Mapping and Physical Planning of Networks-on-Chip Architectures with Quality-of-Service Guarantees

Srinivasan Murali
CSL, Stanford University
Stanford, CA-94305
smurali@stanford.edu

Luca Benini
DEIS, Univ. of Bologna
Bologna, Italy
lbenini@deis.unibo.it

Giovanni De Micheli
CSL, Stanford University
Stanford, CA-94305, USA
nanni@stanford.edu

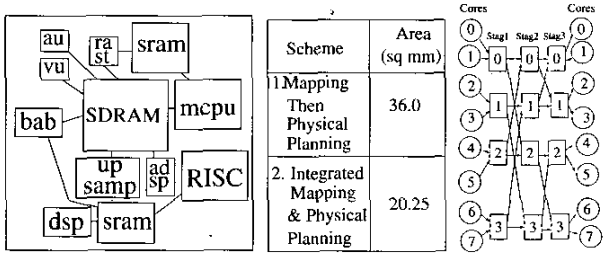
Abstract—*Networks on Chips (NoCs)* have evolved as the communication design paradigm of future *Systems on Chips (SoCs)*. In this work we target the NoC design of complex SoCs with heterogeneous processor/memory cores, providing Quality-of-Service (QoS) for the application. We present an integrated approach to mapping of cores onto NoC topologies and physical planning of NoCs, where the position and size of the cores and network components are computed. Our design methodology automates NoC mapping, physical planning, topology selection, topology optimization and instantiation, bridging an important design gap in building application specific NoCs. We also present a methodology to guarantee QoS for the application during the mapping-physical planning process by satisfying the delay/jitter constraints and real-time constraints of the traffic streams. Experimental studies show large area savings (up to 2×), bandwidth savings (up to 5×) and network component savings (up to 2.2× in buffer count, 3.8× in number of wires, 1.6× in switch ports) compared to traditional design approaches.

Keywords: Networks on Chips, Systems on Chips, Mapping, Physical Planning, QoS, Optimization.

I. Introduction

With scaling of transistor sizes, the number of processor and memory cores on *Systems on Chips (SoCs)* and their speed of operation is increasing. Future SoCs will have many different applications such as speech recognition, ambient intelligence, high-end video processing capabilities, 3D gaming, etc on the same device. In such complex systems, communication between the cores will become a major bottleneck as current bus-based communication architectures (either single bus or the state-of-the art multiple buses) will be inefficient in terms of throughput, latency and power consumption [2], [10]. Scalable interconnection networks are needed to interconnect the cores and the resulting communication centric design paradigm is called as *Networks on Chip (NoC)* [2].

SoCs are aggressively designed to meet the performance requirements of diverse applications that need to be supported. In most cases the cores in the SoC are heterogeneous in nature with each core performing a set of specialized functions in order to maximize performance and satisfy design constraints such as *Quality-of-Service (QoS)* for the applications. As an example, consider an efficient design of an *MPEG4 decoder* shown in Figure 1(a) [12]. In this design, there are several processors (for e.g. `RISC`), several hardware cores (e.g. `Upsampler`) and memory cores (e.g. `SDRAM`). Each core has different functionality, size and communication requirements. Some of the cores are *hard cores*, with size fixed during design (e.g. `RISC`) and some of the cores are *soft cores*, whose size can be varied with some restrictions on the aspect ratios (e.g. `Upsampler`). Figure 1(b) shows the design area for the best mappings of the *MPEG4* onto a mesh topology for two schemes: in the first scheme the mapping of the cores is done logically (without considering the physical planning of



(a) MPEG4

Scheme	Area (sq mm)
1 Mapping Then Physical Planning	36.0
2. Integrated Mapping & Physical Planning	20.25

(b) Mapping Schemes (c) Butterfly Network

Fig. 1. MPEG4 mapping schemes and example butterfly topology

the cores) followed by a separate physical planning phase and in the second scheme the mapping and physical planning are done together, so that the mapping process takes the physical planning information, i.e., the position of the cores and network components (e.g. switches, links) and the size of soft cores and switches in the 2-D plane. There is significant area improvement in the second scheme where mapping and physical planning are integrated together. This improvement will be even more pronounced for indirect topologies such as the *butterfly* network shown in Figure 1(c). In a butterfly topology, logically, the switches are arranged as stages with the switches in the first and last stages connected to the cores. Ideally we would like to distribute the switches around the cores so that performance of the NoC is maximized and mappings onto the butterfly should take this physical planning information into account.

Another important design consideration for SoCs is to guarantee *Quality-of-Service (QoS)* for the application. As an example, in many video applications, data should be communicated in such a way that the system supports a pre-determined frame rate (e.g. 30 frames/s in many video displays). The network should support the QoS requirements of the applications satisfying the delay constraints of the traffic streams. It should also provide support for real-time communication. These QoS guarantees need to be considered during the mapping process. Moreover the burstiness in the traffic streams (that makes providing QoS guarantees harder) needs to be considered.

In this work, we propose a design methodology for building complex application-specific NoCs. We provide an integrated approach to mapping and physical planning, where we determine the 2-D position of the cores and network components and the size of soft cores and switches during the mapping process. The physical planning phase also automatically computes the switch buffers needed to support the application traffic and integrates this in the switch size computation. We also present a method to provide QoS guarantees for the application during the mapping-physical planning phase. For QoS guarantees, we consider the burstiness in the application traf-

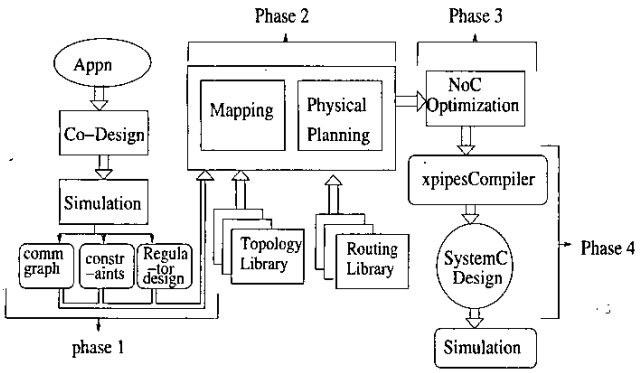


Fig. 2. Design Methology

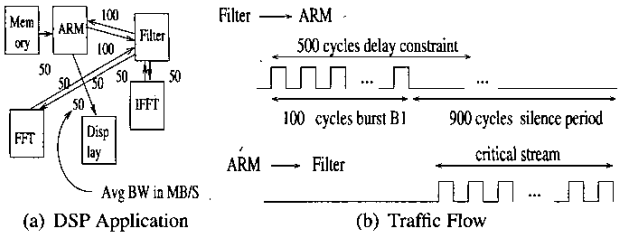


Fig. 3. *DSP Filter* application and traffic flow between *ARM* & *Filter* cores

fic, delay/jitter constraints of the individual traffic streams and provide support for real-time communication. The additional power-area overhead in obtaining the QoS guarantees is negligible. We integrate all these features into our tool presented in [21]. The mapping and physical planning of the cores is applied to several topologies defined in a topology library and the best topology for the application is automatically selected. In the resulting topology, the switches and links are optimized for the traffic characteristics, followed by automatic instantiation of the topology. Thus our integrated design methodology automates *mapping, physical planning, topology selection, optimization* and *instantiation* for an application providing *QoS guarantees*, thereby bridging an important design gap in building application-specific NoCs.

II. Previous Work

The use of NoCs to replace global wiring is presented in [1], [2]. The use of an *Octagon topology* for Network Processor design is presented in [3]. In recent years a large body of research focuses on the design methodologies for NoCs such as the *Nostrum* [4], *Scalable Programmable Interconnection Network (SPIN)* [5], *aSoC* [8], etc. We refer the reader to [10] for details of several NoC design methodologies. Several research works, such as [18], [19], [9], [7] have been presented for automatically instantiating software/RTL design of network components (such as network interfaces, switches and links) of an NoC. QoS guarantees to applications is provided by efficient router design in the *Aethereal NoC* [6]. In [17], performance analysis of communication architectures is presented. Stochastic analysis of on-chip MPEG traffic is presented in [22].

Topology design of NoCs for well-behaved traffic patterns is explored in [13], [14], [21], [20]. Mapping of cores onto NoC topologies is presented in [15], [16], [21],[20], [13]. In all these works, the mapping process is based on the average communication demands of the cores. A physical planner for homogeneous NoCs is presented in [23].

As the main contributions of this paper, we present an integrated approach to mapping and physical planning, guaranteeing QoS for applications by using a traffic regulator based design and integrate the mapping/physical planning phase with our existing tools to automate the complex design flow of NoCs. During the mapping-physical planning process we consider the burstiness, criticality and delay constraints of the traffic streams and ensure that QoS is guaranteed. We integrate these features to [21], so that our mapping and physical planning phases are followed by automatic topology selection and instantiation (of the SystemC design) of the NoC. The resulting NoC can be simulated at cycle-accurate level.

III. Design Methodology

The design methodology of the mapping and physical planning is illustrated in Figure 2. We assume that the parallel kernels in the application are mapped onto hardware/software cores using existing tools (such as [11]).

In the first phase, an initial simulation is carried out, from which the traffic characteristics (such as the burstiness, delay/jitter constraints and criticality of the traffic streams) are obtained. Based on the traffic characteristics of the application, graphs representing the traffic flow between cores and bandwidth constraints required for the various flows are generated. At this step, traffic regulators for ensuring QoS guarantees [25] are developed. These steps are explained in detail in sections IV and V.

In the next phase, the cores are mapped onto the topologies that are defined in the topology library. The mapping process is integrated with the physical planning process, where the positions and sizes of the various cores and switches are generated. The mapping/physical planning considers several design objectives such as minimizing design area, power or hop delay, satisfying the QoS constraints of the application (such as criticality constraints, delay/jitter constraints). As in [21], several routing functions defined in a library are also considered. This phase is explained in sections VI and VII.

In the third phase, the best network topology for the application (the topology that minimizes the design objective and satisfies the design constraints) is selected and a post-optimization of the network resources is carried out. In this step, the redundant switches, ports and links are removed and the bit-width (or frequency) of the links is adjusted to match the application needs.

In the last phase, the SystemC design of the resulting NoC is automatically generated using ×pipesCompiler [18]. The ×pipesCompiler instantiates SystemC network components (switches, links and network interfaces) for the NoC and automatically integrates them with the cores. The resulting design is then simulated at cycle-accurate level.

IV. On-Chip Traffic Modeling

In this section, we develop traffic models to characterize the application traffic, providing QoS guarantees for the application. As an example, consider the traffic flowing between the *Filter core* and the *ARM core* in a *DSP Filter* application (refer Figure 3). Without loss of generality assume that the packet size is such that a packet is sent in one cycle, although the following discussion also applies when a packet is sent over multiple cycles (i.e. when a packet has multiple flits). There are three important features to be noted from Figure 3(b).

- *Bursty Traffic Flows:* The application traffic from *Filter* to *ARM* core is bursty in nature, with a burst period of 100 cycles followed by 900-cycles of silence period. The peak-bandwidth of the traffic (100 packtes/100 cycles) is an order of magnitude higher than the average bandwidth (100 packets/1000 cycles).
- *Delay/Jitter Constraints:* Each burst from the *Filter* core has a delay constraint by which it should reach the *ARM* core. In this example, we assume that the burst B1 has to reach the *ARM* core by 500 cycles, which is obtained from the application characteristics.

TABLE I
LINK IMPLEMENTATION

Scheme	BW (pk/cy)	Delay (cyles)
1. Avg	100	1000
2. Peak	1000	100
3. Opt	200	500

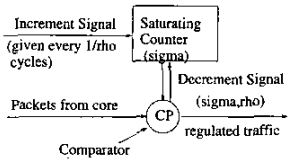


Fig. 4. A (σ,ρ) regulator

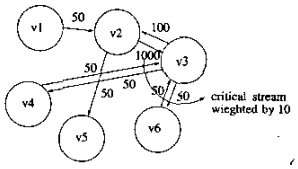


Fig. 5. Weighted core graph

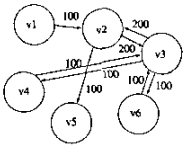


Fig. 6. Constraint Graph:BW in MB/S

- *Real Time Constraints:* The *ARM* core issues a control stream to the *Filter* which is assumed to be critical and needs to reach the *Filter* as quickly as possible. These real-time requirements need to be satisfied by the network.

Consider three implementations of the communication link (refer Table I) between the *Filter* core and *ARM* core (for illustrative purposes assume other cores don't send traffic on this link). In the first case, the link is designed to support the average bandwidth of traffic flowing between *Filter* and *ARM*. As seen from Table I, the delay incurred in this scheme for the burst B1 violates the delay constraint for the stream. In the second case, the link is designed for the peak bandwidth requirements and the delay constraints are met. However, the link is over-designed with 5× the capacity that is needed to support the delay constraints of the burst. In the third case, the link is optimally designed to support the burst without violating the delay constraints.

From this example, it is clear that the communication links should be designed optimally in a way such that they support the traffic flowing through them, satisfying the delay/jitter constraints of the traffic streams. Moreover, there should be a mechanism that ensures that each core sends traffic so that the links can support the traffic and the delay constraints are met. Clearly these two objectives complement each other and to ensure that the objectives are met we propose the use of traffic regulators for NoCs. Traffic regulators are widely used in *ATM* networks to guarantee QoS to applications [25]. A traffic regulator can be abstracted as a hardware block with two parameters: σ and ρ. The parameter ρ represents the bandwidth required to support the traffic streams so that the delay constraints are met and the parameter σ represents the variations permitted over the ρ value. Such a regulator is also called as a (σ,ρ) regulator [25]. The traffic flow between each source-destination is represented by a (σ, ρ) value. As an example, the *Filter* to *ARM* communication is represented by (0,0.2), which means that one packet can be be sent every 5 cycles (i.e. one packet can be sent every $1/\rho = 1/0.2 = 5$ cycles) and no variations over the required rate is permitted (as the σ value is 0). A (1,0.2) regulator would allow a burst of one packet over the required packet rate. In the rest of this paper, we assume that the σ value is chosen to be equal to 0, so that no variation is permitted over the required rate. To ensure that each core sends data according to the regulator values, we need to add small hardware to each core (or to the Network Interface connecting the core to the network), which is shown in Figure 4. The additional hardware consists of a saturating credit counter and a comparator. The saturating counter is incremented at rate ρ and saturates when it reaches a count of $(1 + \sigma)$. A packet is transmitted only if the credit counter is non-zero and when a packet is transmitted the counter is decremented by 1. This counter ensures that the amount of traffic transmitted by the source matches the rate for which the links are designed to handle. For traffic streams to different destinations, different sets of (σ, ρ) values are used in the regulator. Note that power-area overhead of such a regulator is negligible as it's just a counter and a comparator. For supporting real-time constraints, we assume tight latency bounds for the real-time stream and during the mapping process we consider the criticality of the stream by using a weighted communication graph (called as *weighted core graph*), where the weights are a function of the criticality. In the next section, we explain this in more detail, where we present mathematical models for modeling the ρ value for the regulators.

V. PROBLEM FORMULATION

The communication between the cores of the SoC is represented by the *weighted core graph*:

Definition 1 *The weighted 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 average bandwidth of the communication from v_i to v_j weighted by the criticality of the communication.*

As an example, the weighted core graph of the *Filter* application is given in Figure 5. The edge weights are a function of the criticality of the stream (which depends on the application characteristics) and the amount of traffic communicated in the stream. The value of the weights depends on how critical are the streams and on the number of classes of streams. In this work, we assume two classes of streams: non-critical and critical and weigh the critical streams by a factor of 10 compared to non-critical streams. Other approaches such as weighing a stream based on the amount of slack permitted for the stream (as used in [17]) can also be used.

Definition 2 *In $G(V, E)$, the traffic flow from each source v_i to each destination v_j, $\forall i, j \in V$ is represented by the set $T_{i,j}$. Each $T_{i,j}$ comprises of $M_{i,j}$ bursts, with each burst $b_{i,j,k}, \forall k \in M_{i,j}$, having a burst length of $blen_{i,j,k}$ cycles and a latency window of $blat_{i,j,k}$ cycles.*

In the above example, the traffic flow between the *Filter* and the *ARM* (v_3 and v_2) is represented by the set $T_{3,2}$. The set $T_{3,2}$ consists of 1 burst (we assume such a small sampling window for illustrative purposes), with $M_{3,2}$ equal to 1, $blen_{3,2,1}$ equal to 100 cycles and $blat_{3,2,1}$ equal to 400 cycles. The latency window, $blat_{i,j,k}$ is the deadline (or slack) that is permissible for the burst, which is obtained from the initial simulation of the application and the application characteristics.

The $\rho_{i,j}$ values of the regulator for each source v_i to destination v_j is obtained by:

$$\rho_{i,j} = max_{\forall k \in M_{i,j}} \left(\frac{blen_{i,j,k}}{blen_{i,j,k} + blat_{i,j,k}} \right) \forall i, j \; s.t. \; e_{i,j} \in |E| \quad (1)$$

Definition 3 *The bandwidth constraint graph $CG(Q, R)$ is a directed graph where the vertex and edge sets are equal to the vertex and edge sets of $G(V, E)$ but with edge weights $r_{i,j}$ equal to $\rho_{i,j} \times PacketSize/Cycletime$, $\forall i, j \in \; s.t. \; r_{i,j} \in |R|$.*

```
Mapping_and_physicalplanning(G,CG,P)
{ obtain an initial greedy mapping of G onto P;
  obtain greedy mappings of higher dimensional
  topologies onto 2D plane;
  In each iteration of the robust tabu search  perform
  {  for all i and j in U of topology graph
     {  generate current move by swapping the cores
        in positions i and j ;
        if current move is not tabu
        {  compute routes based on routing function;
           Simultaneously perform following steps
           in physical planning using MILP:
              Opimize design area, power or hop delay;
              Compute the core and switch sizes and positions;
              Compute number of buffers;
           Check whether QoS constraints are satisfied;
           Update tabu list and check whether aspiration
           condition is met;
           if current iteration's cost is better than best cost
           update the best cost and best solution;
        }
     }
  }
```

Fig. 7. Mapping and Physical Planning Algorithm

The bandwidth constraint graph for the above example is given in Figure 6. The edge weights in the graph are $\rho \times packetsize/Cycletime$ values for the corresponding traffic flows. When calculating the ρ values, we neglect the network latency as it is of the order of tens of cycles (refer section VI-IIB), while burst lengths and latency windows are of the order of hundreds of cycles.

The NoC topology is defined by the adjacency information of nodes in the topology and by the capacity of the links. Formally the NoC topology is defined as:

Definition 4 *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 application cores onto an NoC architecture is defined by the one-to-one mapping function:

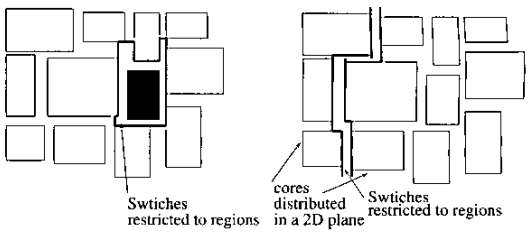
$$map : V \rightarrow U \text{ , s.t. } map(v_i) = u_j, \forall v_i \in V, \exists u_j \in U \quad (2)$$

Each link in the mapped NoC should satisfy the bandwidth constraints corresponding to the constraint graph $CG(Q, R)$. The design objective (area, power or hop delay) of a mapping is obtained from the physical planning of the mapping. This ensures that the heterogeneity in the size of the cores and network components is taken into account for accurate estimation of the design objectives.

VI. Mapping and Physical Planning Algorithm

In this section we present the algorithm for mapping and physical planning. The general problem of embedding one graph into another is intractable and is a special case of the *Quadratic Assignment Problem (QAP)* [26]. *QAP* is well studied in the literature with many heuristic algorithms available [27]. In [27], *robust tabu search* is shown to be most effective for many classes of QAP and we use this to solve our mapping problem. The general structure of the mapping-physical planning algorithm is shown in Figure 7.

In the first step an initial greedy mapping of the cores onto the topology is obtained. We also assume a greedy mapping



(a) Direct topologies (b) Indirect topologies

Fig. 8. Switch Position Restriction for direct and indirect topologies

of higher dimensional topologies (such as hypercube) onto the 2D plane. Then for each iteration of the *robust tabu search*, we perform the following computations:

- Compute the routes for the traffic flowing between the cores, based upon the routing function chosen from the library. Details of the algorithms for route computation and their implementation for different topologies are presented in our earlier works [20],[21].
- Physical planning for this mapping. This includes computing the positions of the cores and the switches, sizes of the switches & soft cores and automatic computation of switch buffers needed for the application. These steps are explained in detail in the next section.
- Check whether the mapping satisfies the delay/jitter and area constraints. For delay constraints, the links in the NoC should support the traffic through them, which is determined by the (σ,ρ) regulator values as explained in sections IV and V. We also check whether the real-time constraints for the critical streams are met by checking whether the hop delay for the streams are lower than the required value, which is obtained from the application characteristics.

In each step of the tabu search we try to optimize the design objective (area, power or hop delay) satisfying the QoS and criticality constraints. The area and power values are obtained from physical planning of that particular mapping. The parameters of the tabu search (such as the *size of tabu list, aspiration function computation*, etc.) are chosen as explained in [27]. This tabu search is applied to all topologies in the library. Our library currently has *mesh, torus, hypercube, Clos and butterfly* topologies, while other topologies can be easily added to the library. The best topology is selected and the switches and links are optimized to match the application characteristics. In this step, redundant switch ports and links (i.e. the links that don't carry any traffic and the corresponding switch ports) are eliminated. The links are sized (by changing the bit-width of the links or frequency of operation) according to the traffic flowing through them. After this network optimization step, the SystemC design of the NoC is generated automatically using the ×pipesCompiler. All these steps of topology mapping, physical planning, topology selection, optimization and instantiation are seamlessly integrated, so that all these processes are completely automated [1].

VII. Physical Planning

We use a *Mixed Integer Linear Program (MILP)* based physical planning algorithm. An MILP based physical planning for minimizing area, power of a design is presented in [24]. We modify this approach for NoCs by considering NoC specific features such as switch positioning, switch buffer calculation, etc.

[1] We also provide a feature that allows the user to interact in each of the phases if manual intervention is desired.

TABLE II
DESIGN AREA FOR VIDEO APPLICATIONS

Appln	Area-1 sqr mm	Area-2 sqr mm	Ratio
VOPD	20.25	18.01	1.12
MPEG	36	20.25	1.19
PIP	20.25	10.565	1.92
MWA	33	25	1.32
Avg	-	-	1.39

As the cores are pre-designed components, we assume the area and power values of the cores as an input. We also assume the type of the core (hard or soft) and aspect ratio constraints as an input. We use area, power libraries for various configuration of switches that are developed in [18],[21].

For a given mapping, the relative position of the cores with respect to each other is obtained from the tabu search, but the relative position of the switches is unknown. The switches in a direct topology (such as mesh, torus, hypercube) can be placed anywhere around the core to which it is connected. An important constraint to be considered in the MILP is that the switches and the cores should not overlap each other. If the switch positions are not restricted to a small region around the core, solving this overlap calculation as an MILP will be time consuming for large problem sizes (for > 20 cores). To allow scaling of the algorithm, we restrict each switch to lie in a region of adjacent cores surrounding the core to which it's connected (refer Figure 8(a)). By restricting the switch positions to a small region, the overlap calculations are several orders of magnitude faster and are scalable for large problem sizes. The solution obtained in this scheme, for all our simulations, are within 1% from the solution obtained without restricting switch sizes as the switch position tends to be close to the core to which it's connected.

For the indirect topologies (such as the Clos and butterfly), we distribute the switches along the cores in a 2D plane, based on their connectivity to the cores and to other switches (refer Figure 8(b)). Here again we restrict switch locations to lie with in certain regions as shown in the figure. Then during each step of the tabu search, we compute the actual positions of the switches and cores.

During the physical planning, we also compute the buffering needed at each switch. We assume that the links are pipelined with the number of pipeline stages depending upon the link length. For wormhole (or virtual channel) based switches with credit based flow control, for maximum throughput, the number of buffers in the switches should be equal to $2N + M$, where N is the number of pipeline stages in the link and M is the delay incurred for credit processing at the upstream and downstream switches [25]. As the switch size (power) depends on the number of buffers, we integrate this as a constraint in the MILP by breaking down the switch area (power) as a sum of buffer area (power) and crossbar (including logic) area (power). The buffer area (power) is a function of link length and is automatically calculated during physical planning.

VIII. EXPERIMENTS AND CASE STUDIES

A. Effect of Physical Planning

In this sub-section we investigate the effect of combined mapping and physical planning applied to a variety of video applications. We consider four different video applications: *Video Object Plane Decoder* (VOPD-12 cores), *MPEG4 decoder* (mapped onto 12 cores), *Picture-In-Picture* application (PIP-8 cores), *Multi-Window Application* (MWA-14 cores). We assume that our design objective is to minimize design area subject to delay/jitter and criticality constraints. We consider two schemes: in the first scheme the mapping and physical planning phases are done separately (as in past works) and in the second scheme we use our integrated approach to mapping and physical planning.

The design area for the video applications as obtained for both the schemes are presented in Table II. On an average we have 1.4× area savings in our approach.

B. Design for QoS Guarantees

In traditional design methodology, QoS can be guaranteed by designing the network to support the worst-case bandwidth needs of the application. Such a worst-case design approach, however, leads to an over-design of the network components. By using (σ, ρ) traffic regulation methodology for NoCs presented in this paper, the network components are designed optimally to support the QoS constraints of the application.

As an example, for the *DSP Filter* application (Figure 3), the minimum bandwidth needed (assuming minimum-path routing) for our design methodology is 5× lower than a worst-case design approach. Moreover, in our design methodology, the network is made to operate at very low contention, thereby reducing contention delay and power. Figure 9 shows the packet latency as obtained from the actual simulations of the *DSP Filter* application. In the first case, the links are designed to handle the average traffic through them. As the traffic is bursty in nature, such a design approach leads to high network contention resulting in large packet latency. In the second case, the links are designed with our design methodology. The average latency is almost equal to the worst-case design approach (case 3) where the network components are over designed. As our design methodology for traffic regulators is based on initial simulation, it is static in nature and doesn't capture dynamic variations in the input data streams. But for many SoC applications, the traffic characteristics don't vary a lot with the input data [14], [15]. Thus our design methodology incurs only slight increase in latency (around 10%) due to dynamic changes in data when compared to the worst-case design approach.

C. VOPD Design

In our earlier work [21], butterfly topology was found to be the best topology for VOPD. However, the design approach was based on average case analysis without considering QoS guarantees. In this sub-section, we explore VOPD mapping and physical-planning with QoS guarantees. We assume a conservative link bandwidth of $2GB/S$.

The bandwidth constraint graph for the VOPD application, based on the traffic characteristics and QoS needs of the application is presented in Figure 10. For minimum-path mapping, the minimum bandwidth needed to support the application is $2.4GB/S$ and can't be supported by any of the topologies. So we apply split-traffic routing, spreading the traffic between the cores across multiple paths. As a butterfly network has no path diversity (only one path from any source to any destination) [25], it can't support the traffic requirements of the application. All other topologies produce feasible mappings with split-traffic routing. We assume that the objective is to minimize power consumption of the design, satisfying QoS and area constraints. Figure 11 shows the power consumption of the topologies. Mesh has the least power and is the best topology for VOPD for the chosen design objective.

D. Buffer Sizing and Network Optimization

During physical planning, the number of buffers needed for the switches is automatically computed based on the link lengths (refer section VII) and this is integrated into the area (power) calculations of the physical planner. When the number of buffers is lower than the required number, throughput of the network is low. On the other hand, when the number of buffers is more than needed, the throughput remains the same, but switch area and power are increased. As an example, let us consider a homogeneous 16-node torus NoC in which each link has 4 pipeline stages. Let us assume that the credit processing delay (the M value) is 2 cycles, which is typical for most credit-based switches. Figure 12 shows the throughput

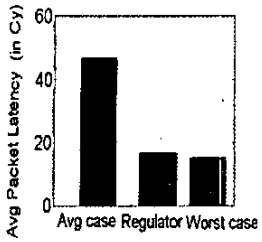


Fig. 9. Avg. Latency for *DSP*

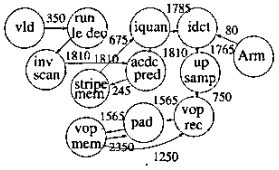


Fig. 10. Bandwidth constraint graph for VOPD with bandwidth in MB/S

Topol.	Power (in mW)
Bfly	No mapping
Mesh	542
Torus	930
Hyp.	960
Clos	753

Fig. 11. VOPD Design

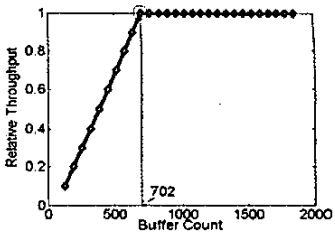


Fig. 12. Throughput vs. buffer count

TABLE III
NETWORK OPTIMIZATION

Component	Savings
Buffers	2.2×
Wire count	3.77×
Ports	1.6×

dependence on the total number of buffers in the switches for the NoC. As seen, the relative throughput increases till the optimal count of 702 buffers, after which it remains constant. With our buffer computation methodology, the physical planner automatically computes this optimum number of buffers needed to support maximum throughput. Note that in a heterogeneous SoC, the number of buffers can be different for different switches and even different for different inputs of the same switch as the link lengths are non-uniform in nature. Even in this case, the physical planner automatically computes the optimum number of buffers needed at each input of the switch based on the corresponding link lengths. For the VOPD application, compared to an average-case design (where all the switches have the same number of buffers) we get 2.2× reduction in buffer count in our scheme.

After the topology selection phase, the network components (switches and links) are optimized based on the traffic flowing through them. The links and switch ports that don't carry any traffic are removed. Other links and switches are optimized to match the traffic rate through them by changing the bit-width of the links. The effect of network optimization on VOPD design is reported in Table III.

For all our experiments, the mapping and physical planning phases are executed in few minutes on a 1GHz SUN workstation and the algorithms are scalable for hundreds of cores.

IX. CONCLUSIONS AND FUTURE WORK

Networks on Chip (NoC) based communication architectures are needed to handle the heavy communication demands of future SoCs. SoCs are aggressively designed with mostly heterogeneous processor/memory cores to maximize system performance. The mapping of such heterogeneous cores onto NoCs, physical planning (computing position and size of the cores and network elements), topology selection, topology optimization and instantiation are important phases in designing application-specific NoCs. In this work we have presented a design methodology that automates all these steps. Mapping and physical planning phases are integrated together, resulting in better NoC design (up to 2× area improvement) than traditional methodology that decouples the phases. Another important design consideration for NoCs is to guarantee Quality-of-Service (QoS) for the application. In this work we have presented a methodology for guaranteeing QoS during mapping/physical planning phase, considering the burstiness of traffic, satisfying delay/jitter constraints and real-time constraints for the traffic streams. Our design approach results in large bandwidth savings (up to 5×) and network component savings compared to traditional design approaches. In future, we plan to enhance the methodology for guaranteeing QoS to take dynamic variations in the input streams.

X. ACKNOWLEDGEMENTS

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

REFERENCES

[1] W.J.Dally, B.Towles, "Route Packets, not Wires: On-Chip Interconnection Networks", DAC 2001, pp. 684-689, Jun 2001.

[2] L.Benini and G.De Micheli, "Networks on Chips: A New SoC Paradigm", IEEE Computers, pp. 70-78, Jan. 2002.

[3] F.Karim et al., "On-chip communication architecture for OC-768 network processors", Design Automation Conference, June 2001.

[4] S.Kumar et al., "A network on chip architecture and design methodology", ISVLSI 2002, pp.105–112, 2002.

[5] P.Guerrier, A.Greiner,"A generic architecture for on-chip packet switched interconnections", DATE 2000, pp. 250-256, March 2000.

[6] 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.

[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] A.Jantsch, H.Tenhunen, "Networks on Chip", Kluwer Academic Publishers, 2003.

[11] S.J.Krolikoski et al., "Methodology and Technology for Virtual Component Driven Hardware/Software Co-Design on the System-Level", ISCAS, June 1999.

[12] 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.

[13] A.Pinto et al., "Efficient Synthesis of Networks on Chip", ICCD, Oct 2003.

[14] W.H.Ho, T.M.Pinkston, "A Methodology for Designing Efficient On-Chip Interconnects on Well-Behaved Communication Patterns", HPCA, pp.377-388, Feb 2003.

[15] J.Hu, R.Marculescu,"Energy-Aware Mapping for Tile-based NOC Architectures under Performance Constraints", ASP-DAC 2003, Jan 2003.

[16] J.Hu, R.Marculescu, "Exploiting the Routing Flexibility for Energy/Performance Aware Mapping of Regular NoC Architectures", Proc. DATE 2003, March 2003.

[17] K.Lahiri, A.Raghunathan, S.Dey, "System-Level Performance Analysis for Designing On-Chip Communication Architectures", IEEE TCAD, vol.20,no.6,pp.768-783,June 2001.

[18] A.Jalabert et al., "xpipesCompiler: A Tool for Instantiating Application Specific Networks on Chip", vol. 2, pp. 20884-20890, Proc. DATE 2004.

[19] A.Radulescu et al., "An Efficient On-Chip Network Interface Offering Guaranteed Services, Shared-Memory Abstraction & Flexible Network Programming", DATE 2004.

[20] S.Murali, G.De Micheli, "Bandwidth Constrained Mapping of Cores onto NoC Architectures", vol. 2, pp. 20896-20902, Proc. DATE 2004.

[21] S.Murali, G.De Micheli, "SUNMAP: A Tool for Automatic Topology Selection and Generation for NoCs", pp.914-919, DAC 2004.

[22] G. Varatkar, R. Marculescu, "On-chip Traffic Modeling and Synthesis for MPEG-2 Video Applications", IEEE Trans. on VLSI, Vol.12, No.1, Jan. 2004.

[23] T.T.Ye, G.De Micheli, "Physical Planning for Multiprocessor Networks and Switch Fabrics", ASAP, 2003.

[24] J.G.Kim, Y.D.Kim, "A linear programming-based algorithm for floorplanning in VLSI design ", IEEE TCAD, pp. 584 -592, Vol. 22, Issue: 5 , May 2003,

[25] W.J.Dally, B.Towles, "Principles and Practices of Interconnection Networks", Morgan Kauffman Publishers, 2003.

[26] M.R.Garey, D.S.Johson, "Computers and Intractability: A Guide to the Theory of Np-Completeness", W.H.Freeman Publishers, 1979.

[27] E.D.Taillard, "Robust tabu search for the quadratic assignment problem", Parallel Computing 17, pp. 443-455, 1991.