The Y-Architecture: Yet Another On-Chip Interconnect Solution

Hongyu Chen, Bo Yao, Feng Zhou, and Chung-Kuan Cheng

Department of Computer Science and Engineering, University of California, San Diego
9500 Gilman Dr.; La Jolla, CA 92093-0114; U.S.A.

Abstract In this paper, we propose a new on-chip interconnect scheme called Y-architecture, which can utilize the on-chip routing resources more efficiently than traditional Manhattan interconnect architecture by allowing wires routed in three directions (0°, 60°, and 120°). To evaluate the efficiency of different interconnect architectures, we assume mesh structures with uniform communication demand and develop a multi-commodity flow (MCF) approach to model the on-chip communication traffic. We also extend the combinatorial MCF algorithm in [5] to compute the optimal routing resource allocations for different interconnect architectures. The experiments show that: (1) Compared with Manhattan architecture, the Y-architecture demonstrates a throughput improvement of 30.7% for square chip. The throughput of the Y-architecture is only 2.5% smaller than that of X-architecture. (2) A chip with the shape of a convex polygon produces better throughput than a rectangular chip: For Y-architecture, a hexagonal chip provides 41% more throughput than a squared chip using the Manhattan architecture. For Manhattan architecture, a diamond chip achieves a throughput improvement of 19.5% over the squared chip using the same interconnect architecture. (3) Compared with Manhattan architecture, the Y-architecture reduces the wire length of a randomly distributed two pin net by 13.4% and the average wire length of Y-architecture is only 4.3% more than that of the X-architecture.

I Introduction

With rapid technology scaling, the interconnect becomes one most precious resource on a chip. Traditional Manhattan interconnect architecture organizes wires on two orthogonal routing directions, 0° and 90° directions, for the simplicity of routing embedding and design rule checking. However, its artificial restriction on routing directions adds significant wire length over the Euclidean optimum and thus decreases the communication capability of the on-chip interconnects.

In the past decade, many researchers have explored the possibility of using nonrectilinear wires to improve the efficiency of on-chip interconnects.[4] Most of these work discussed about how to introduce 45-degree short jogs to improve the routability of the chip in the detailed routing stage. Majority of the wires on the chip are still routed on either 0-degree or 90-degree direction.

Recently, Mutrunoi et. al. [7] proposed a new on-chip interconnect architecture named the X-architecture, which is targeting at the designs with 5 or more routing layers. In the X-architecture, the wires are organized in 0-degree, 45-degree, 90-degree and 135-degree directions. The experimental results show that it achieves a chip performance improvement of 10% and power reduction of 20% than Manhattan architecture for a high performance design.

In the foreseeable future, more than 12 routing layers will be available in the high performance circuit designs.[1] It is both possible and desirable for us to explore the different ways to organize the on-chip routing resources. However, the prohibitive cost of actually designing and manufacturing a chip with new interconnect architectures makes it very hard to implement and test new interconnect architectures one by one. It is necessary to develop a quantitative framework to evaluate the efficiency of different interconnect architectures.

From early 1990s, researchers [2][6] studied the wire length reduction by allowing more routing directions. Most of these works concentrated on the Steiner cost of a single signal net. The competition on the routing resources between different nets is ignored in these works.

We adopt Multi-commodity flow (MCF) approach to model the on-chip communication traffic. We assume a mesh structure with uniform communication demand, and use the MCF throughput of the mesh to measure the communication capability of different interconnect architectures. This method is independent with particular test cases and can reflect the exact communication bottleneck on the chip.

Recent advance in MCF algorithm [5] allows us to solve MCF problem much more efficiently. Our implementation can compute the throughput of meshes with up to 289 nodes within 12 hours computing. We also extend this algorithm to compute the optimal routing resource allocation for different interconnect architectures.

In this paper, we propose the use of Y-architecture, which has three symmetrical routing directions (0-degree, 60-degree, and 120-degree). We compare this interconnect architecture with the Manhattan architecture and the X-architecture. Based on our experiments on throughputs and analytical analysis on wire length, we have following reason to motivate the use of the Y-architecture:

(1) Using Y-architecture can gain a throughput improvement of 33.3% over the traditional Manhattan architecture on a squared mesh. Comparing with the X-architecture, which has one more routing direction, the Y-architecture produces almost the same (2.6% less) throughput on a squared mesh.

(2) The Y-architecture achieves an average of 13.4% wire

length reduction over Manhattan architecture for a two-pins connection. The more complex X-architecture can only improves this reduction of 4.3% further.

(3) Making chip shape close to a circle can significantly improve the throughput over the rectangular chip. Using Y-architecture, we can make hexagonal chip, which can produce 41% more throughput than square chip using Manhattan architecture, without causing dead space on the wafer.

The rest of this paper is organized as follows: Section 2 presents the problem formulation in an MCF model. Section 3 compares the throughput of meshes using Manhattan architectures, the Y-architecture, and the X-architecture. In subsection 3.1, we compare the throughputs of n by n meshes using different interconnect architectures, these experiments are designed to test the communication capability of different interconnect architectures on a square chip. In subsection 3.2, we set the chip shape to be close to a circle, e.g. hexagons for the Y-architecture, octagons for the X-architecture, and diamonds for the Manhattan architecture, and compare the throughput of meshes using different interconnect architectures. Following that we discuss the wirelength of two pin nets using different interconnect architectures in section 4. Finally, we conclude this paper in section 5.

II. Problem Formulation

2.1 MCF with constant edge capacities

For Manhattan architecture, we decompose the communication resources into an array of $n \times n$ slots. Each slot contains a communication terminal, say, a processor. The slots are aligned in rows and columns. The slot array forms a 90-degree mesh structure. Figure 1(a) illustrates an example of a 90-degree mesh structure with 25 slots. Each square tile represents a slot. The mesh structure can be mapped to a graph $G=\{V, E\}$ according the following rules:

(1) Each slot i corresponds to the node i in the graph.

(2) The adjacency between two slots (i,j) is represented by an edge $e = (i,j)$ in the graph

(3) The edge capacity $c(e)$ is proportional to the length of the line segment separating the adjacent slots, and the number of routing layers.

Figure 1(b) describes the graph corresponding to the mesh in Fig.1(a).

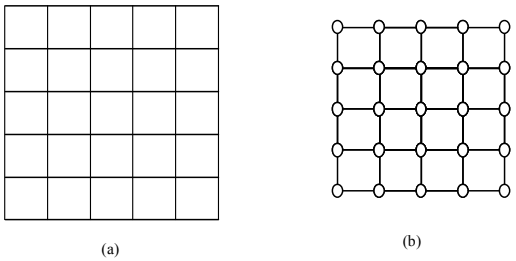


Fig. 1 A 5 by 5 communication mesh and its graph representation

We assume a uniform communication requirement i.e. every pair of nodes communicate with an equal demand. All communications happen at the same time. Note that the model can be extended to various communication demands, e.g., Poisson distribution, Rents rule, etc., depending on specific applications. In this paper, the uniform pairwise communication model is adopted because of its simplicity and genericalness. Moreover, the communication demand presents an unbiased symmetry, which makes the solution independent of the test cases, placement, and routing.

We define the throughput, z, to be the maximum amount of communication flow between every pair of nodes. We find the throughput using a multi-commodity flow model. The flow that starts from node i is deemed as commodity i. Commodity i starts from node i with the amount of $z \cdot (N-1)$, where $N = n^2$ is the number of nodes in the graph, to each of the rest nodes with the amount of z. We solve the multi-commodity flow problem to find the throughput z.

The above MCF problem can be formulated as a linear program in either the node-arc form (LP1) or the edge-path form (LP2).

a. Node-Arc form of MCF (LP1):

$$Maximize: z$$

$$S.t. \sum_{j \in neighbor\ of\ i} (f_{ji}^v - f_{ij}^v) = \begin{cases} -z \cdot (n^2 - 1) & if\ i = v \\ z & otherwise \end{cases} \quad \text{for all nodes } v, i \in V \quad (1.1)$$

$$\sum_{v \in V} (f_{ij}^v + f_{ji}^v) \le c_{ij} \quad \text{for all edges } (i,j) \in E \quad (1.2)$$

In this linear program, flow variable f_{ij}^v represents the flow amount of commodity v on edge (i,j). The edge capacity c_{ij} represents the flow capacity of edge (i,j), in a uniform mesh using X-architecture, we set c_{ij} to be unit for all (i,j). We set that the flow injecting to a node is positive and the flow ejecting from a node is negative

b. Edge-Path form of MCF (LP2):

Maximize: z

S. t.: $\sum_{p \in P_{ij}} f(p) - z \ge 0$ for nodes $i,j \in V, i \ne j$ (2.1)

$\sum_{p \in Pe} f(p) \le c(e)$ for all edges $e \in E$ (2.2)

In this linear program, we denote by P_e the set of all paths p containing the edge e, and by P_{ij} the set of all paths

between nodes *i,j*, the flow variables *f(p)* represents the flow amount of path *p*.

Note that the number of linear constraints in linear program LP1 is $|V|^2 + |E|$. Thus the linear program LP1 can be solved in polynomial time using any polynomial time linear program solver[6]. However, when *n* increases, the number of linear constraints explodes at the rate of n^4 for a *n* by *n* mesh. So, for large cases it is impractical to solve the MCF using linear programming.

In [5], a combinatorial $(1+\varepsilon)$-approximation approach is proposed to solve the MCF problem. It adopts the primal-dual structure of the linear program LP2. The algorithm assigns a nonnegative shadow cost [9] to each edge according to the congestion level on that edge. Initially, all the shadow costs are set to be equal. Then, the algorithm proceeds in iterations. In each iteration, we reroute a fixed amount of flow along the shortest path for every commodity. At the end of each iteration, we adjust the capacity of every edge and its shadow cost according to the dual linear program.

For any given error tolerance ε, this MCF algorithm can find a $(1+\varepsilon)$ approximation of the throughput in $O(\frac{1}{\varepsilon'}\log_{1+\varepsilon'}\frac{n}{1-\varepsilon'}n^4\log n)$ time, where $\varepsilon' = 1-(1+\varepsilon)^{-1/3}$.

2.2 MCF with edge capacities in linear constraints

For X-architecture, we add 45-degree edges to the 90-degree mesh of Manhattan architecture. Fig. 2 illustrates an example of 5 by 5 mesh with X-architecture. Fig. 2(a) shows the slots arrangement and Fig. 2(b) is the corresponding communication graph.

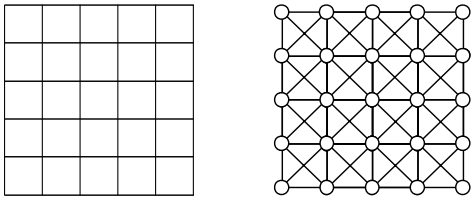


Fig. 2 A 5 by 5 mesh with X-architecture

In Fig. 2(b), the edges are oriented in 0˚, 90˚, 45˚ or 135˚ angle. All nodes are aligned in rows and columns. Thus, all the edges in 45˚ and 135˚ directions have the same capacity and all the edges in 0˚ and 90˚ directions have the same edge capacity. Note that the length of an edge in 45˚ or 135˚ direction is $\sqrt{2}$ times of that of an edge in 0˚ or 90˚ direction. Hence, if route some number of wires on an edge in 0˚ or 90˚ direction consumes one unit of routing area, then route same number of wires on an edge in 45˚ or 135˚ direction costs $\sqrt{2}$ units of routing area. In other words, for a pair of routing layers, if we can allocate a capacity of *x* to 0˚ and 90˚ edges, we can only allocate a capacity of $x/\sqrt{2}$ to 45˚ and 135˚ edges. Let c_1 be the capacity of horizontal and vertical edges, c_2 be the capacity of 45˚ and 135˚ edges. The area constraints can be expressed as $c_1 + \sqrt{2}\cdot c_2 = 1$. Thus, the total area is equal to the constant.

For Y-architecture, we set the shape of slots to be hexagon. Fig. 3(a) illustrates a hexagonal mesh with 19 slots. Fig. 3(b) is the corresponding communication graph. In Fig. 3(b) all edges are symmetrically oriented in 0˚, 60˚ or 120˚ direction and every edge has the same length. Hence, the routing area constraint for Y-architecture can be expressed as $c_1+c_2+c_3=2$, where c_1, c_2, and c_3 are edge capacity for edges oriented in 0˚, 60˚ and 120˚ directions, respectively.

We can add the routing area constraint into the Linear program LP1 or LP2 and treat edge capacities as variables.

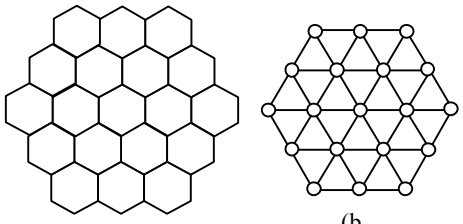


Fig. 3 A hexagonal mesh of 19 nodes with Y-architecture

The optimal solution of the linear program produces an optimal routing resource allocation for different routing directions. Following is the formal formulation of routing resource allocation problem.

Input: communication graph $G = (V, E)$, k different routing channels $\{R_1, \ldots, R_k\}$, where $\bigcup_i R_i = E$ and $\bigcap_i R_i = \Phi$; edge capacity c_i for every edge in the routing channel R_i,; and area constraints $\sum_i \alpha_i C_i = 1$

Output: an routing resource allocation $\{c_i\}$, such that the communication graph $G = (V,E)$ has maximum throughput.

The routing resource allocation problem can be written as the following linear program:

$$Min : \sum_i \alpha_i C_i$$

$$S.t. \sum_{p \in P_{ij}} f(p) \geq 1$$

$$\sum_{p \in P_e} f(p) \leq C_i \quad \text{for all distinct vertices pair } i,j \in V \qquad (3.1)$$

$$\text{for all routing channel } R_i \qquad (3.2)$$

This linear program finds the minimum routing area that can satisfy the unit pairwise communication demand. The dual program of this linear program is:

$$Max: \sum_{ij} \lambda_{ij}$$

$$S.t. \quad \lambda_{ij} \le \sum_{e \in P_{ij}} d_e \quad \text{for all distinct vertices pair } i,j \in V \quad (3.1)$$

$$\sum_{e \in R_i} d_e \le \alpha_i \quad \text{for all edge } e \in R_i \quad (4.2)$$

The dual program assigns a nonnegative shadow cost d_e to each edge e, such that the sum of the shortest distances between every distinct pair of nodes is maximized. The constraints (3.2) denotes that the total shadow costs of all edges in a routing channel is smaller than or equal to the area coefficient of that routing channel.

We extend the combinatorial $(1+\varepsilon)$-approximation scheme in [5] to solve the routing resource allocation problem. We also adopt the primal-dual structure of the linear program. The shadow cost is determined by the flow congestion level on the edge. Let $g(e) = \frac{f(e)}{c_e}$ be the congestion level of edge e, where $f(e)$ is the total flow amount going through edge e, and C_e is the capacity of e. The shadow cost is computed using $d(e) = \frac{\exp(\beta(g(e) - g^*)}{\sum_{e' \in E} \exp(\beta(g(e') - g^*)}$, where $g^* = max\{g(e) | e \in E\}$, and β is a constant related to desired approximation error ε. Initially, all the shadow costs are set to be equal. Then, the algorithm proceeds in iterations. In each iteration, we route a fixed amount of flow along the shortest path for every commodity. At the end of each iteration, we adjust the capacity of every edge and its shadow cost according to the dual linear program. Fig. 4 shows the pseudo-code of our routing resource allocation algorithm.

Algorithm
For all $e \in E$, set d_e = *constant*
Repeat
  For $j := 1$ to k do //k: number of distinct flow demands
    Begin
      Set $d(j) = \sigma$
      While $d(j) \neq 0$ do
        Begin
        Find shortest path P for commodity flow demand j.
        Route $f = min\{c, d(j)\}$ units of flow along P, where c is the capacity of the minimum capacity edge on this path.
        $d(j) = d(j) - f$
        Update $\{d_e\}$.
        End while
    End for
  Find $\{C_1, C_2, ..., C_m\}$, such that $\sum_{e \in R(i)} d_e = \alpha_i$ and $\sum_i \alpha_i C_i = 1$
  Update $\{d_e\}$
Until flow solutions converge

Fig. 4. Routing Resource Allocation Algorithm

III Throughputs of Meshes with Different Interconnect Architectures

We first use Matlab's linear program package on a Sun Ultra10 workstation to compute MCF solutions. For the case with 100 nodes, the run time exceeds 24 hours. We then implement the MCF algorithm [5] and our routing resource allocation algorithm using C programming language. Our implementation derives the MCF solutions for cases with up to 289 nodes within 12 hours.

We compare the throughputs of meshes with different interconnect architectures. In subsection 3.1, we set the chip shape to be square. Then use our extended flow approach to compute the throughputs of different interconnect architectures with optimal routing channel allocation. In subsection 3.2, we set the chip shape to be fully symmetrical to all routing directions, i.e. diamond for Manhattan architecture, hexagon for Y-architecture, and octagon for X-architecture.

3.1 Throughputs with square chip shape

In this subsection, we compare the throughput of n by n meshes using Manhattan architecture, Y-architecture, and X-architecture. Fig. 5 demonstrates three 7 by 7 meshes using different interconncect architectures. For an n by n mesh, the enclosing box of the slots is close to a rectangle. The throughput of an n by n mesh using a certain interconnect architecture demonstrates the communication ability of that interconnect architecture on a rectangular chip.

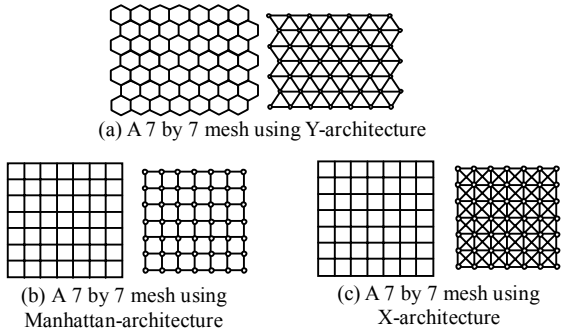


Fig.5 7 by 7 meshes with different interconnect architectures

Note that for an n by n mesh with Y-architecture, there are $3n^2-4n+1$ edges, for an n by n mesh with Manhattan architecture, there are $2n^2-2n$ 0-degree and 90-degree edges; for an n by n mesh with X-architecture, there are $2n^2-2n$ edges on 0˚or 90˚ direction and $2(n-1)^2$ edges on 45˚or 135˚ direction. To fairly compare the throughput of meshes with different interconnect architectures, we need to allocate the same amount of routing resources to meshes of the same size. In our experiments, we set the sum of all edge capacities equal to $2n^2-2n$ for all n by n meshes, and use our routing resource allocation algorithm to find the optimal allocation

of edge capacities.

For *n* from 2 to 17, Table 1 lists the throughputs of *n* by *n* meshes with different interconnect architectures. We normalized the throughput using a factor $m^{0.5}(m-1)$, where *m* is the number of nodes in the mesh. By doing so, we can keep the total amount of communication demand and total edge capacities independent with the dimension of the mesh. The third and the forth column show throughput and normalized throughput of meshes using Manhattan architecture. The fifth and seventh column depicts the normalized throughput of meshes using Y-architecture and X-architecture, respectively. We list the throughput improvement achieved by Y-architecture and X-architecture in the sixth and the eighth column.

n	#nodes	M-architecture		Y-architecture		X-architecture		
		thrpt	n. thrpt	n..thrpt	impr. (%)	n. thrpt	impr. (%)	$\sqrt{2}\cdot c_2/c_1$
2	4	2.50e-1	2.00	2.00	0	2.00	0	0.00
3	9	8.33e-2	2.25	2.25	0	2.25	0	0.00
4	16	3.12e-2	2.00	2.54	27.2	2.60	29.8	3.36
5	25	1.67e-2	2.09	2.58	23.4	2.68	28.1	2.88
6	36	9.26e-3	2.00	2.59	29.6	2.65	32.8	4.39
7	49	5.95e-3	2.04	2.59	27.2	2.67	31.1	3.94
8	64	3.90e-3	2.00	2.56	27.9	2.69	34.6	5.19
9	81	2.78e-3	2.03	2.63	29.5	2.69	32.7	4.89
10	100	1.98e-3	2.00	2.61	30.3	2.67	33.3	5.44
11	121	1.51e-3	2.01	2.64	31.8	2.70	34.4	5.12
12	144	1.16e-3	2.00	2.61	30.4	2.69	34.5	5.26
13	169	9.15e-4	2.01	2.61	30.0	2.70	34.4	5.33
14	196	7.29e-4	2.00	2.61	30.3	2.69	34.5	5.62
15	225	5.95e-4	2.01	2.61	30.6	2.70	34.5	5.51
16	256	4.88e-4	2.00	2.62	31.0	2.69	34.6	5.65
17	289	4.08e-4	2.00	2.62	31.1	2.70	34.6	5.56

Table 1. Throughput of rectangular meshes

For *n* from 10 to 17, Y-architecture provides an average throughput improvement of 30.7% for an *n* by *n* mesh, and X-architecture achieves 34.5%. For a 17 by 17 mesh, Y-architecture provides an throughput improvement of 31.1% and X-architecture achieves an improvement of 34.6%.

For Y-architecture and Manhattan architecture, equally distributed edge capacities produces maximum throughput on *n* by *n* meshes. For X-architecture, we show the optimum ratio of the area of diagonal routing edges to that of Manhattan edges in the last column. That ratio approaches 5.65 when *n* increases.

Fig. 6 shows bottlenecks of communication flows for three 12 by 12 meshes using different interconnect architectures. The fully saturated edges are highlighted with bold lines. Note that the saturated edges form vertical and horizontal cut sets for both interconnect architectures. The cut lines are shown with dashed lines.

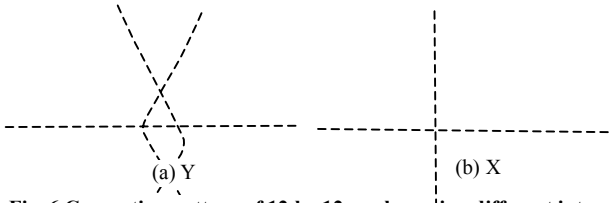
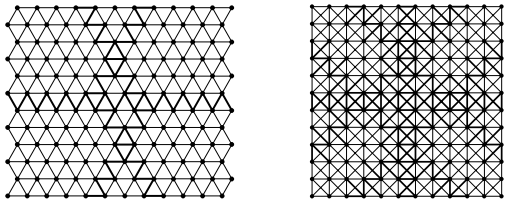


Fig. 6 Congestion pattern of 12 by 12 meshes using different interconnect architectures



Summing up the capacities of the edges passing across the cut lines, we can derive a throughput upper bound for *n* by *n* meshes with different interconnect architectures.

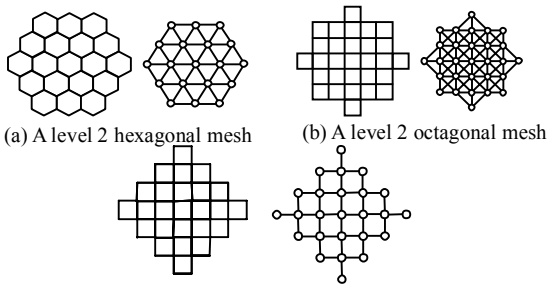
For Manhattan architecture, there are *n* edges crossing each cut line. The total edge capacity is *n*. For Y-architecture, there are *2n-1* edges passing across each cut line and each edge has capacity 2/3, the total edge capacity crossing the cut line is *(4n – 2)/3*, when *n* approaches infinity, an *n* by *n* mesh using Y-architecture can have (4/3 –1) = 33.3% more flow crossing the cutline. Thus, Y-architecture can achieve at most 33.3% throughput improvement over Manhattan architecture on a squared mesh.

For X-architecture, there are *2(n-1)* diagonal edges and *n* Manhattan edges crossing each of the two cut lines. To achieve maximum throughput, the ratio of the capacity for diagonal edges and the capacity for Manhattan edges is 5.6. Under this ratio, the edge capacities are 0.1515 and 0.6 for Manhattan edges and diagonal edges respectively. The total flow amount can go across the cut line is *1.3515n-1*. When *n* approaches infinity, the throughput improvement bound is 35.6%.

For all the cases have been tested (*n = 2* to *17*), we all observed this kind of central horizontal and vertical cut sets in n by n meshes using both X, Y and Manhattan architectures. Furthermore, in all these cases, there is no flow passing through the same cut set more than once. If this is true for all *n* by *n* meshes, the improvement upper bounds we derived are exact throughput improvement rates.

3.2 Throughputs with symmetrical chip shape

As we have already seen in last subsection, a rectangle shaped chip has communication bottlenecks on theirs two middle cut lines. The physical dimension of the middle part of the chip restricts the communication flow and thus prevents us from getting larger throughput. As suggested in



(a) A level 2 hexagonal mesh (b) A level 2 octagonal mesh

(c) A level 2 Diamond mesh

Fig. 7 Meshes with symmetrical structures

[3], a convex shaped chip may produce better throughput by allowing more wires crossing the original middle cut lines. In this section, we set the chip shape to be close to a circle and symmetrical to all routing directions and then compare the throughput of different structures.

Fig. 7 shows an example of meshes symmetrical to the routing directions for different interconnect architecture. Fig.7(a) is a level 2 hexagonal mesh, which is the symmetrical structure corresponding to the Y-architecture. Fig. 7(b) illustrates an octagonal mesh, which is X-architecture's corresponding symmetrical mesh. Fig 7(c) shows a diamond shaped mesh, which is symmetrical to the Manhattan architecture.

We compute the throughput of symmetrical structures for the Y-architecture, X-architecture, and Manhattan architecture. Table 2 lists the throughput of Hexagonal meshes from level 1 to level 7. Table 3 shows the throughput of octagonal meshes from level 2 to level 4. Table 4 illustrates the throughputs of diamond meshes from level 1 to level 12. We also normalized the throughput by total edge capacities.

For Y-architecture, a hexagonal mesh with 169 mesh produces a normalized throughput of 2.82, which is 7.6% more than that of a square mesh using the same interconnect architecture. For X-architecture, our experiments have not reached the converged throughput yet. For largest case we have tested, which has 281 nodes, the normalized throughput is 2.84, which is 5.2% more than that of square mesh using the same interconnect architecture. For Manhattan architecture, a diamond shaped mesh with 265 nodes provides a normalized throughtput of 2.39. The throughput improvement of diamond mesh over square mesh for Manhattan architecture is 20%.

Table 2. Throughput of Hexagonal Meshes

Level	#nodes	throughput	Normalized throughput
1	7	2.00e-1	2.12
2	19	1.82e-2	2.49
3	37	1.24e-3	2.66
4	61	5.73e-3	2.77
5	91	2.45e-3	2.81
6	127	4.74e-4	2.81
7	169	1.39e-4	2.82

Table 3. Throughput of Octagonal Meshes

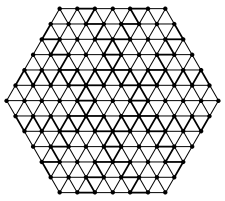
Level	#nodes	Throughput	Normalized throughput
2	29	2.31e-2	2.34
3	61	5.45e-3	2.51
4	101	3.01e-3	2.63
5	169	1.36e-3	2.74
6	281	5.75e-4	2.84

Table 4. Throughput of Diamond Meshes

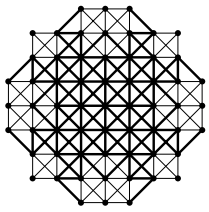
level	#nodes	Throughput	Normalized throughput
2	5	1.25e-1	1.78
3	13	4.20e-2	1.80
4	25	1.74e-2	2.09
5	41	8.71e-3	2.23
6	61	4.92e-3	2.30
7	85	3.00e-3	2.32
8	113	1.89e-3	2.36
9	145	1.39e-3	2.37
10	181	9.23e-4	2.38
11	221	6.90e-4	2.38
12	265	5.11e-4	2.39

The meshes with symmetrical structures produce different flow congestion pattern from n by n meshes. Fig. 8 illustrates the flow congestion patterns of a level 6 hexagonal mesh, a level 3 octagonal mesh and a level 8 diamond mesh. We mark the cut edges using red bold line. The symmetrical meshes displays a more evenly distributed congestion pattern than *n* by *n* meshes. The middle cut lines do not exist any more.

(a) flow bottleneck on a level 6 hexagonal mesh

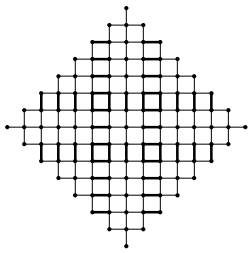


(b) flow bottleneck on a level 3 octagonal mesh



(c) flow bottleneck on a level 8 diamond mesh

Fig. 8 Flow congestions on meshes with symmetrical structures



VI. Wire-length of Different Interconnect Architectures

Wire length has significant impact on virtually every important measure of chip quality. From physical point of view, decreasing wire length directly reduces the resistance and capacitance of the interconnect, thus improves the performance and power consumption of the circuits. From designer's point of view, shorter total wire length can produces less routing congestion on the chip, hence improve the routability and signal integrity of the design. At the same time, from manufacturing point of view, shortening the wirelength can improve the manufacturability and reliability of the chip.

Because of its few freedom on choosing routing directions, Manhattan architecture adds significant amount of wire length to the Euclidean optimum. Apparently, allowing more routing directions may shorten the total wire length. Previously, researchers has studied the impact of using different interconnect architecture on the wire length. Many of these efforts were put on constructing the Steiner routing trees under different routing direction restriction [2] [8]. Due to the inherent hardness of Steiner minimum tree problem, most of these work develop heuristics to construct Steiner trees for a randomly generated net, and statistically calculate the average wire length for different interconnect architectures. In this section, we derive the quantitative comparison of wire lengths needed to connect a two pin net using different interconnect architectures.

In-order to generalize the non-rectilinear routing structure, Burman et al. [2] introduces the concepts of λ-geometry. In λ-geometry, wires with angles $i\pi/\lambda$, for all i are allowed, where λ is a positive integer. $\lambda = 2, 3, 4$ corresponds to the Manhattan architecture, Y-architecture, and X-architecture, respectively. In the following discussion, we follow this notation of λ-geometry.

Lemma1: In λ-geometry, given two points A and B, if AB are not on any of the λ feasible routing directions, then the shortest path connecting AB consists two segments AC and CB, where the angle between AC and CB is $(1-1/\lambda)\pi$

Lemma2: Let A, B be any two points on the plane, r_e be the Euclidean distance between A and B, and r_λ be the length of the shortest wire to connect AB in λ-geometry, then

$$\max_{A,B} \frac{r_e}{r_\lambda} = \csc\left(\left(\frac{\lambda-1}{2\lambda}\right)\pi\right)$$

Lemma3: Let A, B be two random points on the plan, r_e be the expected Euclidean distance between A and B, and r_λ be the expected length of the shortest wire to connect AB in λ-geometry, then $r_\lambda = \frac{2\lambda(1-\cos(\pi/\lambda))}{\pi \sin(\pi/\lambda)} r_e$

Lemma1 states that in order to connect two pins with the shortest wire, there is at most one turn on the path, and we need to maximize the angle between two segments of the path for the given interconnect `architecture`.

For different interconnect architectures, Lemma2 shows that in worst case how much additional wire length could we pay over the Euclidean distance. For Manhattan architecture, in worst case the wire length is 41.2% longer than the Euclidean distance. For Y-architecture and X-architecture, the additional wire length over Euclidean distance is at most 15.47% and 8.23% respectively. We list these numbers in the Table 5.

Lemma3 discuss the average wire length of a two pin net using different interconnect architecture. For Manhattan architecture, the average wire length is 27.32% longer than its Euclidean distance. For Y-architecture, the average wire length is 10.27% longer than its Euclidean distance and achieves a average wire length reduction of 13.4%. The X-architecture further reduces the average wire length to be within 5.48% of the Euclidean optimum and it produces 4.3% wire length reduction over Y-architecture with the cost of one more routing direction.

Table 5. Worst case wire length overhead of different interconnect architectures

M (%)	Y (%)	X (%)
41.2	15.47	8.23

Table 6. Average case wire length overhead of different interconnect architectures

M (%)	Y (%)	X (%)
27.32	10.27	5.48

V. Conclusions and Future Directions

In this paper, we propose a new on-chip interconnect architecture named Y-architecture. Comparing with the traditional Manhattan architecture, Y-architecture apparently improves the throughput (31.1% more for a 17 by 17 mesh) and significantly reduces the wire length. (an average reduction of 13.4% for a two pin net). Comparing with recently proposed X-architecture, Y-architecture produces very close throughput (only 2.6% less for a 17 by 17 mesh) and slightly longer wire length (Averagely 4.3% longer for a random two pins connection).

According to our experiments, making the chip shape close to a circle can improve the throughput of rectangular chip. A hexagonal chip using Y-architecture produces a throughput improvement of 41% over the rectangular chip using Manhattan architecture.

Some interesting research directions about Y-architecture

include: (1) Developing new models to take vias into consideration when evaluating different interconnect architectures, and (2) Designing a sample chip to justify our theoretical prediction.

VI. Acknowledgment

This work was supported in part under grants from NSF project number MIP-9987678, the California MICRO program, SRC support, and Cal-$(IT)^2$ graduate fellowship.

VII References

[1]International Technology Roadmpa for Semiconductors, 2001 Edition-Interconnect

[2]S.Burman, H. Chen, and N. Sherwani, "Improved global routing using λ-geometry," in Proc. of 29th Annual Allerton Conference on Communication, Computing, and Controls, Oct. 1991

[3]H. Chen, B. Yao, F. Zhou, and C. K. Cheng, "Physical Planning of On-Chip Interconnect Architectures," In Proc. of ICCD, pp.30-35, Sep. 2002

[4]C. Chiang and M. Sarrafzadeh, "Wirability of Knock-knee Layouts with 45-degree wires," IEEE Trans. on Circuits & Systems, vol. 38, No.6, June 1991, pp. 613-624

[5]N. Garg, and J. Konemann, "Faster and Simpler Algorithms for Multicommodity Flow and other Fractional Packing Problems," In Proc. Of the 39th Annual Symposium on Foundations of Computer Science, pp.300-309, 1998

[6]N. Karmarkar, "A new polynomial-time algorithm for linear programming," Combinatorica, 4(4):373--395, 1984

[7]I. Mutsunori, T. Mitsuhashi, A. Le, S. Kazi, Y. Lin, A. Fujimura, and S. Teig, "A Diagonal Interconnect Architecture and Its Application to RISC Core Design," Proc. ISSCC, pp. 684- 689. San Jose, CA, Feb. 2002.

[8]M. Sarrafzadeh, C.K. Wong, "Hierarchical Steiner tree construction in uniform orientations,". IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol.11, (no.9), Sept. 1992. p.1095-103

[9]F. Shahrokhi and D. Matula, "The maximum concurrent flow problem," J. ACM, 37(2): pp.318-334, 1990

[10] S. L. Teig, "The X Architecture: not your father's diagonal wiring," in Proc. of SLIP, pp.33-37, Apr. 2002