Layout Aware Design of Mesh based NoC Architectures [*]

Krishnan Srinivasan
Dept. of Computer Science and Engineering
Arizona State University
Tempe, AZ
ksrinivasan@asu.edu

Karam S. Chatha
Dept. of Computer Science and Engineering
Arizona State University
Tempe, AZ
kchatha@asu.edu

ABSTRACT

Design of System-on-Chip (SoC) with regular mesh based Network-on-Chip (NoC) consists of mapping processing cores to routers, and routing of the traffic traces on the topology such that power consumption is minimized, and performance constraints are satisfied. Technology scaling increases the contribution of the link power to the overall power consumption of the NoC. Since link power consumption is dependent on the length of the link, its contribution cannot be accurately estimated without system-level floorplanning. In this paper, we propose a novel design technique that integrates system-level floorplanning into the NoC design flow. Our technique invokes an existing floorplanner to generate an initial layout of the cores. This is followed by invocation of a novel low complexity algorithm that generates the mesh based NoC architecture with complete information of the floorplan. In comparison to an existing approach, our technique results in lower total power consumption and much lower link power consumption.

Categories and Subject Descriptors

B.4 [**Input/Output Data Communications**]: Interconnections

General Terms

Algorithm, Performance, Design

Keywords

Network-on-Chip, Automated design, Mesh topology

1. INTRODUCTION

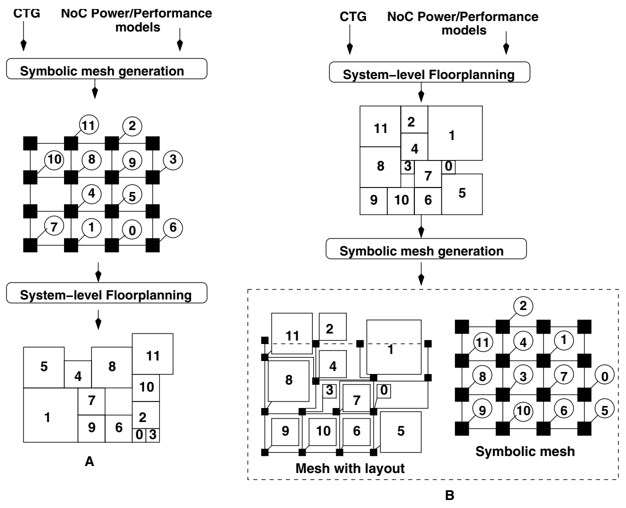

Figure 1: NoC design flows

NoC has been proposed by academia [1] [2] and industry [3] as the solution for the on-chip communication challenges of future SoC architectures. NoC is characterized by packet switching based communication mechanism that is enabled by on-chip routers. NoC architectures can be classified in to two categories on the basis on their topologies: custom or regular. Regular topologies, in particular the mesh topology, have emerged as popular architecture for NoC design due their simple structure, ease of implementation, and support for reuse.

The paper addresses the design of mesh based NoC architectures for application specific SoC. Examples of such SoC include high performance multimedia processors for set-top boxes. NoC design is initiated after the computation architecture design stage that specifies the computing and storage cores in the SoC. A core in the application specific SoC implements a limited functionality. Consequently, the inter-core communication depicts well defined patterns. Thus, the input to the NoC design problem can be specified as a communication trace graph (CTG) where the nodes denote the cores of the computation architectures, and the directed edges signify communication traces that are annotated with required bandwidth between the cores.

The optimization problem for mesh based NoC architectures consists of mapping the computation architecture cores to routers, and routing of the traces such that the total communication power is minimized, and bandwidth constraints are satisfied. The communication power is composed of the power consumption of the router elements, and the physi-

[*]The research presented in this paper was supported in part by grants from the National Science Foundation (CAREER 0546462, CSR-EHS 0509540) and Consortium for Embedded Systems.

cal links. The percentage of power consumption in physical links in a NoC has been found to be more than 30 % of the total power consumption in 100 nm technology [5]. This value will be higher with further technology scaling. Therefore, any NoC design technique must account for link power consumption in addition to the router power consumption. The power consumption in the physical links is directly dependent upon the length of the link and bandwidth of traffic flowing through the link, and can be estimated only after system-level floorplanning has been performed.

Figure 1 (A) presents the existing NoC design methodology that consists of generation of a symbolic mesh, followed by system-level floorplanning to generate the final SoC architecture. Since the symbolic mesh is generated without any knowledge of floorplanning, the final architecture can result in large link length and corresponding link power consumption. For the example shown in the figure, mapping and routing were performed by assuming a 4mm inter-router distance, and there was a deviation of about 50% in link power consumption, and 80% in wire length between the symbolic mesh, and the mesh after floorplanning. With further shrinking of technology, and increased contribution of link power to the overall power consumption, this deviation will result in a significant error between the estimated, and actual power consumption of the NoC.

This paper obtains a more accurate estimation of the link lengths by integrating system-level floorplanning in the NoC design flow. Our design methodology is depicted in Figure 1(B). Given the design specification in the form of a communication trace graph (CTG), our technique invokes an existing floorplanner with a cost function that minimizes the weighted sum of layout area and link power consumption. We then invoke a novel algorithm that generates the symbolic mesh from the given layout. Finally, our technique invokes a deadlock avoiding routing technique that routes traces along shortest paths subject to bandwidth constraints on the router ports. Since we have complete information about the floorplan, our technique accurately accounts for link power consumption. For the example shown in the figure, our technique resulted in 25% less power consumption, and 60% less wirelength compared to the existing method.

In the following sections, we describe and characterize the router architecture utilized in the paper, followed by a formal definition of the NoC design problem.

1.1 Router architecture and characterization

The NoC router architecture supported by our technique has been described in [5]. The router architecture consists of unit routers that communicate with each other through the crossbar, and with the neighboring routers. The router supports wormhole switching of packets. Each unit router consists of an input and output port for bi-directional data transfer. Custom routing is achieved by including a communication trace identifier (ID) in every packet header. On reception of the packet header, the header decoder decides the output port based on the ID, and requests the arbiter of the selected output port for access to the output FIFO. The arbiter controls the crossbar and enables the transfer of the packet from input to output FIFO. The output link controller transfers the packet from the output FIFO to the neighboring router.

The power consumption of both the input and output port of a NoC router is directly proportional to the bandwidth of traffic flowing through the ports [5]. Therefore, they can be characterized as the power consumed per unit bandwidth of traffic flowing through the ports (or $nW/Mbps$). The power consumption in the physical link is directly proportional to the bandwidth of traffic flowing through it, and its length. Therefore, the physical link can be characterized by power consumed per unit bandwidth of traffic per unit length (or $nW/Mbps/mm$).

1.2 Problem definition

Given:

- A directed communication trace graph $CTG(V, E)$, where each $v_i \in V$ denotes either a processing element or a memory unit (henceforth called a node), and the directed edge $e_k = \{v_i, v_j\} \in E$ denotes a communication trace from v_i to v_j. For every $v_i \in V$, the height and width of the core is denoted by $\mathcal{H}_i$ and $\mathcal{W}_i$, respectively. Every edge $e_k = \{v_i, v_j\} \in E$ has an associated $\omega(e_k)$ that denotes the bandwidth requirement of the communication in bits per second [1].
- Router architecture, where Ω denotes the peak input and output bandwidth that the router can support on any one port. Thus, each unit router can support equal bandwidth on input and output ports. Since a node is attached to a port of a router, the bandwidth to any node from a router, and from any node to a router is less than Ω. Two quantities Ψ_i and Ψ_o that denote the power consumed per $Mbps$ of traffic bandwidth flowing in the input and output direction, respectively for any port of the router.
- A physical link power model denoted by Ψ_l per $Mbps$ per mm.

Let $\mathcal{R}$ denote the set of routers utilized in the synthesized architecture, E_r represent the set of links between two routers, and E_v represent the set of links between routers and nodes. The objective of the NoC design problem is to generate a system-level floorplan, and a mesh based NoC $T(\mathcal{R}, V, E_r, E_v)$, such that:

- for every $e_k = (v_i, v_j) \in E$, there exists a route $p = \{(v_i, r_i), (r_i, r_j), \ldots (r_k, v_j)\}$ in T that satisfies $\omega(e_k)$,
- the bandwidth constraints on the ports of the routers are satisfied, and
- the total NoC power consumption for inter-core communication is minimized.

Once the system-level NoC architecture has been determined, global and detailed physical routing of the links is performed by invoking an existing algorithm that either performs channel routing, or over the cell routing. In Figure 1(B) the dotted lines between the routers denote over the cell routing.

The paper is organized as follows: in Section 2 we discuss previous work, in Section 3 we present our technique, in Section 4 we discuss our experimental results, and finally in Section 5 we conclude the paper.

2. PREVIOUS WORK

Many researchers [6] [7] [8] [9] have presented core mapping and routing techniques for regular mesh based NoC architectures. Guz et al. [10] presented a technique for

[1]In the case that there is a wide variation in the bandwidth requirements, the designer can specify either average or worst case communication traffic.

link capacity allocation for mesh based NoC architectures. Angiolini et al. [11] compared the power and performance of mesh based NoC architectures with AMBA bus architectures. Steenhof et al. [12] proposed NoC as a solution to high-end consumer electronics systems, and presented a SoC with mesh based NoC for a TV system architecture. Lee et al. [13] implemented a SoC with a hierarchical star based NoC topology. The above cited papers solve the NoC mapping problem by either neglecting floorplanning completely, [6] [7] [8] [10], or invoke the floorplanner after the NoC mapping problem has been addressed [11] [12] [13]. Due to the increased contribution of link power to the overall power consumption, addressing the NoC design problem without accounting for the link power gives a very poor estimate of the total power consumption of the interconnection architecture. We overcome this problem by integrating system-level floorplanning with the core to router mapping stage.

3. MESH BASED NOC DESIGN

Our technique integrates system-level floorplanning with the core to router mapping problem, thus enabling us to accurately account for link power consumption, as well as router power consumption. Our technique operates in three phases. In the first phase, we invoke an existing system-level floorplanner with a cost function that minimizes the overall link power consumption. Once the floorplan is obtained, we invoke a novel low complexity core to router mapping algorithm that generates the mesh based NoC. Finally, we invoke our deadlock avoiding routing algorithm to route traces by shortest paths subject to bandwidth constraints on the router ports.

3.1 System-level floorplanning

Floorplanning is a well researched problem, and we utilize an ILP based floorplanner [5] and an existing floorplanner called Parquet [14] to generate the layout. The floorplanner minimizes a cost function that is expressed as a linear combination of the power consumption due to the physical links, and the area of the layout [5]. The cost function is given by

$$\alpha \cdot \left[\sum_{\forall e(u,v) \in E} dist(u,v) \cdot \Psi_l \cdot \omega(e) \right] + \beta \cdot area$$

where $dist(u,v)$ is the Manhattan distance between the cores u and v, α and β are designer specified constants, and $area$ represents that total area of the layout. The output of the system-level floorplanning is a layout of the computation architecture cores.

3.2 Core to router mapping

We observe that the physical dimensions of the routers are much lower than the sizes of the cores. This assumption is supported by Dally et al. [2] who observed that the entire NoC places an area overhead of 6.6% on the SoC architecture. We also estimated the area consumption of the router architecture utilized in this paper. For a 5 port router, with 2 virtual channels, a FIFO depth of 8, and a width of 32, the router architecture consumed an area of 0.21 mm^2 in 130 nm technology. On the other hand, the areas of the computation architecture cores are of the order of several mm^2. Therefore, introducing the mesh based NoC in the layout, and mapping the cores to routers does not alter the floorplan significantly.

The objective of core to router mapping and corresponding NoC generation is to maintain the structure of the floorplan as far as possible. Since the floorplanner places highly communicating cores close to each other, maintaining the structure of the floorplan ensures that highly communicating cores are mapped to adjacent routers, thus minimizing router power consumption. Our technique invokes a novel low complexity algorithm that maps cores to routers row by row in the Y direction, and column by column in the X direction respectively, and thus generates the mesh based NoC.

Our technique generates the symbolic mesh based NoC by assuming that all cores are placed in the first quadrant ($X \geq 0, Y \geq 0$). We assume that in the logical mesh, two adjacent Y-coordinates differ by a unit distance. Similarly, two adjacent X-coordinates differ by a unit distance. Figures 2, 3, and 4 depict the different stages of our core to router mapping and NoC generation technique. Initially, the Y-coordinate of the X axis is set to 0. A core v_i is defined to be closest to the X axis if the following two conditions hold.

- No other core that overlaps v_i along its X-offset is closer to the X axis than v_i
- the distance of v_i from the X axis is less than the height of the largest sized core in the floorplan.

Our technique obtains all such cores that are closest to the X axis. In Figure 2(B) cores {9,10,6,5} form this set. The selected cores are removed from the floorplan, and their Y-coordinate in the symbolic floorplan is assigned to be the Y-coordinate of the X axis. The X axis, and the remaining floorplan is now shifted up by a unit distance in the Y direction. The procedure is repeated for the remaining cores, until all cores are assigned a Y-coordinate in the symbolic mesh.

In Figure 2, sub-figures (B), (C), (D) and (E) depict the successive stages of assigning Y-locations to the cores. As shown in Figures 3(A) and 3(B) a similar procedure is invoked to assign X-locations to the cores. Once the X and Y coordinates of the cores are determined, we have the symbolic mesh depicted in Figure 4(A). The actual mesh with floorplan is shown in Figure 4(B). The mesh is formed by placing the routers at the lower left hand corners of the cores, and interconnecting them in accordance with the symbolic mesh topology. Note that the global and detailed physical routing algorithm assumed in this paper allows over the cell routing (as signified by the dotted lines between the routers).

3.2.1 Algorithm

Figure 5 presents the algorithm for our floorplan to symbolic mesh generation. The algorithm invokes two recursive procedures *Get_Y_locations*, and *Get_X_locations* that calculate the relative Y and X locations of the cores in the symbolic mesh based NoC. The procedures take the floorplan of the cores, and the current Y location of the X axis (initially set to 0) as parameters. Line 1 through 3 check if there are any cores in the floorplan, and exit if the floorplan is empty. Line 6 obtains the cores that are closest to the X axis. Lines 7 through 9 update the Y locations of the cores in the set C. Lines 10 and 11 remove the cores from the current floorplan, shift the X axis up by one unit. Finally

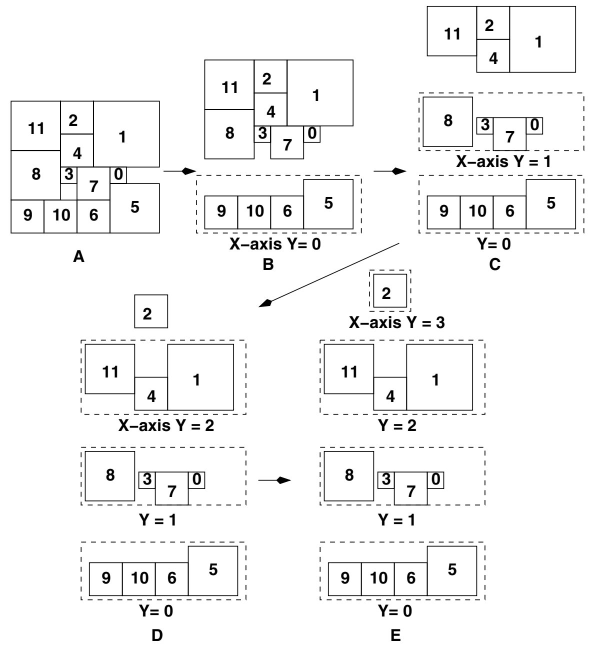

Figure 2: Mesh Y-coordinate determination

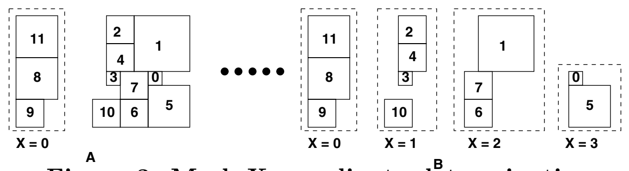

Figure 3: Mesh X-coordinate determination

line 12 invokes the procedure recursively with the truncated floorplan and shifted X axis. The *Get_X_locations* procedure is similar to the *Get_Y_locations* procedure. The only difference is that instead of the X axis, the Y axis shifts from left to right, and each call to the procedure determines the X location of a subset of the cores.

3.2.2 Complexity analysis

The *Get_X_locations* and the *Get_Y_locations* functions are called at most $O(|V|)$ times. If the nodes are sorted by their X and Y co-ordinates respectively, determining the set of cores in each iteration of the function takes $O(|V|)$ time. Hence, the overall complexity of the technique is $O(|V|^2)$.

3.3 Routing

After the mapping phase, we calculate the power consumption due to the traces assuming that they are routed through the shortest path, and sort them in the order of decreasing power consumption. Let *sorted_list* denote the list of traces in the sorted order. Our routing technique operates on the traces in *sorted_list* as follows.

- It routes the traces in the list by invoking the X-Y routing technique. The X-Y technique routes all traces along the X direction until X coordinate of the sink core is reached, and then routes the trace along the Y direction, until the sink core is reached. X-Y routing is easy to implement, has an O(1) complexity, and is known to be deadlock free. However, certain traces may not be routed due to bandwidth violations on router ports.
- Our technique adds all such un-routed traces to a list called *tbd_trace_list*. A modified shortest path (MSP) algorithm is invoked to determine alternative routes for these traces.

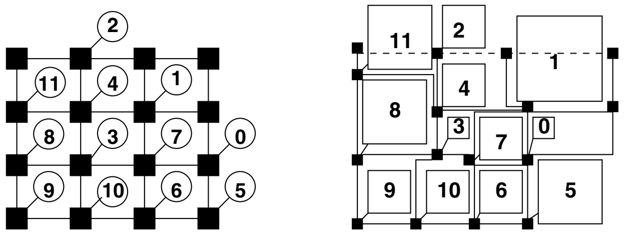

Figure 4: Symbolic and floorplanned mesh

```
Generate_Mesh(floorplan F)
1   Initialize X = 0, Y = 0
2   Get_Y_locations(F, X)
3   Get_X_locations(F, Y)
end Generate_mesh
Get_Y_locations(floorplan F, int X)
1   if F = NULL
2       return
3   end if
4   $F_{cur} = F$ /* copy Floorplan to local data structure */
5   $X_{cur} = X$ /* copy X axis local data structure */
6   $C$ = get_lowest_core_set(X) /* get cores closest to X axis */
7   for c ∈ C /* each core in core set C */
8       Yloc (c) = X
9   end for
10  $F_{cur} = F_{cur} - C$ /* remove cores C from current floorplan */
11  $X_{cur} = X_{cur} + 1$ /* shift the X axis up */
12  Get_Y_locations($F_{cur}, X_{cur}$) /*recursive call*/
end
```

Figure 5: Generating mesh from floorplan

3.3.1 MSP Routing

The MSP router is called for each traffic trace that is left unmapped at the end of the X-Y routing stage. It attempts to achieve the following objectives in decreasing order of priority.

- *Routes for the traces must satisfy bandwidth constraints.* It makes sure that traces are not routed through ports that see a bandwidth violation.
- *The routes should minimize power consumption.* It invokes Dijkstra's shortest path algorithm to route traces such that the number of hops, and corresponding power consumption is minimized.
- *The routes should avoid deadlocks.* For deadlock avoidance, we first define the channel dependency graph (CDG) for the mesh. Given a mesh $G(N, L)$ where N denotes the set of routers, and L denotes the set of physical links, and a routing function R, the CDG is a directed graph $G'(V', E')$, where each edge $l \in L$ has a corresponding unique node $v \in V'$, and there is an edge $e' \in E'$ for two adjacent links in $\{l1, l2\} \in L$ such that some trace is routed through $l1$ followed by $l2$. A routing technique is deadlock free if the CDG does not contain cycles [2]. While routing traces, our algorithm finds shortest paths by avoiding turns that cause a cycle in the graph and therefore, avoids deadlocks. If deadlocks cannot be avoided, virtual channels are introduced to break them.

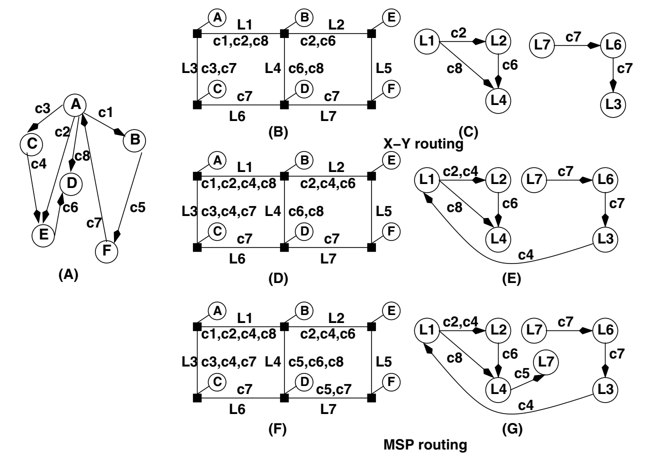

Figure 6: X-Y and MSP routing

```
MSP(R, $G(N, L)$, tbd_trace_list)
1   $G'$ = obtain_CDG(R, $G(N, L)$)
2   for t $\in$ tbd_trace_list
3       for e $\in L$ /* For all physical links in $I$ */
4           if ($\omega(e) + \omega(t) > \Omega$) /* BW violation */
5               edge_weight(e) = $\infty$ else edge_weight(e) = 1
6           end if
7       end for
8       $S$ = prevent_turns($G'$)
9       shortest_path($t, I, \mathcal{R}, S$)
10      update_CDG($G'$, $G$)
11  end for
```

Figure 7: Bandwidth constrained route generation

The MSP algorithm is illustrated in Figure 6. In the figure, the traces are denoted by cm, where m is the trace number, and the links are denoted by Ln, where n is the link number. The MSP initially generates the CDG of the mesh based on the traces that are routed by X-Y technique. For the CTG shown in 6(A), 6(B) depicts the core-router mapping, and the trace routes. The corresponding CDG is depicted in 6(C). The edges of the CDG are annotated by the traces that induce them. Traces $c4$ and $c5$ are left unmapped as they violate the bandwidth constraints. Hence, they are added to the *tbd_trace_list*.

For each trace in *tbd_trace_list*, MSP sweeps the links L of the mesh, and assigns an edge weight of ∞ to all links that would see a bandwidth violation on the corresponding ports constituting the links, if the trace was routed through that link. The algorithm then inspects the CDG, and prohibits turns that will cause a cycle in the graph. This step is followed by invoking the Dijkstra's shortest path algorithm to find a route for the trace on the mesh, subject to the prohibited turns. Figure 6(D) depicts the routing of trace $c4$ by invoking the shortest path algorithm. The updated CDG is shown in Figure 6(E).

Since power minimization has a higher priority over deadlock avoidance, we allow a deadlock creating turn if any other turn results in a path that is longer than the shortest path route for the trace. In case a deadlock creating turn is taken, an additional virtual channel is assigned to that port to remove the deadlock. In the figure, trace $c5$ is routed such that it creates a deadlock at the router to which core D is mapped. Hence, a virtual channel is introduced, denoted by two $L7$ nodes in the CDG of Figure 6(G).

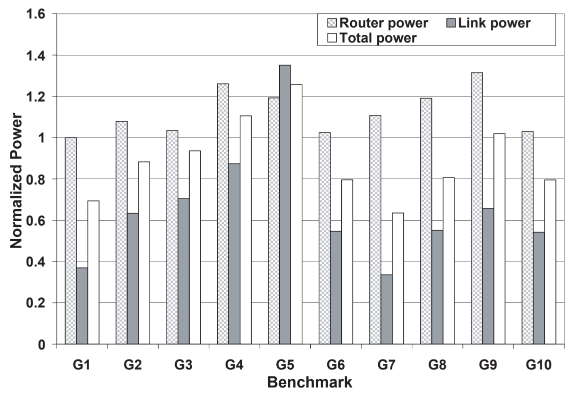

Figure 8: Power comparisons

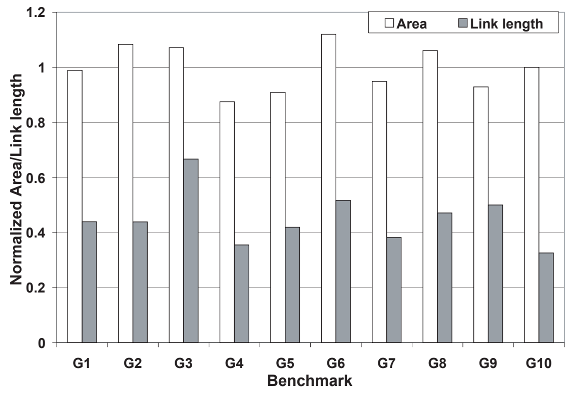

Figure 9: Area and link length comparisons

3.3.1.1 Algorithm.

Figure 7 details the MSP algorithm. Line 1 generates the CDG for the mesh corresponding to the traces that have already been routed. For each trace in the list, lines 2 through 7 set the edge weight of all the links to infinity that result in bandwidth violations if the trace is routed through that link. Line 8 generates a list S of turns that have to be avoided to ensure deadlock freedom. Line 9 invokes Dijkstra's shortest path algorithm to find an alternative route for the trace, subject to the edge weights, and the prohibited turns. Finally, line 10 updates the CDG.

3.3.1.2 Complexity analysis.

The complexity of the MSP algorithm is dominated by Dijkstra's shortest path algorithm for each trace, and has a complexity of $O(E^2)$. Although the worst case complexity of the routing technique is that of the MSP, the actual complexity is much lower as MSP is invoked only on traces that violate bandwidth constraints on the router ports.

4. RESULTS

We present experimental results obtained by execution of our technique on multimedia applications obtained from Hu et al. [7], and Jalabert et al. [15] (see table 1).

We compared our results with the existing approach that optimally generates the symbolic mesh based NoC by invoking an ILP formulation, and then invokes an existing floorplanner (Parquet [14]) to generate the final layout. Figures 8 compares the router power consumption, link power consumption, and total power consumption respectively, between our technique and the existing approach. In the fig-

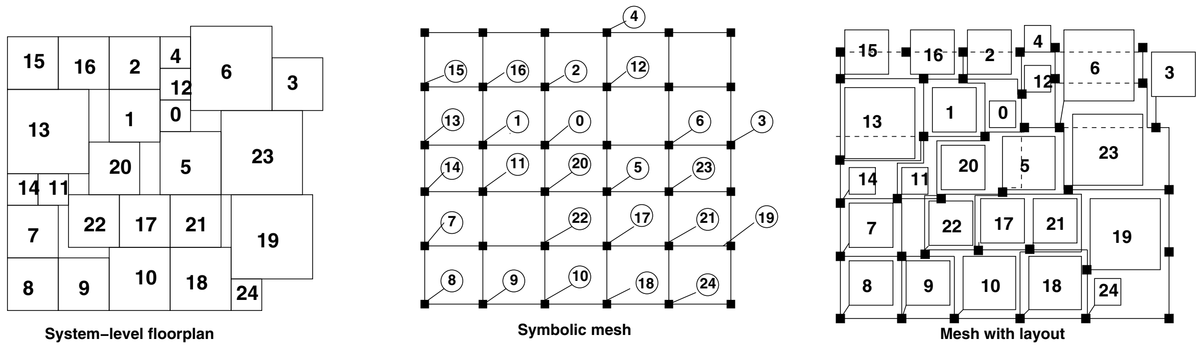

Figure 10: Results for 25 node set top box

Benchmark	$\lvert V\rvert$	$\lvert E\rvert$	Benchmark	$\lvert V\rvert$	$\lvert E\rvert$
G1: mp3 decoder	5	3	G6: MWD	12	13
G2: 263 encoder	7	8	G7: 263 enc mp3 dec	14	12
G3: mp3 encoder	8	8	G8: mp3 enc mp3 dec	15	12
G4: 263 decoder	8	9	G9: 263 enc mp3 enc	15	17
G5: MPEG4	12	13	G10: Set-top box	25	40

Table 1: Benchmarks

ID	Core type	Functionality	ID	Core type	Functionality
0	ASIC	ME	13	MEM	Buf.
1	DSP	DCT and IDCT	14	ASIC	Demux
2	DSP	Qnt & IQnt	15	DSP	VLD
3	DSP	FP	16	DSP	IQ
4	ASIC	VLE	17	DSP	IDCT
5	CPU	MC, and ADD	18	CPU	MC/Add
6	MEM	FS0, FS1, FS2	19	MEM	FS4, FS5
7	DSP	FP	20	DSP	Huff.dec.
8	DSP	FFT, PA model	21	DSP	Bitres.
9	DSP	Filter, MDCT	22	DSP	IMDCT/sum
10	CPU	It.enc.	23	MEM	Buf.
11	ASIC	Bitres.	24	ASIC	Synch.
12	ASIC	Synch			

Table 2: Node descriptions

ure, the bars are normalized to the corresponding value of the solution produced by the existing approach. On an average our technique consumed 0.69 times the link power, 1.12 times the router power, and 0.89 times the total power, compared to the existing approach. As technology shrinks further, the contribution of link power to the total power consumption will increase and consequently, the total power consumption of the designs generated by our technique will reduce further.

Figure 9 compares the area consumption of the SoC and the total link length respectively, between our technique and the existing approach. In this figure as well, the bars are normalized to the corresponding solutions produced by the existing approach. The designs produced by our technique on average consume 0.99 the area and 0.45 times the link lengths compared to the existing approach. Thus, our technique generates designs with much lower physical link lengths with minimal impact on the area.

We applied our technique to a set-top box application that we obtained from the work presented by Hu et al. [7]. We refer the reader to [7] for complete description of the CTG. Table 2 presents the identifier, type, and the functionality of the cores of the CTG. Figure 10 presents the floorplan, symbolic mesh based topology, and the actual mesh based architecture for the application. In the figure, the dotted lines in part (C) refer to over the cell routing of the wires.

5. CONCLUSION

We presented a novel mesh based NoC design technique that effectively accounts for the increased link power consumption that is observed in nanoscale technologies. The technique accounts for the link power consumption by integrating the system-level layout in to the design process. The SoC floorplan is generated by optimizing a weighted sum of the area and link power consumption. The core to router mapping stage of the technique operates on the system-level layout of the SoC and attempts to maintain the overall structure of the floorplan. Thus, the mesh design is obtained with an awareness of the SoC layout. The experimental results demonstrate that in comparison to an approach that first generates the NoC and then obtains the SoC floorplan, our technique produces solutions with much shorter link lengths, dramatically lower link power consumption, lower overall power consumption and minimal impact on area. The benefits of our technique are expected to increase with technology scaling.

6. REFERENCES

[1] Benini et al. "Networks on Chips: A New SoC Paradigm". *IEEE Computer*, pages 70–78, January 2002.

[2] Dally et al. "Route Packet, Not Wires: On-Chip Interconnection Networks". In *DAC*, June 2002.

[3] STMicroelectronics Inc. http://www.st.com/stonline/press/news/year2005/t1741t.htm, December 2005.

[4] N. Banerjee, P. Vellanki, and K. S. Chatha . "A Power and Performance Model for Network-on-Chip Architectures ". In *DATE*, Paris, France, February 2004.

[5] Srinivasan et al. . "Linear Programming based Techniques for Synthesis of Network-on-Chip Architectures" . *IEEE Transactions on VLSI Systems*, 14(4):407–420, 2006.

[6] Murali et al.. "Bandwidth-Constrained Mapping of Cores onto NoC Architectures". In *DATE*, 2004.

[7] Hu et al.. "Energy-Aware Mapping for Tile-based NoC Architectures Under Performance Constraints". In *ASP-DAC*, 2003.

[8] Manolache et al.. "Fault and Energy-Aware Communication Mapping with Guaranteed Latency for Applications Implemented on NoC". In *DAC*, 2005.

[9] Srinivasan et al.. "A Technique for Low Energy Mapping and Routing in Network-on-Chip Architectures". In *ISLPED*, 2005.

[10] Zvika Guz et al. "Efficient Link Capacity and QoS Design for Network-on-Chip". In *DATE*, March 2006.

[11] Angiolini et al. "Contrasting NoC and a Traditional Interconnect Fabric with Layout Awareness". In *DATE*, March 2006.

[12] Steenhof et al. "Network-on-Chip for High-End Consumer Electronics TV System Architectures ". In *DATE*, March 2006.

[13] Kangmin Lee et al. . "Low Power Network-on-Chip for High Performance SoC Design ". *IEEE Transactions on VLSI Systems*, 14(2):148–160, 2006.

[14] Adya et al.. "Fixed Outline Floorplanning: Enabling Hierarchical Design". *IEEE Transactions on VLSI Systems*, 11(6):1120–1135, December 2003.

[15] Jalabert et al.. "xpipesCompiler: A tool for instantiating application specific Networks on Chip". In *DATE*, 2004.