SUNMAP: A Tool for Automatic Topology Selection and Generation for NoCs

Srinivasan Murali
Computer Systems Lab
Stanford University
Stanford, CA-94305, USA
smurali@stanford.edu

Giovanni De Micheli
Computer Systems Lab
Stanford University
Stanford, CA-94305, USA
nanni@stanford.edu

ABSTRACT

Increasing communication demands of processor and memory cores in *Systems on Chips* (SoCs) necessitate the use of *Networks on Chip* (NoC) to interconnect the cores. An important phase in the design of NoCs is the mapping of cores onto the most suitable topology for a given application. In this paper, we present SUNMAP a tool for automatically selecting the best topology for a given application and producing a mapping of cores onto that topology. SUNMAP explores various design objectives such as minimizing average communication delay, area, power dissipation subject to bandwidth and area constraints. The tool supports different routing functions (dimension ordered, minimum-path, traffic splitting) and uses floorplanning information early in the topology selection process to provide feasible mappings. The network components of the chosen NoC are automatically generated using cycle-accurate SystemC soft macros from ×pipes architecture. SUNMAP automates NoC selection and generation, bridging an important design gap in building NoCs. Several experimental case studies are presented in the paper, which show the rich design space exploration capabilities of SUNMAP.

Categories and Subject Descriptors

B.8.2 [**Performance and Reliability**]: Performance Analysis and Design Aids; B.4.3 [**Input/Output and Data Communications**]: Interconnections—*Topology*

General Terms

Design, Algorithms, Performance

Keywords

Systems On Chip, Networks on Chip, Topology, Mapping, SystemC.

1. INTRODUCTION

The heavy communication demands of future *Systems on Chips* (SoCs) require scalable communication architectures to interconnect the cores. The interconnect scalability for bus based systems that are widely used in current SoCs is limited. This motivates a shift in the communication design paradigm from non-scalable bus based architectures to scalable *Networks on Chip* (NoC) based architectures [1, 2, 3, 4]. NoCs are compatible with core design and re-use and have several advantages including better structure, performance and modularity.

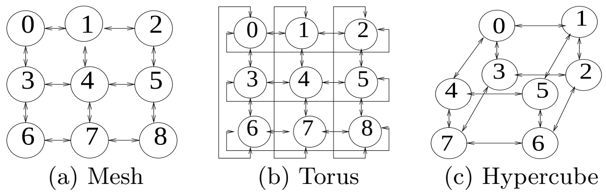


Figure 1: Direct NoC Topologies

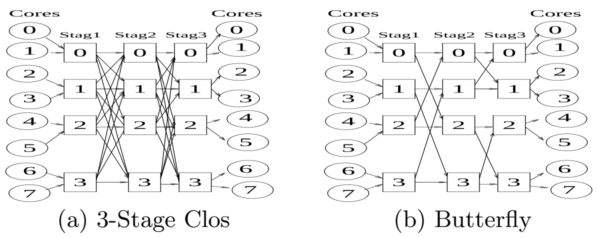
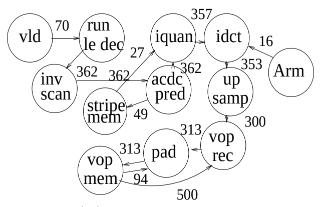


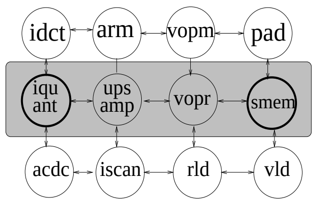
Figure 2: Indirect NoC Topologies

An important phase in the design of NoCs is choosing the most suitable NoC topology for a particular application and mapping of the application on to that topology. There are several standard topologies onto which an application can be mapped. They are broadly classified as direct topologies where each switch is connected to a single core (Figure 1) and indirect topologies where a set of cores are connected to a switch (Figure 2). Choosing the right topology involves exploring various design objectives such as minimizing average communication delay (i.e. latency), power consumption, area, etc. Moreover, in the resulting NoC the links should support the desired traffic between various cores. Thus the mapping of cores onto the NoCs must satisfy the bandwidth constraints of the application.

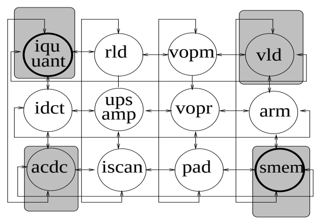
As a motivating example, a *Video Object Plane Decoder* (*VOPD*) [13] mapped onto two different topologies (mesh and torus) is considered (Figure 3). The cost of the mappings in terms of area, power and communication delay are evaluated (as explained in Section 5) and are presented in Figure 3(d). The average communication delay for the torus network is slightly lower (10%) as more network resources are utilized (more links and larger switches). The area, power analysis show large savings for the mesh network (20% power savings and 5% area savings) and it is more suitable



(a) *VOPD* graph



(b) Mesh



(c) Torus

Perf Met	Mesh	Torus	Ratio tor/mesh
avg hops	2.25	2.03	0.9
des area (mm ^{2})	54.59	57.91	1.06
des pow (mw)	372.1	454.9	1.22

(d) Design Parameters

Figure 3: Example Mappings of *VOPD*

than a torus for this application. This elucidates the need for exploring various design objectives for choosing the most suitable network topology for a particular application.

The purpose of this research is to describe a method and a tool, `SUNMAP`, for high-level mapping of cores onto various network architectures and to choose the architecture that is most efficient for the application. `SUNMAP` maps cores onto several standard NoC topologies (mesh, torus, hypercube, butterfly, clos) with various objective functions such as minimizing average communication delay, design area and power dissipation, satisfying the bandwidth and area constraints. The tool also supports various routing functions (dimension ordered, minimum-path, traffic splitting across minimum-paths, traffic splitting across all paths) and chooses the best topology from the library of available topologies for the given application. Note that the approach presented here is general and other topologies (such as octagon network or star network [6, 10]) can be easily added to the topology library. `SUNMAP` has an interface to the $\times$`pipesCompiler` [18], which is a tool for instantiating network components (switches, links, network interfaces) using a library of composable SystemC soft macros. After topology selection and mapping, the SystemC description of network components and their interconnection with the cores are automatically generated. The resulting SystemC code for the whole design can be simulated at the cycle-accurate and signal accurate level.

For the mapping process, we use traffic models as the abstraction of communication between cores. Obviously, the use of high-level models hides a lot of complex technological aspects, but facilitates fast exploration of the design space. Moreover as `SUNMAP` generates SystemC models of the mapped NoC, the design can be accurately simulated, providing a reliable path for system design and verification.

2. PREVIOUS WORK

The need to replace busses with on-chip networks has been presented in [1, 2]. Several NoC architectures and design methodologies have been presented recently, such as mesh based Nostrum [4] and aSoC [8], fat-tree based SPIN [3], ring based Proteo [9], etc. In [6], the use of octagon communication topology for network processors is presented. In [5], designing NoC with Quality-of-Service (QoS) guarantees has been explored. Hierarchical approach for designing on-chip networks was presented in [7]. Design methodologies for building irregular networks has been presented in [14], [15]. We refer the reader to [11] for surveys on several aspects of NoC design.

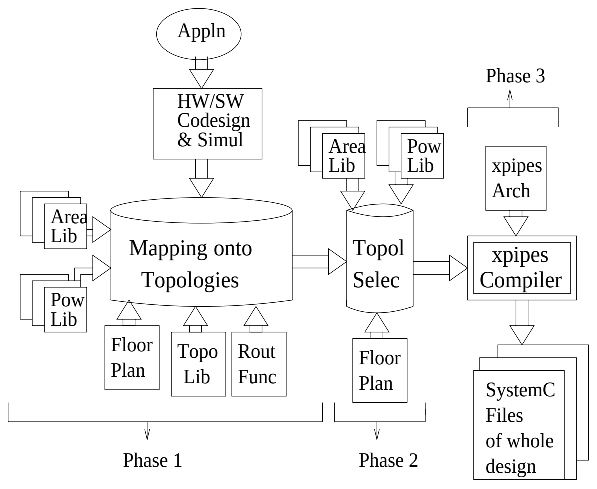


Figure 4: Design Flow of `SUNMAP`

The problem of mapping cores onto NoC architectures is presented in [16],[19]. In [16], a branch-and-bound algorithm is used to map cores onto a mesh-based architecture with the objective of minimizing energy, satisfying the bandwidth constraints of the NoC. A simple dimension-ordered routing is assumed in the work. In [19], fast algorithms for mesh NoC architectures under different routing functions, minimizing the average communication delay and satisfying bandwidth constraints is presented.

To the best of our knowledge, this is the first work targeting the problem of application-specific NoC topology selection and generation. We consider several different topologies, routing functions and design objectives and automate the mapping and topology selection processes. We use floorplan information for area-power estimates early in the mapping process and consider area and bandwidth constraints of the NoC to evaluate the feasibility of mappings. The network components of the chosen topology are automatically built using SystemC soft macros from $\times$`pipes` library. Thus, `SUNMAP` bridges an important design gap in building NoCs.

3. DESIGN METHODOLOGY

The design flow of the `SUNMAP` tool is presented in Figure 4. We assume that the application is mapped onto cores using existing tools such as [12]. By static analysis or simulation, the amount of data transfer between the cores is obtained. The resulting cores and communication demands between them is the input to our tool. `SUNMAP` has three phases of operation.

In the first phase, for a chosen routing function and design objective, mappings onto various network topologies in the topology library are obtained. For each mapping, the area and bandwidth constraints are evaluated to produce a feasible mapping. The area-power libraries and floorplanner are built into `SUNMAP`, so that area-power estimates can be incorporated early in the mapping process. The details of the area-power models and floorplanner are presented in Section 5.

In the second phase, the various topologies (with mappings produced from the first phase) are evaluated for several design objectives and the best topology is chosen.

In the third phase, the tool generates SystemC description of the network components using the $\times$`pipesCompiler` and $\times$`pipes` architecture. The three design phases are delineated in Figure 4. In the following sections we elaborate on the first two phases of the design methodology and refer the reader to [17], [18] for a detailed description of the architecture of the network elements and their automatic generation for a chosen topology.

4. MAPPING OF CORES ONTO TOPOLOGIES

In [19], mapping of cores onto a mesh topology is presented along with the performance gains of the algorithm compared to previous mapping approaches. In this section we extend the mapping algorithms in [19], for several standard topologies (torus, hypercube, butterfly and clos) for minimum-path routing. Mapping algorithms for other routing functions are similarly extended for various topologies and are incorporated in the tool, but due to lack of space, here we present only the minimum-path mapping algorithm.

Before presenting the algorithm, we formulate the mapping problem mathematically. The communication between the cores of the SoC is represented by the *core graph*:

Definition 1. *The core graph is a directed graph, $G(V,E)$ with each vertex $v_i \in V$ representing a core and the directed edge (v_i, v_j), denoted as $e_{i,j} \in E$, representing the communication between the cores v_i and v_j. The weight of the edge $e_{i,j}$, denoted by $comm_{i,j}$, represents the bandwidth of the communication from v_i to v_j.*

The connectivity and link bandwidth of the NoC is represented by the *NoC topology graph*:

Definition 2. *The NoC topology graph is a directed graph $P(U,F)$ with each vertex $u_i \in U$ representing a node in the topology and the directed edge (u_i, u_j), denoted as $f_{i,j} \in F$ representing a direct communication between the vertices u_i and u_j. The weight of the edge $f_{i,j}$, denoted by $bw_{i,j}$, represents the bandwidth available across the edge $f_{i,j}$.*

The mapping of the core graph $G(V,E)$ onto the topology graph $P(U,F)$ is defined by the one-to-one mapping function *map*:

$$map : V \rightarrow U \text{ , s.t. } map(v_i) = u_j, \forall v_i \in V, \exists u_j \in U \quad (1)$$

The mapping is defined when $|V| \leq |U|$. An example mapping of the *VOPD* core graph onto mesh and torus topology graphs is shown in Figure 3. The communication between each pair of cores (i.e. each edge $e_{i,j} \in E$) is treated as a flow of single commodity, represented as d_k, $k = 1, 2, \cdots, |E|$. The value of d_k represents the bandwidth of communication across the edge and is denoted by $vl(d_k)$. The set of all commodities is represented by D and is defined as:

$$D = \left\{ \begin{array}{l} d_k : vl(d_k) = comm_{i,j}, k = 1, 2, \cdots, |E|, \forall e_{i,j} \in E, \\ \text{with } source(d_k) = map(v_i), dest(d_k) = map(v_j) \end{array} \right\} \quad (2)$$

4.1 General Mapping Algorithm

In this sub-section, we present the generalized minimum-path mapping algorithm and in the next sub-sections we show how the algorithm is modified for each topology.

The mapping algorithm is presented in Figure 5. As the mapping problem is intractable [19], we use a heuristic approach with three phases: in the first phase an initial mapping is obtained using a greedy algorithm. In the initial mapping procedure (step 1 in Figure 5), first the core that has maximum communication is placed on to the NoC node with maximum neighbors. Then the core that communicates the most with placed cores is chosen. This core is placed onto the NoC node that minimizes the cost function and this procedure is repeated until all the cores are placed.

Once an initial mapping is obtained, in the second phase (steps 2 to 8), the commodities are sorted in decreasing order of their values. Then, for each commodity in order, a quadrant graph between the source and destination of the commodity is formed, as the shortest path between the source and destination lies within the quadrant between them. The shaded regions in Figure 3 are examples of quadrant graphs for the communication between the cores `smem` and `iquant`.

```
Mapping (G(V,E), P(U,F))
{
 1 obtain an initial greedy mapping of G onto P;
 2 sort commodities in D with decrasing comm
   costs;
 3 for each $d_k$ in D in order of decreasing cost
   {
 4      make quadrant graph Q($d_k$) with source($d_k$)
        and dest($d_k$) as end vertices;
 5      Path(source($d_k$), dest($d_k$)) = minpath(Q($d_k$));
 6      Increase edge weigths in Path by vl($d_k$);
   }
 7 floorplan_area_power_estimates(P);
 8 if bandwidth and area constraints are satisfied,
   find the mapping cost;
 9 repeat steps 2 to 8 for each pair−wise swap of
   vertices in P;
10 return the mapping with lowest cost of all evaluated
   mappings;
}
```

Figure 5: Minimum-path Mapping Algorithm

Then, Dijkstra's shortest path algorithm is applied (step 5) to the quadrant graph and the minimum path is obtained. The edge weights are incremented suitably and the procedure is repeated for each commodity in order. After routing all commodities, if the bandwidth and area constraints are satisfied, the cost of communication is calculated. Bandwidth constraints are satisfied, if in the resulting mapping, the traffic across any link is smaller than or equal to the capacity of the link.[1] The area constraints are satisfied when the mapped design area is lower than the maximum allowed area and aspect ratios of the design and soft core blocks (blocks that have flexible sizes) are within permissible ranges. For these area estimates (and power calculations that are needed when the mapping objective is power minimization), floorplanner and area-power libraries are incorporated into `SUNMAP` as explained in Section 5.

The mapping algorithms can have many different objectives such as minimizing communication delay, area or power dissipation and is an input parameter to `SUNMAP`. Depending on the objective function, the cost function calculation (done as part of step 8) varies.

In the last phase of the algorithm (steps 9 - 10), for each pair-wise swapping of vertices, phase-2 is repeated. Finally, the best mapping from all evaluated mappings is returned by the procedure.

As the minimum-path computations are performed on the quadrant graph instead of the entire NoC graph, large computational time savings is achieved, as the number of nodes in a quadrant graph is much smaller than the total NoC nodes. The above algorithm was validated for a mesh topology in [19], where the mappings produced by this algorithm was shown to be superior when compared to previous approaches.

In the mapping algorithm presented above, all but two of the steps are common to all topologies: the first step is the formation of NoC topology graph, which is obviously specific to a particular topology and the second step is the procedure used to form quadrant graphs, which varies with the topology used. These two steps are explained in detail in the following sub-sections.

[1]Capacity of a link in a NoC is technology and implementation dependent and is assumed as an input to `SUNMAP`.

4.2 NoC Topology Graph Definition

The formal definition of the NoC topology graph was presented in the beginning of this section. The edges in the NoC graph represents connection between adjacent NoC nodes. Thus for defining a topology graph, we simply need to define the nodes that are adjacent to a particular node in that topology.

For a mesh, each node, except the nodes on the edges have four neighbors (such as node 4 in Figure 1(a)), nodes in the four corners (e.g. node 0) have two neighbors, and other nodes in the edges (e.g. node 1) have three neighbors. A torus has similar structure as the mesh, but has additional wrap-around channels between the edge nodes (e.g. node 0 in Figure 1(b) is connected to nodes 2 and 6 on the opposite edges).

For a hypercube with N nodes (also called as 2-ary n-cube), each node has n (which is equal to $\log_2 N$) neighbors. A node u_i in such a network is represented by the n-tuple: (h_1,h_2, ..., h_n), which is the binary representation of the decimal i. Intuitively, each h_j represents a single dimension and thus an n-tuple uniquely identifies every node of a 2-ary n-cube. As an example, node 2 in Figure 1(c) is represented by $(0, 1, 0)$. All nodes with n-tuples distance 1 apart from the n-tuple for u_i are neighbors of u_i (node 6 whose 3-tuple is $(1, 1, 0)$ is adjacent to 2).

In this work, we consider 3-stage clos networks, where each switch in a stage is connected to every switch in the next stage (e.g. switch 0 of stage 1 in Figure 2(a) is connected to switches $0, 1, 2, 3$ of stage 2). Thus adjacency calculations are trivial for this topology. Butterfly networks (with N core nodes) are also known as k-ary n-fly networks, where k is the radix of switches in the network, and n is the number of stages in the network ($n = \log_k N$). A 2-ary 3-fly network is presented in Figure 2(b). As seen, the switches in each stage are connected to 2 switches in the next stage. The maximum distance between the adjacent switches halves with each stage (e.g. switch 0 of stage 1 is connected to switches 0 and 2 of stage 2, resulting in a maximum distance of 2. Switch 0 of second stage is connected to switches 0 and 1 of third stage, thus resulting in a maximum distance of 1).

4.3 Quadrant Graph Formation

The procedure for forming quadrant graphs is specific to a topology as the nodes that lie in the shortest path of a commodity is topology specific. Example quadrant graphs for mesh and torus networks were presented in Figure 3 (shaded areas in the Figure). For a mesh network, the nodes that are within the bounding box formed by the row and column boundaries of the source and destination nodes of a commodity form the elements of the quadrant graph of that commodity (Figure 3(b)). For a torus network, the wrap-around channels need to be considered for computing the smallest bounding box between the source and destination nodes (Figure 3(c)).

For hypercubes, all nodes that have matching h_j values (of the n-tuple) as that of the source and destination nodes of a commodity are included in the quadrant graph. As an example, for source node 0 (represented by (0,0,0)) and destination 3 (represented by (0,1,1)), all nodes with n-tuples of the form (0,*,*) (* represents don't care values), form the quadrant graph (i.e. nodes 0,1,2,3 are the elements of the quadrant graph). Intuitively, all nodes that have the same dimensions as the source and destination nodes are included in the quadrant graph.

As clos networks have full interconnection pattern between switches of adjacent stages and butterfly networks have no path diversity (a single path from any source to any destination), the quadrant graph formation for these networks is trivial.

5. AREA-POWER MODELS AND FLOORPLANNING

We developed analytical models for estimating the area of switches. The architecture of the switches is assumed to be based on the ×pipes architecture presented in [17]. The area calculations include the crossbar area, buffer area, logic (including control) area. The models take into account the nuances of individual switch configurations and include fine granularity of details (like accounting for pipeline registers, cross points, etc). We used ORION [22], a power modeling tool, for developing bit energy models for the switches. The area-power models are used to generate area-power libraries for various switch configurations for different technology parameters. In the rest of this paper, we assume 0.1μ technology and use the area-power libraries generated by the models for this technology. We use wiring parameters from [23] to estimate link power dissipation. We assume that the area-power values of the cores are an input to our tool.

The general solution to the floorplanning problem has two basic steps: first is finding the relative positions of modules and the second is finding the exact positions, area and size of the modules [20]. For a particular mapping that needs to be evaluated for area-power-latency, the relative positions of the cores and switches are known. Thus the floorplanning problem is reduced to the one of finding the exact positions and sizes (for soft blocks) of the cores and switches. We use a simple *Linear Program* (LP) based floorplanner existing in literature [21] for this purpose. Note that a more sophisticated floorplanner such as the one presented in [20] can be used in place of the simple floorplanner. As the major focus of this work is not floorplanning, we use a simple floorplanner in the tool. In our subsequent work, we plan to enhance the floorplanner to take specific features of NoCs into account.

The area and aspect ratio constraints (for feasibility of mapping) are evaluated and link lengths in the NoC are obtained from the floorplanner. Using the built-in power libraries, power dissipation for the switches and links are calculated based on the average traffic (shown as edge annotations in Figure 3(a)) through them. The computed area, power values are returned (step 7 in Figure 5) to the mapping algorithm.

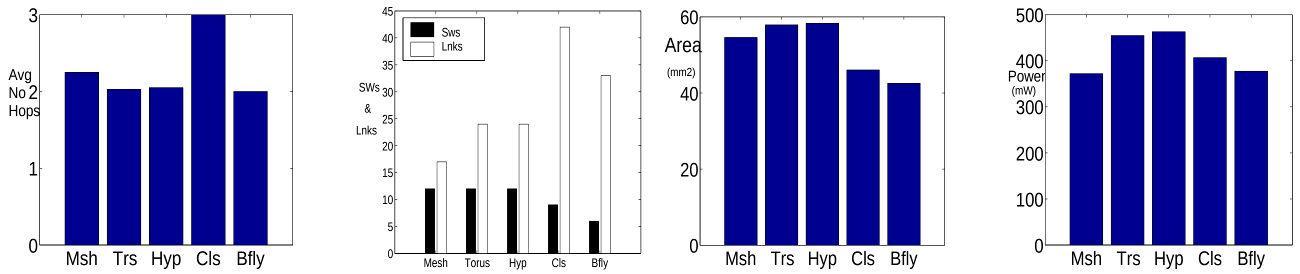
6. EXPERIMENTS AND CASE STUDIES

6.1 Experiments on Video Applications

We applied SUNMAP to two different video processing applications: the *Video Object Plane Decoder* (*VOPD*-mapped onto 12 cores) and the *MPEG4 decoder* (14 cores) [13]. The core graphs of the *MPEG4* and *VOPD* are presented in Figures 3(a), 7(a). The maximum link bandwidth for the NoCs is conservatively assumed to be 500 MB/s.

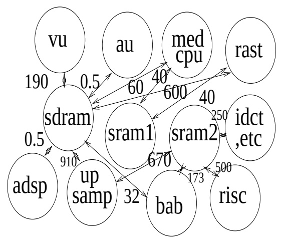
The results of mapping *VOPD* onto various topologies are presented in Figure 6. As seen from Figure 6(a), the butterfly topology (4-ary 2-fly) has the least communication delay out of all topologies. The lower communication delay is due to the fact that a 4-ary 2-fly has 2 stages of switches, which means an average delay of 2 hops for all communication. Mesh, torus and hypercube networks have a higher average hop delay as the least possible hop delay (that of adjacent nodes) itself is two and it was not possible to place all communicating nodes adjacent to each other. As the clos network has three stages, the average hop delay is three. The area, power estimates for the topologies are presented in Figures 6(c), 6(d). As seen from Figure 6(b), the butterfly topology has the least number of switches, but has more links when compared to mesh, torus or hypercube.

The large power savings achieved by the butterfly network (Figure 6(d)) is attributed to the fact that there are fewer switches and smaller number of hops for communication. Moreover, all the switches are 4x4, while the di-



(a) Avg hop delay (b) Resource Util (c) Design Area (d) Design Power

Figure 6: Mapping Characteristics of *VOPD*



(a) *MPEG4* core graph

Topo	av hops	des area (mm ²)	des pow (mw)
Mesh	2.49	62.51	445.4
Torus	2.47	66.03	541.4
Hyper	2.48	67.05	546.7
Clos	3.0	64.38	504.1
Bfly	No Feasible Mapping		

(b) Design Param

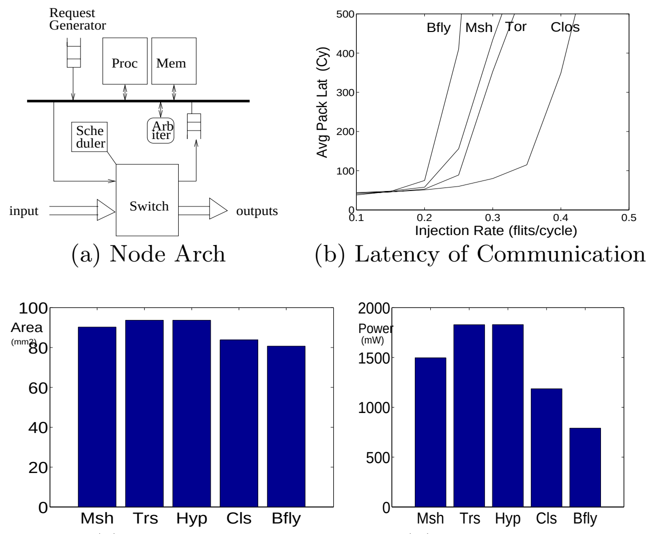
Figure 7: *MPEG4* Mappings

rect topologies have 5x5 switches (for communicating with four neighbors and to the core). The average link length in the butterfly network (obtained from floorplanner) was observed to be longer than the link lengths (around 1.5×) of direct networks. However, as the link power dissipation is much lower than the switch power dissipation, we get large power savings for the butterfly network. The smaller number of switches and smaller switch sizes also account for the large area savings achieved by the butterfly network. Thus, butterfly is the best topology for *VOPD*. The performance gains for the butterfly over other topologies may be surprising, but after careful inspection we see the reason. Butterfly network trades-off path diversity for network switches and average hop delay. As the *VOPD* example has lower bandwidth demands compared to *MPEG4*, we are able to satisfy the bandwidth demands of the application using a butterfly network.

The results of mapping *MPEG4* are presented in Figure 7(b). As seen from the core graph of *MPEG4* decoder (Figure 7(a)), the amount of communication between the cores (such as to/from the shared SDRAM) is much higher than that can be supported by minimum-path routing. All topologies violate the bandwidth constraints for minimum-path routing. So we apply multi-path routing, splitting the traffic across many paths. As the butterfly network has no path diversity, it is unable to support the communication between the cores, and thus doesn't produce any feasible mapping for *MPEG4*. All other topologies produce feasible mappings with split-traffic routing. As seen from the figure, the torus network has lower communication hop delay than the mesh as it utilizes more network resources. However, the mesh network has large savings in area and power which overshadow the slightly higher communication delay cost. Thus a mesh topology is more suitable for the *MPEG4* than other topologies.

6.2 Experiments on Network Applications

We consider a network processor with 16-nodes, each node having the architecture shown in Figure 8(a), obtained from [6]. The objective of the communication architecture is to



(a) Node Arch (b) Latency of Communication

(c) Design Area (d) Design Power

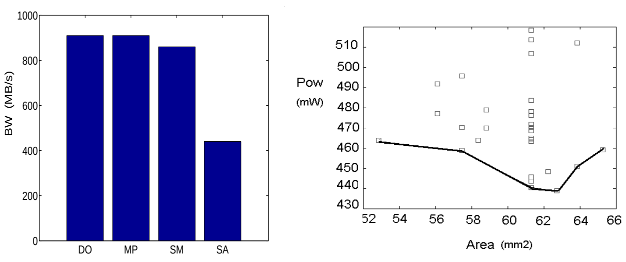
Figure 8: Network Processing Application

provide low contention for the data transfer between the nodes. As clos networks have maximum path diversity, they have the least congestion for large data flows and are more suitable for the network applications. We validated the need for clos networks by producing mappings onto various topologies by relaxing the bandwidth constraints and simulating the resulting SystemC design. We use traffic generators to generate adversarial traffic pattern for each topology. As seen from Figure 8(b), where the average packet latency is plotted with increasing traffic injection (which in turn mean that the network processor is processing larger amounts of data), the clos clearly outperforms other topologies. Moreover, the area and average power dissipation in a clos network (Figures 8(c), 8(d)) is only slightly higher than the butterfly topology, justifying its use for network processing applications.

6.3 Exploring Design Space of Topology

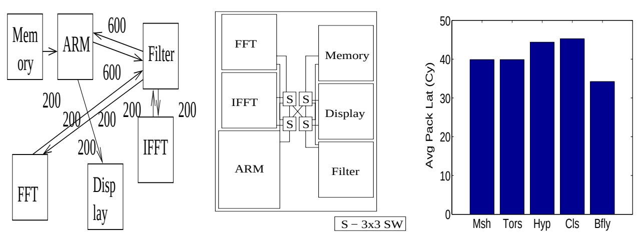
In this sub-section, we explore the *MPEG4* mappings onto a mesh topology. There are two ways in which a chosen topology can be explored: first is to evaluate the effects of various routing functions and second is to obtain a set of Pareto points for the mappings from which the optimum design point can be chosen, thereby performing area-power-performance tradeoffs.

The minimum bandwidth for different routing functions (DO - Dimension Ordered, MP - Minimum-path, SM - Split-traffic across Minimum-paths, SA - Split-traffic across All



(a) Effect of Routing (b) Area-Power Exploration

Figure 9: Design Space Exploration of a Topology



(a) Filter Appln (b) Bfly Floorplan (c) SystemC Plots

Figure 10: Mapping of DSP Application

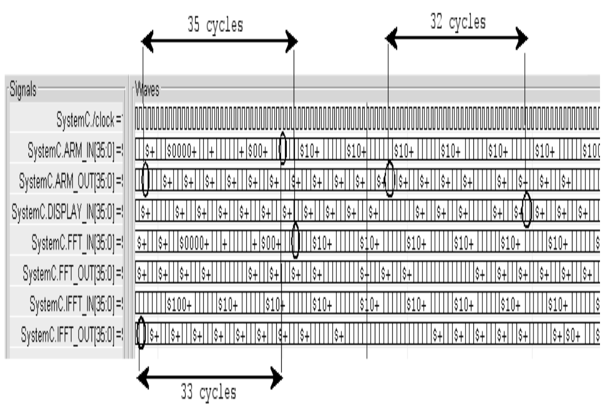


Figure 11: SystemC Snapshot

paths) is shown in Figure 9(a). When maximum available link bandwidth is 500 MB/s, only split-traffic routing can be used for mapping *MPEG4*. Figure 9(b) shows the Pareto points (for area-power trade-offs) in the design space of the mapping from which the optimum point can be chosen.

6.4 DSP Application and SystemC Simulations

We applied the `SUNMAP` algorithm to a DSP Filter design with six cores (refer Figure 10(a)). The cores are modeled in SystemC and the design is simulated at the transaction level. The resulting core graph is used by the `SUNMAP` which produces mappings onto the butterfly topology (Figure 10(b)). Then the network components for the butterfly topology are automatically generated and the resulting NoC design of the DSP is simulated at cycle accurate and signal accurate level in SystemC. A snap-shot of the SystemC simulation is shown in Figure 11. We also generated the best mappings of other topologies for comparison purposes. The SystemC simulation of all topologies is carried out, and the observed average packet latency for the topologies plotted (shown in Figure 10(c)). As seen from the Figure, the butterfly topology indeed has the minimum latency.

For all these applications NoC selection and generation was obtained in few minutes on a 1GHZ SUN workstation. The SystemC simulations were also checked for functional and timing correctness validating the output of `SUNMAP`.

7. CONCLUSIONS AND FUTURE WORK

Future Systems on Chips need scalable Networks on Chip architectures to interconnect the cores. Selecting the most suitable topology for an application, mapping of cores onto that topology and generating the resulting network are important phases in designing NoCs. In this paper we have presented `SUNMAP`, a tool that automates all these steps, bridging an important design gap in building NoCs. In future, we plan to enhance the tool with automatic heterogeneous topology modeling and guaranteeing Quality-of-Service for applications.

8. ACKNOWLEDGEMENTS

This research is supported by MARCO Gigascale Systems Research Center (GSRC) and NSF (under contract CCR-0305718).

9. REFERENCES

[1] M. Sgroi et al., " Addressing the System-on-a-Chip Interconnect Woes Through Communication-Based Design", in Proc. Design Automation Conference, 2001.
[2] L.Benini and G.De Micheli, "Networks on Chips: A New SoC Paradigm", IEEE Computers, pp. 70-78, Jan. 2002.
[3] P.Guerrier, A.Greiner,"A generic architecture for on-chip packet switched interconnections", DATE 2000, pp. 250-256, March 2000.
[4] S.Kumar et al., "A network on chip architecture and design methodology", ISVLSI 2002, pp.105–112, 2002.
[5] E.Rijpkema et al., "Trade-offs in the design of a router with both guaranteed and best-effort services for networks on chip",DATE 2003, pp. 350-355, Mar 2003.
[6] F.Karim et al., "On-chip communication architecture for OC-768 network processors", Design Automation Conference, June 2001.
[7] X.Zhu, S.Malik, "A Hierarchical Modeling Framework for On-Chip Communication Architectures", ICCD 2002, pp. 663-671, Nov 2002.
[8] L. Jian, et. al, "aSOC: A Scalable, Single-Chip communications Architecture", PACT 2000, Oct. 2000, pp. 37-46.
[9] D.Siguenza-Tortosa, J. Nurmi, " Proteo: A New Approach to Network-on-Chip", in CSN 02, Sep. 2002.
[10] S.J.Lee et al.," An 800MHz Star-Connected On-Chip Network for Application to Systems on a Chip", ISSCC 2003, Feb. 2003.
[11] A.Jantsch, H.Tenhunen, "Networks on Chip", Kluwer Academic Publishers, 2003.
[12] S.J.Krolikoski, et. al, "Methodology and Technology for Virtual Component Driven Hardware/Software Co-Design on the System-Level", ISCAS 99, pp. 456-459, June 1999.
[13] E.B.Van der Tol, E.G.T.Jaspers,"Mapping of MPEG-4 Decoding on a Flexible Architecture Platform", SPIE 2002, pp. 1-13, Jan, 2002.
[14] A.Pinto et. al, "Efficient Synthesis of Networks on Chip", ICCD 2003, pp. 146-150, Oct 2003.
[15] W.H.Ho, T.M.Pinkston, "A Methodology for Designing Efficient On-Chip Interconnects on Well-Behaved Communication Patterns", HPCA 2003, pp. 377-388, Feb 2003.
[16] J.Hu, R.Marculescu,"Energy-Aware Mapping for Tile-based NOC Architectures Under Performance Constraints", ASP-DAC 2003, Jan 2003.
[17] "×`pipes`: a Latency Insensitive Parameterized Network-on-chip Architecture For Multi-Processor SoCs", pp. 536-539, ICCD, 2003.
[18] "×`pipesCompiler`: A Tool For Instantiating Application Specific Networks on Chips", Vol. 2, pp. 20884, DATE 2004.
[19] "Bandwidth Constrained Mapping of Cores onto NoC Architectures", Vol. 2, pp. 20896, DATE 2004.
[20] J.G.Kim, Y.D.Kim, "A linear programming-based algorithm for floorplanning in VLSI design ", IEEE Transactions on CAD, pp. 584 -592, Vol. 22, Issue: 5 , May 2003,
[21] N. Sherwani, "Algorithms for VLSI Physical Design Automation". Kluwer Academic Publishers, 1995.
[22] H.S Wang et al., "Orion: A Power-Performance Simulator for Interconnection Networks", MICRO, Nov. 2002.
[23] R. Ho, K. Mai, and M. Horowitz, "The Future of Wires", Proceedings of the IEEE, pp. 490-504, April 2001.