A Low Complexity Heuristic for Design of Custom Network-on-Chip Architectures *

Krishnan Srinivasan, and Karam S. Chatha
Department of CSE, PO BOX 878809, Arizona State University, Tempe, AZ 85287-8809
Email: {ksrinivasan,kchatha}@asu.edu

ABSTRACT

Network-on-Chip (NoC) has been proposed to replace traditional bus based architectures to address the global communication challenges in nanoscale technologies. In future SoC architectures, minimizing power consumption will continue to be an important design goal. In this paper, we present a novel heuristic technique consisting of system-level physical design, and interconnection network generation that generates custom low power NoC architectures for application specific SoC. We demonstrate the quality of the solutions produced by our technique by experimentation with many benchmarks. Our technique has a low computational complexity, and consumes only 1.25 times the power consumption, and 0.85 times the number of router resources compared to an optimal MILP based technique [1] whose computational complexity is not bounded.

1. INTRODUCTION

The high performance requirements of future System-on-Chip (SoC) designs, and long signal propagation delays in nanoscale technologies will make traditional synchronous bus based communication unattractive. Network-on-chip (NoC) has been proposed as an alternative to bus based communication in nanoscale technologies [2] [3]. In simple terms, a NoC is a set of interconnected routers that communicate in globally asynchronous and locally synchronous (GALS) manner. NoC architectures are scalable, provide high bandwidth by distribution of signal delay among routers, and support concurrent communication. These characteristics make them particularly attractive as a solution to global communication challenges in deep submicron technologies. Power reduction will continue to be a first order design goal in nanoscale technologies. Hence, a NoC design framework must aim to minimize the power consumption of the architecture, subject to performance constraints.

A NoC designer has to choose between utilizing a regular NoC architecture like a mesh or torus, or a custom architecture optimized for a given application. Regular architectures offer lower design time, and are useful when implemented in a generic multiprocessor environment such as the MIT RAW [4]. On the other hand, custom NoC architectures are designed with the given application in mind. For example, a NoC may be optimized for the MPEG-4 decoder application. Application specific architectures have been demonstrated to be superior to regular architectures in terms of power, performance and area [1] [5]. In this work, we focus on design of low power custom NoC architectures optimized for application specific SoC.

Figure 1 plots the power consumption of various components of the NoC for two technologies (100, and 180 nm) obtained by utilizing the NoC power and performance evaluator proposed by Banerjee et al. [6]. In the figure, the X axis denotes the various components, and the Y axis denotes the corresponding dynamic power consumption. At $180nm$, the contribution of link power is about 25% of the total communication power. As technology shrinks further to less than $100nm$, the contribution of link power to the overall communication power increases. For example, at $100nm$, the physical link consumes more than 30% of the total communication power. The power consumption of the physical links is dependent on its length, and the bandwidth of traffic flowing through it. The length of the link in turn is determined by the system-level floorplan. Therefore, the design of the NoC architecture must jointly address system-level floorplan and interconnection network generation. In this paper, we present a novel integrated technique that addresses system-level floorplanning, as well as application specific interconnection topology generation.

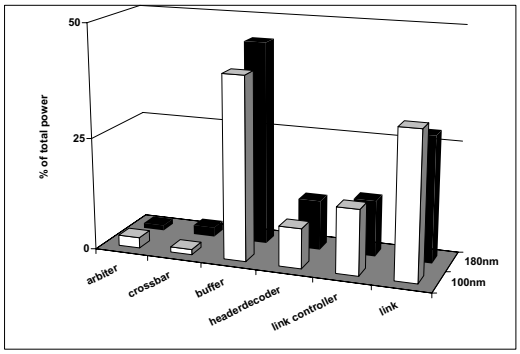

Figure 1: Component power consumption in NoC

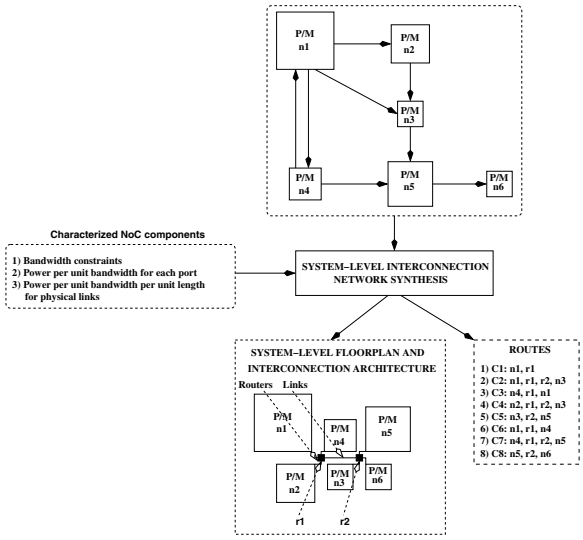

Figure 2: System-level Communication Architecture Design

Application specific NoC architecture design is shown in Figure 2 [1]. The input to the NoC architecture design problem is the computation architecture specification, characterized library of in-

*The research presented in this paper was supported in part by a grant from the National Science Foundation (IIS-0308268) and Consortium for Embedded Systems.

terconnection network components, and performance constraints. The computation architecture consists of processing and memory elements shown by rectangular blocks labeled as "P/M" in the top of the figure. Each "P/M" block is uniquely identified by a node number "n *i*" as denoted within each rectangle. The physical dimensions of the blocks are also specified as part of the inputs.

The directed edges between any two blocks represent the communication traces. They are annotated as "C*m*(B,L)" where '*m*" represents the trace number, "B" represents the bandwidth requirement, and "L" is the average latency constraint. The bandwidth and latency requirements of a communication trace can be obtained by profiling the system-level specification in the context of overall application performance requirements. The traffic model that we assume for synthesis abstracts away the transaction based mechanism to derive a continuous stream model. Burstiness of traffic can be captured by initially allocating a higher bandwidth requirement on the traffic [7]. Applications in multimedia and network processing domains demonstrate fairly well defined communication characteristics and hence can be easily modeled in the trace graph.

The left hand side of the figure depicts the characterized library of NoC components. We characterized the power consumption of the unit router in 100 nm technology with the help of a cycle accurate power and performance evaluator [6]. In the interest of space, we have omitted the complete details of the experiments. The power consumption of the input and output ports varies linearly with the injection rate [1]. We estimated the power consumption of the router ports to be $328nW/Mbps$ for the input port, and $65.5nW/Mbps$ for the output port. The power consumption of the physical links varies linearly with both the supported bandwidth, and the length of the link. We estimated the power consumption of the links to be equal to $79.6nW/Mbps/mm$ [1].

The output of the NoC architecture design problem is a system-level floorplan of the final design, topology of the network, and static routing of the communication traces on the network such that the performance constraints are satisfied, and the power consumption is minimized. Accurate estimation of power consumption due to the physical links requires estimates for link lengths. Consequently, system-level floorplanning is performed as part of the communication architecture design. The topology of the network specifies the number of routers, and their interconnections. The static routing of a communication trace is shown on the right hand side of the figure. For example , C2 begins from "n1", passes through "r1" and "r2", and ends at "n3".

It has been shown that the average network latency remains constant as long as the network is not congested [6]. Our design technique prevents network congestion by static routing of the communication traces subject to the peak bandwidth constraint on the router ports. Since the network is always operated in the un-congested mode, we can represent the average network latency constraint in terms of router hops (such as 1 or 2) instead of an absolute number (such as 100 cycles). In the following section, we formally define the NoC synthesis problem.

1.1 Problem definition

Given [1]:

- A directed communication trace graph (CTG) $G(V, E)$, 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.
- For every $e_k = \{v_i, v_j\} \in E$, $\omega(e_k)$ denotes the bandwidth requirement in bits per cycle, and $\sigma(e_k)$ denotes the latency constraint in hops.

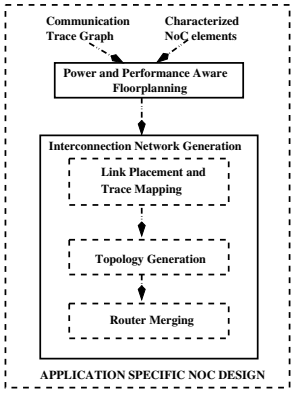

Figure 3: NoC design flow

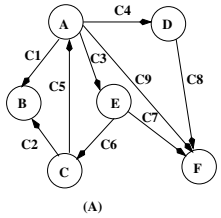

Figure 4: CTG

- A router architecture, where Ω denotes the peak input and output bandwidth that the router can support on any one port. 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 network topology $T(\mathcal{R}, V, E_r, E_v)$, such that: i) 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 $\omega(e_k)$, and $\sigma(e_k)$, ii) the bandwidth constraints on the ports of the routers are satisfied,, and iii) the total system-level power consumption for inter-core communication is minimized.

2. PREVIOUS WORK

Many researchers [8] [9] [10] have presented mapping and routing techniques for regular mesh based NoC architectures. Recently, researchers have begun to address the problem of automated synthesis of application specific NoC architectures. Pinto et al. [11] presented a technique for constraint driven communication architecture synthesis of point to point links by utilizing deterministic heuristic based k-way merging. Their technique results in network topologies that have only two routers between each source and sink. Hence, their problem formulation does not address routing. In our previous work, we have proposed custom NoC design techniques for best effort traffic [12] [13], and guaranteed throughput [14]. Umit et al. [15] presented an application specific NoC synthesis technique based on graph decomposition. The above cited papers do not accurately account for link power. In contrast, we integrate floorplanning in the NoC design flow to account for link power. Our work in [16] presents a linear programming based technique for the NoC design problem. The computational complexity of the technique presented in this paper is much lower than [16].

3. DESIGN OF CUSTOM NOC ARCHITECTURES

In this section, we present our application specific NoC design technique. The overall technique is shown in Figure 3. It takes a communication trace graph, and a characterized library of NoC components as input. Our technique operates in two phases. In the first phase, it invokes a power and performance aware floorplanning technique to obtain an initial physical layout of the nodes constituting the SoC. The second phase is a novel technique that generates a

low power interconnection architecture based on the floorplan from the first phase. In the following sections, we discuss each phase in detail.

3.1 System-level floorplanning

The objective of system-level floorplanning is to generate a layout that would minimize the power consumption of the interconnection architecture. We utilize two existing system level floorplanning techniques, and invoke them with our unique cost function to minimize communication power under performance constraints.

At the floorplanning stage the power consumption due to the interconnection architecture can be abstracted as the power required to perform communication via point to point physical links between communicating cores [1]. Since the power consumption of the NoC is directly proportional to the bandwidth of traffic, it is desirable to place cores communicating with high bandwidth close to each other, so that they are routed with minimum number of routers. At the same time, it is also necessary to place cores communicating with tight latency constraints close to each other to satisfy the performance constraints. Bandwidth requirements on the communication traces can be satisfied by finding alternative routes or adding more interconnection architecture resources. However, satisfying latency constraints is more difficult if the cores are placed wide apart. Hence, the cost function should give a higher priority to latency over bandwidth. We specify our minimization goal as a power aware cost function represented by $\sum_{\forall e(u,v) \in E} dist(u,v) \cdot \Psi_l \cdot \frac{\omega(e)}{\sigma^2(e)}$, where $dist(u,v)$ is the distance between the cores u and v.

We invoke two system level floorplanning techniques that minimize the above mentioned cost function. First, we invoke an existing mixed integer linear programming (MILP) formulation that minimizes the given cost function. MILP formulations for system level floorplanning have been proposed in the literature [17] [18] [19].

The second floorplanning technique that we invoke is based on a slicing tree based recursive partitioning technique proposed in [16].The slicing tree is formed by recursively partitioning the layout area into vertical and horizontal sections. At each partition, the floorplanner invokes a graph equicut algorithm proposed by Fiduccia and Mattheyses (FM) [20] to generate the partitions. The partitioning technique assigns nodes to one of the sub-planes such that the total weight of the edges across the cut is minimized. Finally, the technique incorporates a compaction algorithm that moves the nodes toward the center of the floorplan by iterative displacement along X and Y axis, respectively.

3.2 Interconnection network generation

In this section, we present the topology generation and routing algorithm employed by our communication architecture design technique. For explanation purposes, we utilize the CTG and the corresponding floorplan depicted in Figures 4 and 6(a), respectively. Our topology and route generating algorithm takes a SoC floorplan as input, and generates an interconnection network that minimizes power consumption subject to bandwidth constraints. Latency constraints are addressed by the following observations. First, since the floorplanner assigns a higher weight to latency than bandwidth, traces with tight latency constraints are placed close to each other, and hence, routed through minimum number of router hops. Second, the traces are routed through minimum number of hops, subject to bandwidth constraints.

Our technique operates in three stages. In the first stage, it recursively bisects the channel intersection graph (CIG) of the given floorplan to determine physical links, and maps traces on the links.

```
ANOC (CIG, CTG)
1   map_traces(CIG, V)
2   generate_topology()
end

generate_topology()
1   for e ∈ E
2     get_route(E)
3   end for
4   for core c ∈ V
5     assign_router(c)
6   end for
7   for node n ∈ CIG
8     if (merge_diverge(n))
9       place_router(n)
10    end if
11  end for
12  remove_redundant_routers()
end

map_traces(cig, dir)
1   if |cig| == 1)
2     return
3   end if
4   {lg, rg} = get_cut(cig, dir)
5   L = get_links(cig, dir)
6   C = get_traces(cig,dir)
7   for tr ∈ C
8     get_bb(tr)
9     l = map_link(tr, L)
10    update_locations(tr, l)
11  end for
12  if (dir == V) cd = H
13  else cd = V
14  end if
15  map_traces(lg,cd)
16  map_traces(rg, cd)
end
```

Figure 5: Interconnection network generation

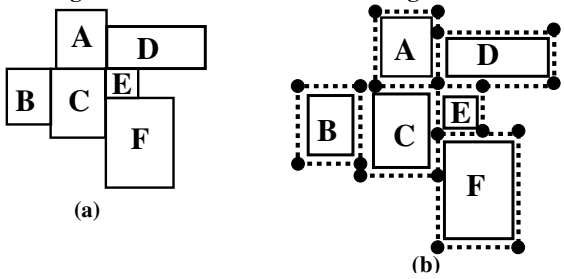

Figure 6: Floorplan and channel intersection graph

In the second stage, it generates a NoC topology by incorporating routers at the nodes of the CIG of the floorplan. Finally, the technique incorporates a router merging stage to further minimize the power consumption and number of router resources in the architecture. The application specific NoC (ANOC) design algorithm is shown in Figure 5.

3.2.1 Link placement and trace mapping

Our technique generates an interconnection network by recursively bisecting the CIG [17] of the given floorplan. To define a CIG, we introduce the notion of a bounding box. A bounding box is a rectangular enclosure of the core such that the bounding boxes of two adjacent cores abut each other. A CIG is defined on the bounding boxes, and is defined as a graph in which the boundaries of the bounding boxes form the edges, and the intersection of two perpendicular boundaries forms a node. Figure 6(b) depicts the CIG of the floorplan depicted in 6(a). The dotted lines denote the edges of the graph, and the dark circles denote the nodes. The edges of the CIG denote the physical links [17] on which traffic traces are routed, and the nodes represent possible locations for router placement.

We define a vertical cut as a line from top to bottom of the floorplan. Similarly, we define a horizontal cut as a line from the left to right of the graph. Initially, a vertical cut divides the graph into left and right sub-graphs of almost equal span in the X axis. All traffic traces crossing the cut are mapped on the links that are intersected by the cut. For each of the sub-graphs thus formed, a horizontal cut divides it into top and bottom sub-graphs of almost equal span in the Y axis. As before, all traffic traces crossing the cut are mapped on the links that intersect the cut. The horizontal and vertical cuts are recursively repeated until each traffic trace is mapped to a link incident on the respective nodes of the CIG that define the bounding box of the source and sink cores.

The actual mapping of traffic traces to links is a bin packing

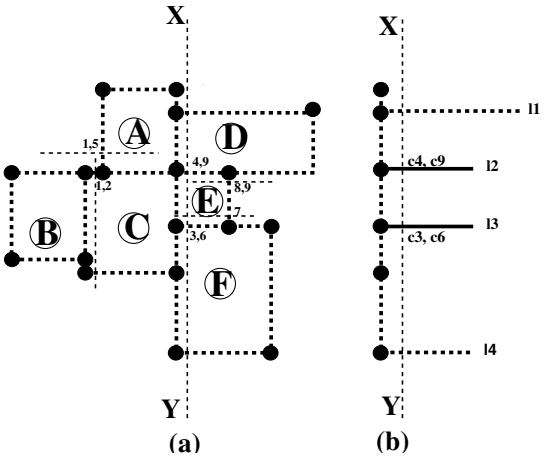

Figure 7: Link placement and trace mapping

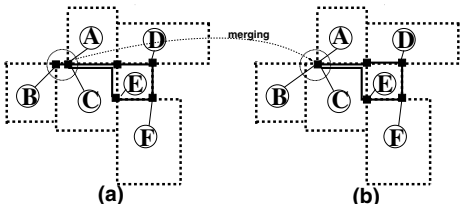

Figure 8: Topology generation

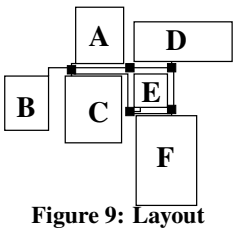

Figure 9: Layout

problem, which is known to be NP Complete. Hence, we invoke a heuristic algorithm to map traces to the links. For each traffic trace crossing the cut, the technique selects a link that minimizes power. Figure 7(a) shows the cuts, links and the corresponding traces mapped on the links. Figure 7(b) shows the cross-section for the first cut denoted as X-Y. As shown in the figure, traces C3, C4, C6, and C9 are mapped on the links that are shown by dark lines. The links that do not map any traces are shown by dotted lines.

Link placement and trace mapping is performed by function called map_traces of algorithm $ANOC$ shown in Figure 5. The function takes the CIG, and the direction of cut dir as input. The function returns if the number of nodes in the CIG is 1, denoting that no cuts are required. Line 4 determines the left and right sub-graphs by bisecting the CIG in the direction given by dir. Line 5 determines the physical links that intersect the cut and adds them to a list L. Line 6 determines the crossing traces. Lines 7 through 11 map the crossing traffic traces to the physical links. Lines 12 through 14 set the direction of cut of the sub-graphs to be complementary to the current direction of cut. Lines 15 and 16 recursively invoke the function for each of the sub-graphs.

Initially, the algorithm sets the variables $cur\_loc(tr, tr.src)$, and $cur\_loc(tr, tr.sink)$ for each traffic trace $tr \in E$ to be the respective locations on source and sink cores in the layout, such that the two points define the bounding box (BB) containing the cores. The routing of the traffic trace is performed by utilizing links within BB. The algorithm selects traffic traces one by one, and performs the following steps. It maps the trace to a link in BB such that i) the sum of the shortest distance from the source to the link, and the link to the sink, is minimized, and, ii) the bandwidth constraint on the link is not violated. Without loss of generality, assume that the cut is vertical, and the source lies in the left partition. After mapping the trace on a link, the algorithm updates $cur\_loc(tr, tr.src)$ for the right partition, and $cur\_loc(tr, tr.sink)$ for the left partition to be the co-ordinates of the point of intersection of the link that maps the trace, and the cut. The updated locations are utilized to determine the bounding box of the trace in subsequent cuts. For example, for the cut X-Y, trace c9 is mapped on link l2. For the subsequent horizontal cut that has c9 as a crossing traffic trace, the bounding box is generated with the first end being the intersection of l2 and X-Y, and the second end being node F. The algorithm recursively performs the link placement and traffic trace mapping, and thus establishes a route for all traffic traces.

3.2.2 Topology generation

The mapping of traces on links by recursive partitioning of the floorplan generates a route for each traffic trace. A route can easily be determined by starting from the source node, and following the links that map the trace to the sink node.

Once the routes are determined, the technique proceeds to generate the interconnection network topology. Intuitively, if a node of the CIG is a point of merging or diverging of three or more edges with some traffic flow, a router is required at the location of the node. We assume that the dimensions of the routers are much lower than the sizes of the cores. This assumption is supported by the observation of Dally et al. [2] that the entire NoC places an area overhead of 6.6% on the SoC architecture. Therefore, the floorplan before and after the introduction of routers does not vary by a significant amount. Hence, the technique considers the nodes of the CIG as possible locations for the routers. The introduction of routers in this manner generates the NoC topology, as discussed in the next paragraph.

For each processing core, the technique inspects the closest links that carry traffic traces with the processing core as a source or sink, and places a router at the location of the node of the CIG on which the link containing maximum bandwidth flow is incident. The processing core is connected to the router, and all traces that has the processing core as a source or sink, are routed through that router. The technique then proceeds to place a router at all locations of the nodes in the CIG that are merging or diverging points for 3 or more links with some traffic flow. Figure 8(a) shows the topology generated by our technique. Note that the nodes of the CIG have been replaced by black squares, denoting routers.

3.2.3 Router merging

In the final stage, we incorporate a merging technique to further minimize the power consumption, and reduce the number of routers in the topology. The technique examines pairs of adjacent routers, and merges them if they are i) close to each other, ii) merging them does not violate bandwidth constraints, and iii) the total power consumption is minimized. The minimum distance between two routers is determined by the maximum link length that can support single clock cycle data transfer between the routers. This distance is specified by the designer. If two routers are further than the specified maximum distance, they are not merged. Merging is performed by collapsing one router into another. The router with higher bandwidth flow is retained, and the one with lower bandwidth flow is collapsed. Figure 8(b) presents the topology after merging. In the figure, the two routers close to node B are merged into a single router. Figure 9 presents the final floorplan and interconnection network architecture for the given CTG.

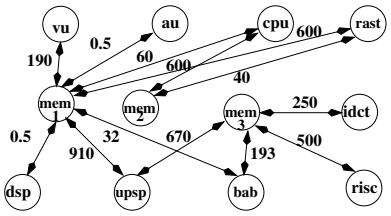

Figure 10: MPEG4 CTG

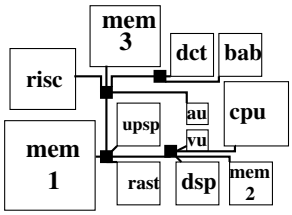

Figure 11: MS_m MPEG4 topology

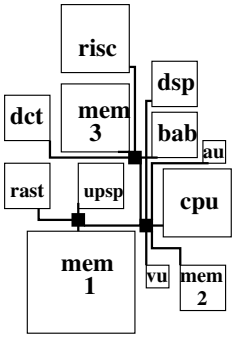

Figure 12: SS_m MPEG4 topology

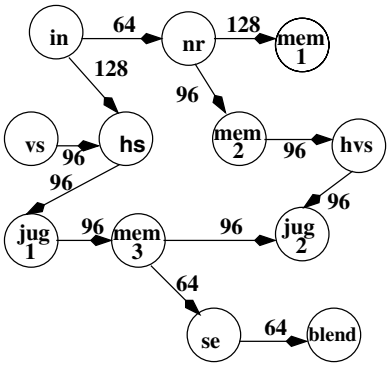

Figure 13: MWD CTG

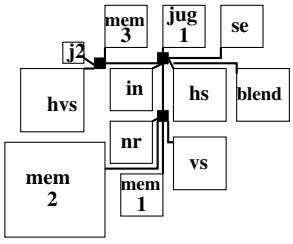

Figure 14: MS_m MWD topology

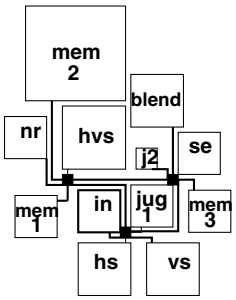

Figure 15: SS_m MWD topology

Benchmark	ID	Nodes	Edges
mp3 decoder	G1	5	3
263 encoder	G2	7	8
mp3 encoder	G3	8	8
263 decoder	G4	8	9
MPEG4	G5	12	13
MWD	G6	12	13
263 enc mp3 dec	G7	14	12
mp3 enc mp3 dec	G8	15	12
263 enc mp3 enc	G9	15	17
263 enc 263 dec	G10	16	17

Table 1: Benchmarks

Floorplanning	Interconnection generation algo	ID
MILP	MILP	MM
MILP	ANOC and merging	MS_m
MILP	ANOC no merging	$MSnm$
Slicing Tree	ANOC and merging	SS_m
Slicing Tree	ANOC no merging	SS_{nm}

Table 2: Techniques

The *generate_topology* function of the ANOC algorithm performs the topology generation, and merging operations. In the function, lines 1 through 3 generate traffic for each traffic trace. Lines 4 through 6 assign a router for each node, lines 7 through 11 places routers at nodes of the CIG that merging or diverging points of three or more edges, lines 9 through 11 generate routes for each trace by following the links on which trace is mapped, and finally, line 12 performs the merging of routers.

Possible deadlocks in the final topology can be removed by introducing virtual channels as a post processing step [21].

4. COMPLEXITY ANALYSIS

In this section, we present an analysis of the complexity of the ANOC algorithm. Since links are placed by recursively partitioning the CIG, there are $K = O(|V|)$ links. Since each trace is mapped at most $O(log(|V|))$ times, mapping of traces takes at most $O(K|E|log(V))$ operations. Hence, the complexity of the function is $O(|V||E|log(|V|))$. The complexity of the topology generation function is given by $O(|E||V| + |V|^2)$, where $|E||V|$ is for topology generation, and $|V||$ is for router merging. Hence, the overall complexity of the ANOC technique can be represented as $O(|V||E|log(|V|))$.

5. RESULTS

We first discuss the benchmark applications, the experimental setup, and finally, we present and discuss the results.

Benchmark	Power(μW)			Routers		
	MM	MS_m	Ratio	MM	MS_m	Ratio
G1	2.62	3.3952	1.29	1	1	1
G2	108.3	162.19	1.49	2	2	1
G3	5.7	6.8709	1.20	2	2	1
G4	5.7	8.2988	1.45	3	2	0.66
G5	3672	5929.2	1.61	4	4	1
G6	6056	11125	1.91	4	3	0.75
G7	179.5	144.05	0.80	5	4	0.8
G8	8.635	11.490	1.33	5	5	1
G9	170.7	138.11	0.80	5	4	0.8
G10	245.4	177.58	0.72	7	4	0.57

Table 3: Comparison of MM and MS_m

Benchmark	Power(μW)			Routers		
	MS_m	MS_{nm}	Ratio	MS_m	MS_{nm}	Ratio
G1	3.3952	3.7980	0.89	1	3	0.33
G2	162.19	228.21	0.71	2	5	0.4
G3	6.8709	7.9698	0.86	2	6	0.33
G4	8.2988	9.6370	0.86	2	4	0.5
G5	5929.2	6482.3	0.91	4	6	0.66
G6	11125	11183	0.99	3	8	0.37
G7	144.05	214.76	0.67	4	12	0.33
G8	11.490	12.454	0.92	5	12	0.41
G9	138.11	184.92	0.74	4	6	0.66
G10	177.58	180.96	0.98	4	10	0.4

Table 4: Comparison of MS_m and MS_{nm}

Benchmark applications: We generated custom NoC architectures for several combinations of multimedia benchmarks namely, i) MP3 audio encoder ii) MP3 audio decoder iii) H.263 video encoder, and iv) H.263 video decoder [9]. In addition, we obtained results for two other benchmarks namely, MPEG4 decoder, and multi-window display (MWD) applications [5]. The description of the benchmarks is shown in Table 1.

Experimental setup: We estimated the power consumption for the input and output traffic of a port in 100 nm technology to be $328\mu W/Mbps$ and $65.5\mu W/Mbps$, respectively. We estimated the physical link power consumption to be $79.6\mu W/Mbps/mm$ [16]. All results were obtained on a 950 MHz dual sparc processor .

Discussion: Table 2 summarizes the different techniques utilized to compare our results. In the table, the first column represents the floorplanning technique (Phase I), the second column represents the interconnection network generation technique (Phase II), and

Benchmark	Power(μW)			Routers		
	SS_m	SS_{nm}	Ratio	SS_m	SS_{nm}	Ratio
G1	3.7487	3.7487	1	2	2	1
G2	128.97	239.61	0.53	1	6	0.16
G3	6.8709	7.9698	0.86	2	6	0.33
G4	8.2988	9.6370	0.86	2	4	0.5
G5	5857.7	8013.9	0.73	3	11	0.27
G6	10381	10756	0.96	3	8	0.37
G7	187.78	190.81	0.98	3	7	0.42
G8	12.190	16.119	0.75	4	11	0.36
G9	200.14	223.44	0.89	5	7	0.71
G10	202.08	242.47	0.83	5	14	0.35

Table 5: Comparison of SS_m and SS_{nm}

Benchmark	Power(μW)			Routers		
	MS_m	SS_m	Ratio	MS_m	SS_m	Ratio
G1	3.3952	3.7487	0.92	1	2	0.5
G2	162.19	128.97	1.25	2	1	2
G3	6.8709	6.8709	1	2	2	1
G4	8.2988	8.2988	1	2	2	1
G5	5929.2	5857.7	1.01	4	3	1.3
G6	11125	10381	1.11	3	3	1
G7	144.05	187.78	0.76	4	3	1.3
G8	11.490	12.190	0.94	5	4	1.2
G9	138.11	200.14	0.69	4	5	0.8
G10	177.58	202.08	0.87	4	5	0.8

Table 6: Comparison of MS_m and SS_m

the third column gives a unique name for each combination.

Comparison with MILP formulation : We compared our technique with an optimal ILP formulation proposed in [1]. The computational complexity of the ILP formulation is exponential in the number of inputs. On the other hand, the technique presented in this paper is based on deterministic algorithms, and we show in Section 4 that the computational complexity of the technique is low. The results are presented in Table 3. In the table, the first column describes the benchmark application, the second column presents the power consumption of ILP based technique, the third column presents the power consumption of our technique, the fourth column presents that ratio of our technique to that of the ILP based technique in terms of power, and the fifth, sixth, and seventh columns present the corresponding values for router resources. Our technique was able to generate topologies that on average consumed only 1.25 times the power, and 0.85 times the router resources, compared to the corresponding topologies generated by the MILP formulation. We set the timeout period of the MILP formulation at 12 hours. The MILP formulation took several hours to generate results. In many cases, the formulation timed out before generating optimal results. Since our design technique has a low complexity, our technique was able to generate results for all benchmarks in negligible time ($< 1sec$).

Comparison of the techniques with and without merging : Table 4 compares results for the ANOC algorithm with and without merging respectively, on the floorplan generated by the MILP formulation. The organization of the table is similar to Table 3. Table 5 presents a similar comparison for slicing tree based floorplan. On average, the merging of routers resulted in 15% reduction in power consumption and 45% reduction in consumption of router resources for the MILP based floorplan. The corresponding values for slicing tree based floorplan were 17% and 46%, respectively. The reduction in router resources due to merging was much more than the reduction in power. The technique merges redundant routers, thus generating topologies that have higher router utilization. Although this paper does not address leakage power reduction explicitly, higher router utilization results to reduced leakage power.

Comparison between MILP and slicing tree floorplan : Table 6 compares the power and router resource consumption of the topologies generated by our technique with merging for MILP and slicing tree floorplans. On average, the topologies for MILP floorplan consumed 0.95 times the power and 1.05 times router resources, compared to the slicing tree floorplan when merging was applied. The corresponding values when merging was not applied were 0.92, and 1.02, respectively. Since our technique minimizes power as a primary goal, and the floorplans generated by the MILP floorplanner are better than the slicing tree based floorplanner, the topologies for MILP floorplans consumed lesser power than the corresponding slicing tree based floorplans

Topologies : The communication trace graph for MPEG4 decoder, and MWD applications are shown in shown in Figures 10, and 13, respectively. In the graph, the circles denote the processing or memory cores, and are annotated by their respective types. The edges denote the communication between cores, and are annotated by bandwidth requirement in Mbps. Figures 11, and 14 present the corresponding physical layout, and interconnection network topology of the applications for the MILP floorplan. Figures 12, and 15 present the corresponding physical layout, and interconnection network topology of the applications for the slicing tree floorplan. In the figure, the black squares denote the router elements. Our floorplanning technique places highly communicating nodes close to each other. For example, in the MPEG4 decoder benchmark, the nodes labeled *mem1* and *upsp* are placed close to each other as the bandwidth of the trace between *mem1* and *upsp* is high. The ANOC algorithm then routes the trace in minimum number of hops.

6. CONCLUSION

In this paper, we proposed a novel low complexity technique for design of application specific custom on-chip interconnection architectures. We experimented with many representative multimedia benchmarks to demonstrate the superior quality of our design. Due to their low complexity, our technique was able to generate results for all benchmarks in less than one sec. We compared our technique with an optimal MILP based technique whose computational complexity is not bounded. Overall, our technique consumes less than 1.25 times the power consumption, 0.85 times the router resources, and generates results in less than one second compared to the MILP technique that took hours to generate results.

7. REFERENCES

[1] K. Srinivasan, K.S. Chatha, and G. Konjevod. "Linear Programming based Techniques for Synthesis of Network-on-Chip Architectures". *Accepted at IEEE Transactions on VLSI*, In Press.

[2] W. J. Dally and B. Towles. "Route Packet, Not Wires: On-Chip Interconnection Networks". In *Proceedings of DAC*, June 2002.

[3] L. Benini and G. De Micheli. "Networks on Chips: A New SoC Paradigm". *IEEE Computer*, pages 70–78, January 2002.

[4] M.B. Taylor, J. Kim, J. Miller, D. Wentzlaff, F. Ghodrat, B. Greenwald, H. Hoffman, P. Johnson, J-W Lee, W. Lee, A. Ma, A. Saraf, M. Seneski, N. Shnidman, V. Strumpen M. Frank, S. Amarasinghe, and A. Agrawal. "The RAW Microprocessor: A Computational Fabric for Software Circuits and General-Purpose Programs". *IEEE Micro*, pages 25–35, March-April 2002.

[5] A. Jalabert, S. Murali, L. Benini, and G. De Micheli. "xpipesCompiler: A tool for instantiating application specific Networks on Chip". In *DATE*, 2004.

[6] Nilanjan Banerjee, Praveen Vellanki, and Karam S. Chatha . "A Power and Performance Model for Network-on-Chip Architectures " . *DATE*, pages 68–75, 2004.

[7] S. Murali, L. Benini, and G. De Micheli. "Mapping and Physical Planning of Networks-on-Chip Architectures with Quality-of-Service Guarantees ". In *Proceedings of ASPDAC*, 2005.

[8] S. Murali, and G. De Micheli. "Bandwidth-Constrained Mapping of Cores onto NoC Architectures". In *DATE*, 2004.

[9] J. Hu and R. Marculescu. "Energy-Aware Mapping for Tile-based NoC Architectures Under Performance Constraints". In *ASP-DAC*, 2003.

[10] K. Srinivasan, and K. S. Chatha. "A Technique for Low Energy Mapping and Routing in Network-on-Chip Architectures ". In *Proceedings of ISLPED*, San Diego, CA, USA, August 2005.

[11] A. Pinto, L. P. Carloni, and A. L. Sangiovanni-Vincentelli. "Efficient Synthesis of Networks On Chip". In *ICCD*, 2003.

[12] K. Srinivasan, K. S. Chatha, and G. Konjevod . "Linear Programming based Techniques for Synthesis of Network-on-Chip Architectures ". In *Proceedings of ICCD*, San Jose, USA, October 2004.

[13] K. Srinivasan, and K. S. Chatha. "ISIS: A Genetic Algorithm based Technique for Synthesis of On-Chip Interconnection Networks ". In *Proceedings of VLSI Design*, Calcutta, India, January 2005.

[14] K. Srinivasan, and K. S. Chatha. "SAGA: Synthesis Technique for Guaranteed Throughput NoC Architectures". In *Proceedings of ASPDAC*, Shanghai, China, January 2005.

[15] Umit Ogras and Radu Marculescu. "Energy and Performance Driven NoC Communication Architecture Synthesis Using A Decomposition Approach". In *DATE*, 2005.

[16] K. Srinivasan, K. S. Chatha, and G. Konjevod . "An Automated Technique for Topology and Route Generation for Application Specific On-Chip Interconnection Networks". In *Proceedings of ICCAD*, San Jose, USA, October 2005.

[17] S.M Sait and H. Youssef. *VLSI Physical Design Automation: Theory and Practice*. McGraw-Hill Inc, 1994.

[18] P. Chen, and E.S Kuh. "Floorplan Sizing by Linear Programming Approximation". In *Proceeding of DAC*, Los Angeles, California, June 2000.

[19] J.G Kim, and Y.D Kim. "A Linear Programming Based Algorithm for Floorplanning in VLSI Design". *IEEE Transactions on CAD*, 22(5), 2003.

[20] C.M Fiduccia and R.M Mattheyses. "A Linear-Time Heuristic for Improving Network Partitions ". In *Proceedings of DAC*, 1982.

[21] W.J. Dally and C.L. Seitz. "Deadlock-free message routing in multiprocessor interconnection networks". *"IEEE Transactions on Computers"*, C-36(5):547–553, 1987.