/O or boundary condition, we need to further normalise the objective function on the X-dimension by using the inner product $\mathbf{x^T x}$.

Now the normalised objective function of equation (1) can be rewritten as

$$\Phi'(x) = \frac{\mathbf{x^T Q x}}{\mathbf{x^T x}}. \quad (2)$$

From the construction of matrix $\mathbf{Q}$, we note that the row sums of $\mathbf{Q}$ are zero, thus $\mathbf{Q}$ has a unit eigenvector $\mathbf{u} = (1, 1, 1, \ldots, 1)^T$. The associated eigenvalue is zero. We also note that $\mathbf{Q}$ is symmetrical and has rank $p - 1$. Therefore, it has $p$ non-negative real eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_p \in \mathbf{R}$. The smallest eigenvalue is $\lambda_1 = 0$.

It can be proven that the first partial derivative with respect to the vector $\mathbf{x}$ of the normalised objective function is zero when

$$(\mathbf{Q} - \lambda \mathbf{I})\mathbf{x} = 0 \quad (3)$$

which yields a non-trivial solution of $\mathbf{x}$ if and only if $\mathbf{x}$ is the eigenvector of the corresponding eigenvalue of $\lambda$. Here $\mathbf{I}$ is the identity matrix.

It can be shown that the normalised objective function is bounded between the minimum and maximum eigenvalues, or

$$\lambda_{\min} \leq \Phi'(x) = \frac{\mathbf{x^T Q x}}{\mathbf{x^T x}} \leq \lambda_{\max}. \quad (4)$$

The minimum eigenvalue, zero, yields the trivial solution of unit vector, where all nodes are to be placed at one single point. Therefore, the second smallest eigenvalue and the associated eigenvector $e_1$ yield the optimal solution.

#### 4.2.2 Y-dimension optimisation

Since we already use $e_1$ to form the location vector $\mathbf{x}$ on the X coordinate, the Y-dimension location vector $\mathbf{y}$ has to be formed from other eigen-vectors, otherwise, the modules will be placed in a diagonal line on the floorplan. This condition add one extra constraint to $\mathbf{y}$ vector.

$$\mathbf{y^T e_1} = 0. \quad (5)$$

Since the eigenvectors of the symmetrical matrix $\mathbf{Q}$ are orthogonal, we will choose for $\mathbf{y}$ the eigenvector corresponding to the third smallest eigenvalue.

Figure 10 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 non-zero eigenvectors of $\mathbf{Q}$. From the locations of the nodes, we can see that not only the regularity formation of the nodes is preserved, but also the hierarchical clustering of the cascaded Octagon rings is shown as well.

Figure 11 shows the initial locations of the cube-connect-cycles obtained from the eigenvectors of the matrix. Again, the formation of the nodes preserves the regularity as well as the hierarchy of the original topology.

**Figure 10** Initial eigenvector locations of 5-ring Octagon network without I/O constraints (see online version for colours)

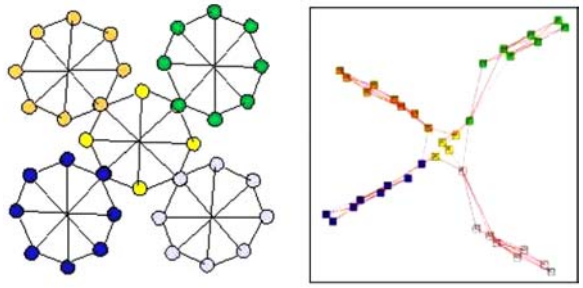
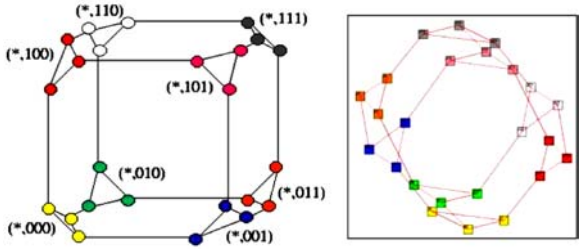


**Figure 11** Initial eigenvector locations of CCC without I/O constraints (see online version for colours)




### *4.3 Floorplan with I/O constraints*

For this problem, we can again decouple the optimisation in the X and Y directions. We will first describe the optimisation in the X direction, the optimisation in the Y direction is similar.

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 in the X direction 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:

$$\Phi(x) = (\mathbf{x_c} \quad \mathbf{x_f}) \begin{pmatrix} Q_{cc} & Q_{cf} \\ Q_{fc} & Q_{ff} \end{pmatrix} (\mathbf{x_c} \quad \mathbf{x_f})^{\mathbf{T}}. \tag{6}$$

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

$$\mathbf{Q_{cc}x_c} = -\mathbf{Q_{cf}x_f}. \tag{7}$$

Here $\mathbf{Q_{cc}}$ is a subset of the original matrix $\mathbf{Q}$. We know that $\mathbf{Q}$ is a symmetrical matrix with the rank of $p-1$. Therefore, if at least one node is fixed, $\mathbf{Q_{cc}}$ is non-singular and invertible, and the equation (7) has real solutions (Alpert et al., 1997).

Figure 12 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 (6). Under the I/O constraints, the Octagon network shows a different formation than that without I/Os (Figure 10). Nevertheless, the regularity as well as the hierarchical formation of the network is still preserved.

**Figure 12** Initial locations of 5-ring Octagon network with I/Os on the corners (see online version for colours)

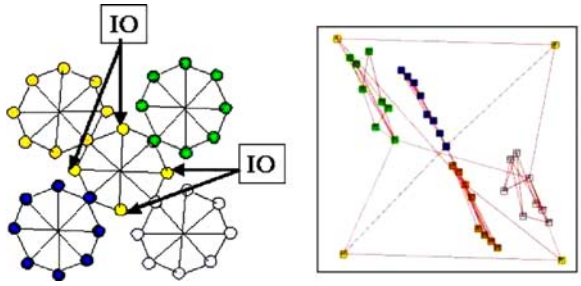
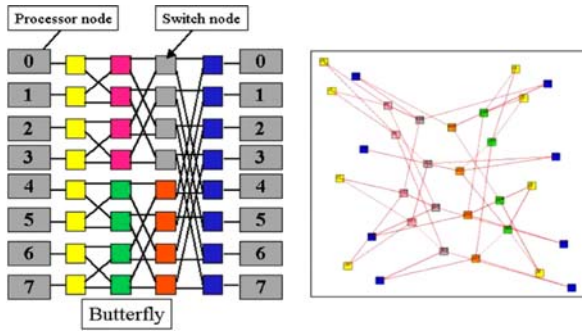



Figure 13 shows the initial locations of the 2-ary 3-fly Butterfly switch fabrics. There are 8 node processors and 32 node switches in the network. The node processors are numbered from 0 to 7, as shown in Figure 13. One node processor connects to two node switches, serving as input switch and output 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 13** Initial locations of butterfly network with I/O constraints (see online version for colours)




## 5 Legalisation

The node positions solved from the quadratic objective function optimisation are real-valued numbers. However, the PE modules of each node are rectangular tiles and have to be abutted next to each other. The position from the location vector cannot be used directly in the tiling placement. Nevertheless, we can still use these values as relative locations and further legalise (quantise) the node positions.

The legalisation procedure also consists of two steps:

- sorting, where the nodes are ordered by the X and Y coordinates
- packing, where the node blocks are assigned to the corresponding tiles (row and column positions).

In the sorting step, as shown in Figure 14, 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 legalisation procedure involves two linear sorting operations, which can be implemented with any existing sorting algorithms.

Figure 15 shows the legalisation results. Figure 15(a) is the legalised floorplan from Figures 10 and 15(b) is legalised from Figure 12. Both floorplans preserve the regularity and hierarchy formations of the original topologies. Furthermore, the floorplan also achieves a shorter total interconnect wirelength compared with other macro cell floorplanning approaches. We will show this comparison in details through several experiments.

**Figure 14** Legalisation of the node locations by sorting and packing (see online version for colours)

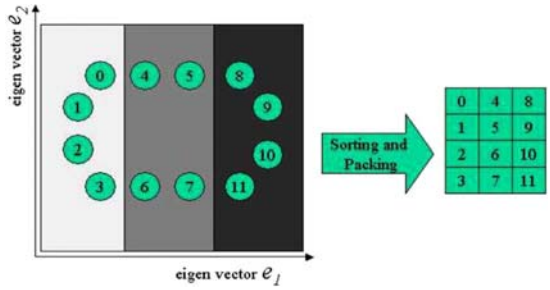
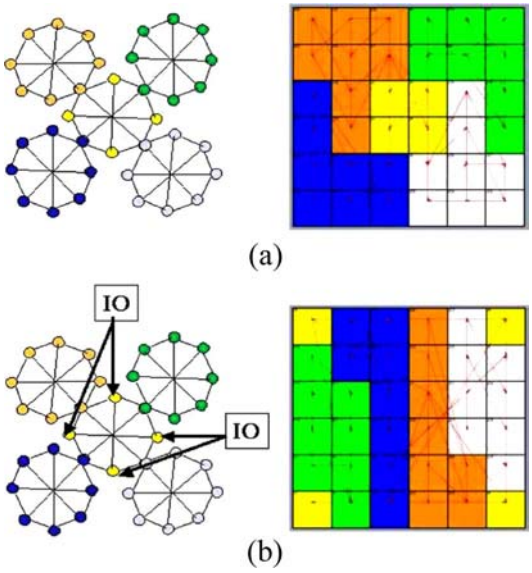



**Figure 15** Legalised floorplan of octagon networks with and without I/O constraints (see online version for colours)

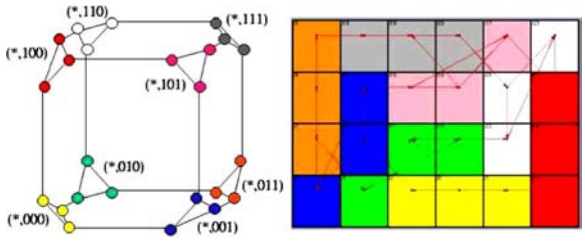



## 6 Experiments

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 our 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 wirelength with the results obtained from ASIC floorplanning approaches. We choose UCLA MCM floorplanner (Cong et al., 1999) 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.

Figure 16 shows the floorplan result of the CCC by *REGULAY*. There are total 24 nodes and 36 nets in the topology. For a better visualisation of the regularity and hierarchy of the resulting floorplan, we assign different colours (or shades, if viewed in black and white displays) 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*.

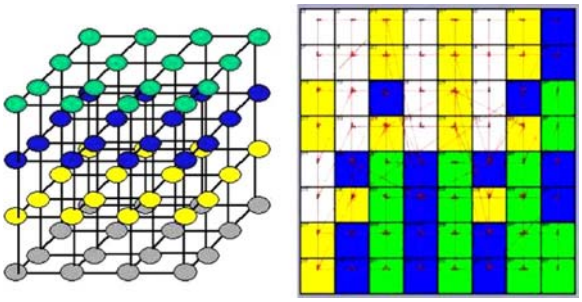
**Figure 16** Floorplan of Cube-Connected-Cycles network (see online version for colours)




The 4-ary 3-mesh network floorplan result is shown in Figure 17. There are 64 nodes and 144 interconnects in this network, and the floorplan is an $8 \times 8$ 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 colours for a better visualisation (In black and white displays, the nodes are shown in four different shades). 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 18 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 $8 \times 8$ 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

**Figure 17** Floorplan of 4-ary 3-mesh network (see online version for colours)



are symmetrical too. This difference is caused by the 'wrap around' nets added in the torus topology.

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 legalised from the initial locations shown in Figure 13. As illustrated in Figure 19, 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 18** Floorplan of 4-ary 3-cube torus network (see online version for colours)

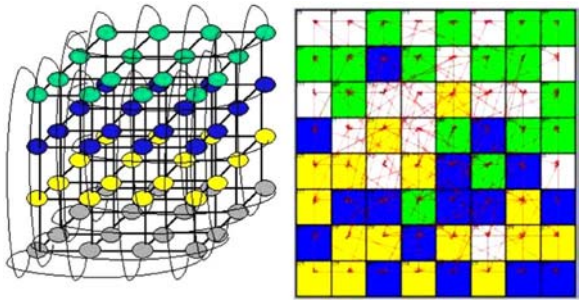
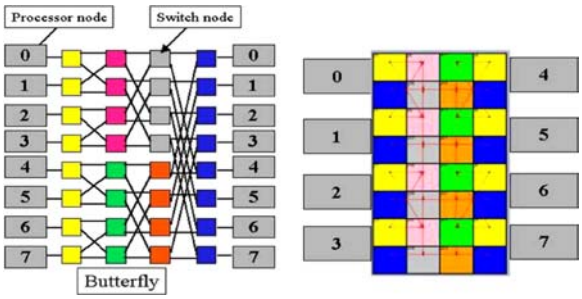


**Figure 19** Floorplan comparison of constrained butterfly network (see online version for colours)




Furthermore, we compare the total network wirelength and average net wirelength between *REGULAY* and UCLA MCM. Each PE in the network is $100 \times 100\,\mu m$ in size. The wirelength is the Manhattan distance between two connected PEs. The results are compared in Table 1. We further calculate the wirelength reduction (both total and average) achieved by *REGULAY* over UCLA MCM. As shown in the table, the wirelength created by *Regular* is $1.8\times$ to $6.5\times$ 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 wirelength reduction than those simpler networks.

**Table 1** Wirelength comparison between REGULAY and UCLA MCM

| | *REGULAY* | | *UCLA MCM* | | |
|---|---|---|---|---|---|
| | *Total wire-length* | *Average wire-length* | *Total wire-length* | *Average wire-length* | *Improvement* |
| 5-ring Oct | 12400 | 206 | 54000 | 900 | 4.4 |
| CCC | 6000 | 166 | 10800 | 300 | 1.8 |
| 4-ary 3-mesh | 28800 | 200 | 115200 | 800 | 4.0 |
| 4-ary 3-torus | 60800 | 422 | 393600 | 2733 | 6.5 |
| 2-ary 3-fly | 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 wirelength compared to traditional floorplanning tools.

## References

Adriahantenaina, A., Charlery, H., Greiner, A., Mortiez, L. and Zeferino, C.A. (2003) 'SPIN: a scalable, packet switched, on-chip micro-network', *Proceedings of the Design Automation and Test in Europe*, March, pp.70–73.

Alpert, C.J., Chan, T., Huang, D.J-H., Markov, I.L. and Yan, K. (1997) 'Quadratic placement revisited', *Proceedings of the Design Automation Conference*, June, pp.752–757.

Benini, L. and De Micheli, G. (2002) 'Networks on chips: a new SoC paradigm', *IEEE Computer*, Vol. 35, No. 1, January, pp.70–78.

Cong, J. *et al.* (1999) 'Relaxed simulated tempering for VLSI floorplan designs', *Proceedings of ASP Design Automation Conference*, January, pp.13–16.

Dally, W. and Towles, B. (2001) 'Route packets, not wires: on-chip interconnection networks', *Proceedings of the 38th Design Automation Conference*, June, pp.684–689.

Dally, W.J. (1990) 'Performance analysis of a k-ary n-cube interconnect networks', *IEEE Transactions on Computers*, June, pp.775–785.

Dehon, A. (2000) 'Compact, multilayer layout for butterfly fat-tree', *ACM Symposium on Parallel Algorithms and Architectures*, pp.206–215.

Duato, J., Yalamanchili, S. and Ni, L. (1997) *Interconnection Networks, an Engineering Approach*, IEEE Computer Society Press.

Greenberg, R.I. and Leiserson, C.E. (1998) 'A compact layout for the three-dimensional tree of meshes', *Applied Math Letters*, pp.171–176.

Hall, K. (1970) 'An r-dimensional quadratic placement algorithm', *Management Science*, Vol. 17, No. 3, November, pp.219–229.

Ho, R., Mai, K. and Horowitz, M. (2001) 'The future of wires', *Proceedings of the IEEE*, April, pp.490–504.

Karim, F., Nguyen, A. and Dey, S. (2001) 'On-chip communication architecture for OC-768 network processors', *Proceedings of 38th Design Automation Conference*, June, pp.678–683.

Preas, B. and Lorenzetti, M. (1988) *Physical Design Automation of VLSI Systems*, The Benjamin Cummings Publishing Company.

Preparata, F.P. and Vuillemin, J. (1981) 'The cube-connected cycles: a versatile network for parallel computation', *Comm. ACM*, May, pp.300–309.