Bandwidth-Constrained Mapping of Cores onto NoC Architectures

Srinivasan Murali, Giovanni De Micheli
Computer Systems Lab
Stanford University
Stanford, California 94305
{smurali, nanni}@stanford.edu

Abstract

We address the design of complex monolithic systems, where processing cores generate and consume a varying and large amount of data, thus bringing the communication links to the edge of congestion. Typical applications are in the area of multi-media processing. We consider a mesh-based Networks on Chip (NoC) architecture, and we explore the assignment of cores to mesh cross-points so that the traffic on links satisfies bandwidth constraints. A single-path deterministic routing between the cores places high bandwidth demands on the links. The bandwidth requirements can be significantly reduced by splitting the traffic between the cores across multiple paths. In this paper, we present NMAP, a fast algorithm that maps the cores onto a mesh NoC architecture under bandwidth constraints, minimizing the average communication delay. The NMAP algorithm is presented for both single minimum-path routing and split-traffic routing. The algorithm is applied to a benchmark DSP design and the resulting NoC is built and simulated at cycle accurate level in SystemC using macros from the `×pipes` *library. Also, experiments with six video processing applications show significant savings in bandwidth and communication cost for NMAP algorithm when compared to existing algorithms.*

Keywords: Systems on Chips, Networks on Chips, cores, mapping, bandwidth, routing.

1 Introduction

Present and future *Systems on Chip* (SoC) are designed using preexisting components such as processors, DSPs, memory arrays [1], which we call *cores*. The use of standard hardwired busses to interconnect these cores is not scalable. To overcome this problem, *Networks on Chips* (NoCs) have been proposed and used for interconnecting the cores [2, 3, 4] and replacing dumb physical routing. The use of on-chip interconnection network has several advantages, including better structure, performance and modularity.

In several application domains, such as multi-media processing, the bandwidth requirement between the cores in SoCs is increasing. The aggregate communication bandwidth between the cores is in the GBytes/s range for many video applications. In the future, with the integration of many applications onto a single device and with increased processing speed of cores, the bandwidth demands will scale up to much larger values [3]. As an example of a media processing application, a Video Object Plane decoder [7] is shown in Figure 1. Each block in the figure corresponds to a core and the edges connecting the cores are labeled with bandwidth demands of the communication between them. As seen from the figure, the bandwidth demands are in the order of hundreds of MBytes/s.

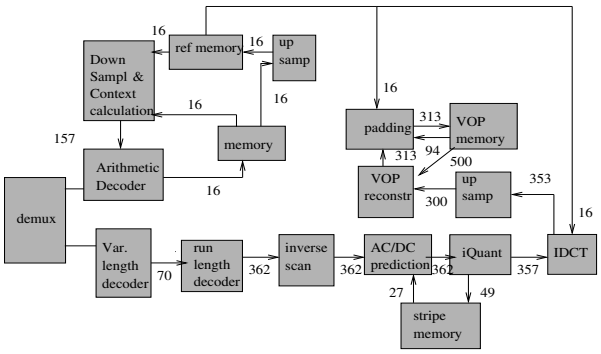


Figure 1. Block diagram of Video Object Plane Decoder, with communication BW (in MB/s).

Networks on Chip can be designed in different ways, according to the network architecture and protocol choice. In this paper, we limit our considerations to *mesh/torus* networks and to *packet-switched* data transmission. Note that our techniques are not limited to mesh topologies, but we stick to this restriction to be more specific in this paper. Packet switching leads to better link utilization. Nevertheless, packet switching with single-path deterministic routing places high bandwidth demands on the network links. By allocating higher bandwidth across the links of the NoC, more energy is dissipated. Thus, it is important to balance the bandwidth needs across the different links. In this paper we apply packet switching in both single and multi-path routing, where the traffic between two end-nodes is split across many paths.

Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE'04)
1530-1591/04 $20.00 © 2004 **IEEE**

The overall objective of this research is to show how to automatically map cores to a network architecture. In particular, we consider mesh/torus topologies, and the mapping of cores to their cross-points. We describe a mapping algorithm called NMAP that satisfies the bandwidth constraints of the NoC and minimizes the average communication delay. The algorithm supports both single-minimum-path routing and split-traffic routing. The mapping of cores is done at a high level of abstraction (based on average traffic between the cores), so that fast exploration of the design space can be performed. The SystemC code generated by the tool can then be simulated to accurately evaluate the chosen mapping. To validate our method, we apply the NMAP algorithm to a DSP system designed in SystemC and we construct the network around the cores using macros from the `×pipes` library [9] that has parameterizable SystemC components for network elements. A cycle accurate simulation of the resulting NoC architecture validates our design approach. We also apply the NMAP algorithm to six other video processing applications, which show significant reduction in bandwidth requirements and communication costs when compared to existing realizations.

2 Previous Works

We refer the reader to several recent surveys [2, 3, 4, 5] on NoCs for pointers to recent research and development. This paper deals with a specific graph embedding problem, which is intractable [10]. The mapping of clusters onto the physical topology of processors has been studied in the field of parallel processing [11, 12]. In [12], PMAP, a two-phase mapping algorithm for placing clusters onto processors is presented. The mappings produced by the PMAP algorithm are shown to have lower communication costs than mappings with previous algorithms.

The mapping of cores onto NoC architecture presents new challenges when compared to the mapping in parallel processing. A major difference is that the traffic requirements on the links of a NoC are known for a particular application, thus the bandwidth constraints in the NoC architecture need to be satisfied by the mapping.

In [8], a branch and bound algorithm is proposed that maps cores onto a tile-based NoC architecture satisfying the bandwidth constraints and minimizing the total energy consumption. In our approach, we consider the mapping problem together with the possibility of splitting traffic among various paths, thus easing the satisfaction of bandwidth constraints and providing a more efficient solution.

3 Methodology

As a starting point we assume to have an application that needs to be mapped onto a SoC populated by cores. Next we assume that the application has parallel kernels, and that the kernels have been associated with processing cores. By means of static analysis or simulation, it is possible to determine the average/mean size of the messages exchanged among cores and their frequency. Our problem is to map the cores onto a mesh NoC, so that the links support the desired message transfer. This paper describes only this important problem, that is formalized in the next section. The parallelization of the application, and the assignment of kernels to processors, can be done with known methods (e.g. [6]) and are not described in this paper.

4 Mathematical Formulation of the Mapping Problem

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 core graph of the decoder in Figure 1 is shown in Figure 2(a) and the NoC graph for a 16-node mesh is shown in Figure 2(b). The mapping of the core graph $G(V, E)$ onto the processor graph $P(U, F)$ is defined by the one-to-one mapping function map:

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

The mapping is defined when $|V| \leq |U|$. An example mapping of the decoder is shown in Figure 2(c). 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:

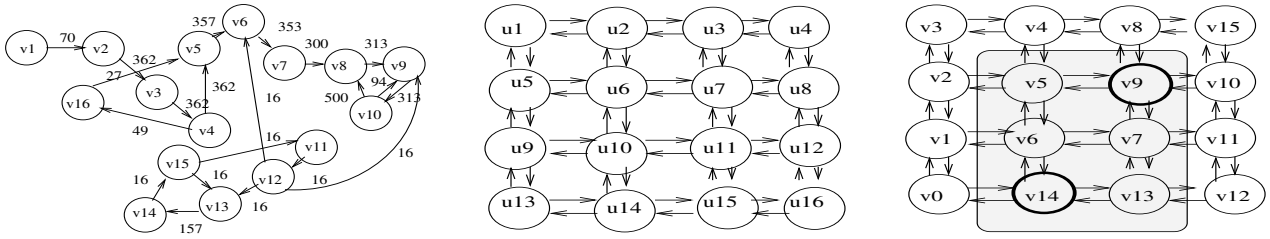
$$D = \left\{ \begin{array}{l} d^k : vl(d^k) = comm_{i,j}, k = 1, 2, \cdots |E|, \forall e_{i,j} \in E, \\ \text{with } source(d^k) = map(v_i), dest(d^k) = map(v_j) \end{array} \right\} \tag{2}$$

The bandwidth constraints are represented by the inequality:

$$\sum_{k=1}^{|E|} x_{i,j}^k \;\; \leq \;\; bw_{i,j}, \; \forall i, j \in 1, 2, \cdots, |U| \tag{3}$$

For single minimum-path routing, $x_{i,j}^k$ are obtained by the following equation:

$$x_{i,j}^k = \left\{ \begin{array}{ll} vl(d^k) & \text{, if } f_{i,j} \in Path(source(d^k), dest(d^k)) \\ 0 & \text{, otherwise} \end{array} \right. \tag{4}$$



(a) Core graph (b) NoC graph (c) Mapping

Figure 2. Mapping of Core graph onto NoC graph

where the set $Path(a, b)$ represents the set of links that form the shortest path between the mesh nodes a and b. For multi-path routing with traffic splitting, the value of $x^k_{i,j}$ are obtained from the following set of equations:

$$\forall i \sum_{j=1}^{|Adj_i|} \sum_{k=1}^{|E|} x^k_{i,Adj_i(j)} - \sum_{j=1}^{|Adj_i|} \sum_{k=1}^{|E|} x^k_{Adj_i(j),i} = \sum_{k=1}^{|E|} flow^k \quad (5)$$

where

$$flow^k = \begin{cases} vl(d^k) & \text{, if } source(d^k) = u_i \\ \text{-}vl(d^k) & \text{, if } dest(d^k) = u_i \\ 0 & \text{, otherwise} \end{cases} \quad (6)$$

and Adj_i is the set of adjacent mesh nodes of u_i. Equation 5 represents the conservation of flow, i.e. the sum of the flow coming into a node and sourced by the node equals the sum of the flow going out of the node and sinked by that node.

5 Mapping with Minimum-Path Routing

In this section, we present the mapping algorithm that uses minimum-path routing between the cores in the mesh architecture. As the problem is intractable, we use a heuristic approach that has three phases: an initialization phase that computes an initial mapping, followed by the second phase, where minimum path computations are performed. In the last phase, the initial solution is iteratively improved by invoking the second phase for each pair-wise swapping of vertices. In the `initialize()` routine, the core that has maximum communication demand is placed onto one of the mesh nodes with maximum number of neighbors. Then for each core yet to be mapped, the core that communicates most with the already mapped cores is selected. The core is placed onto the mesh node that minimizes the communication cost with mapped cores. This best mesh node is obtained by examining every available node in the mesh. The procedure is repeated until all the cores are mapped. The `shortestpath()` routine performs the minimum routing. The commodities are sorted in decreasing order of the value of their flows. For each commodity, a quadrant graph is formed between the source and destination of the commodity, as the shortest path between the source and destination lies within the quadrant between them. The shaded region in Figure 2(c) is an example quadrant graph for the commodity with source `v14` and destination `v9`.

initialize($G(V,E)$,$P(U,F)$){
  initialize $Placed(W,H)$ to ϕ;
  assign the vertex with max communication
  requirements in $G(V,E)$ to $maxs$;
  assign the vertex with maximum neighbors in
  $P(U,F)$ to $maxt$;
  $map(maxs) = maxt$;
  remove $maxt$ from P, $maxs$ from G and add
  $maxt$ to $Placed(W,H)$;
  While($|V| > 0$){
    assign the vertex in G with maximum comm
    with $\forall w_i \in W$, to $nexts$;
    for $\forall u_j \in P(U,F)$ and $w_i \in Placed(W,H)$
      $commcost(u_j) += comm_{nexts,map^{-1}(w_i)}$
      $\times[xdist(w_i,w_j) + ydist(w_i,w_j)]$;
    assign u_j with minimum cost to $nextt$;
    $map(nexts) = nextt$;
    remove $nextt$ from P, $nexts$ from G and
    add $nexts$ to W;}
  return(map,$Placed(W,H)$);}

Then, Dijkstra's shortest path algorithm is applied 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 constraints in Inequality 3 are satisfied, the cost of communication is calculated. The communication cost is given by:

$$commcost = \sum_{k=1}^{|E|} vl(d^k) dist(source(d^k), dest(d^k)) \quad (7)$$

where $dist(a, b)$ is the minimum number of hops between nodes a and b. This is a heuristic procedure to find the minimum paths in the NoC. Finding the shortest paths can also be formulated as an Integer Linear Program (ILP), but the time taken by the ILP is of the order of minutes (the above procedure completes in

few seconds and the solution obtained is experimentally observed to be within 10% of the solution from ILP).

shortestpath($Placed(W,H)$){
initialize edge weights of $Placed$ with total comm
BW for adj nodes and $maxvalue$ for others;
sort commodities in D with decreasing comm costs;
for each $d^k \in D$ do{
make quadrant graph $Q(d^k)$ with $source(d^k)$
and $dest(d^k)$ as end vertices;
$Path(source(d^k), dest(d^k))$ = minpath($Q(d^k)$);
increase edge weigths for edges in $Path$ by $vl(d^k)$;}
if BW constraints are satisfied, find the
comm cost and store it in $cost$;
else assign $maxvalue$ to $cost$;
return($cost$);}

The routine `mappingwithsinglepath()` finds the best mapping obtained by pair-wise swapping of vertices, invoking `shortestpath()` routine $O(U^2)$ times. The worst-case computational complexity of the entire algorithm is $O(|U|^3 E \log |F|)$.

mappingwithsinglepath($G(V,E), P(U,F)$){
$S(A,B) = makeundirected(G(V,E))$;
initialize($S(A,B), P(U,F)$);
$bestcommcost$ = shortestpath($Placed(W,H)$);
assign Placed to Bestmapping;
for i = 1 to $|U|$ do{
for j = i+1 to $|U|$ do{
assign $Placed(W,H)$ to $Ptemp(tW,tH)$
swapping vertices w_i and w_j;
find $commcost$ = shortestpath($Ptemp(tW,tH)$);
if ($commcost < bestcommcost$) assign $Ptemp$ to
$Bestmapping$ and $commcost$ to $bestcommcost$;}
assign $Bestmapping$ to $Placed$;}
return($bestcommcost$,$Bestmapping$);}

6 Mapping With Traffic Splitting

In this section, we present the NMAP algorithm splitting the traffic across multiple paths between the source and destination for each commodity. In the first phase of the algorithm, an initial mapping is obtained using the `initialize()` routine presented in Section 5. In the next phase, mappings obtained by pairwise swapping of vertices are evaluated to get a mapping that satisfies the bandwidth constraints. Once such a mapping is obtained, mappings with pairwise swapping of vertices are evaluated to find the mapping with the best cost.

In order to obtain a mapping that satisfies bandwidth constraints, the following set of Multi-Commodity Flow (MCF) equations are evaluated. The MCF1 are used to obtain a feasible mapping from the set of mappings obtained by pairwise swapping of vertices.

MCF1:

$$\text{min:} \sum_{i=1}^{|U|}\sum_{j=1}^{|U|} s_{i,j}$$

$$\text{s.t} \sum_{k=1}^{|E|} x_{i,j}^k \; - \; s_{i,j} \;\; \leq \;\; bw_{i,j}, \; \forall i,j \in 1,2,\cdots,|U|$$

$$\forall i \;\; \sum_{j=1}^{|Adj_i|}\sum_{k=1}^{|E|} x_{i,Adj_i(j)}^k \; - \sum_{j=1}^{|Adj_i|}\sum_{k=1}^{|E|} x_{Adj_i(j),i}^k \; = \sum_{k=1}^{|E|} flow^k$$

$$x_{i,j}^k \; \geq \; 0, \;\; s_{i,j} \; \geq \; 0 \;\; \forall i,j,k \qquad (8)$$

The $s_{i,j}$ are slack variables and signify the amount by which bandwidth constraints are violated. By reducing the sum of slack variables, mappings that reduce the amount by which bandwidth constraints are exceeded are obtained. Once a mapping that satisfies bandwidth constraints is obtained, a second set of MCF equations (MCF2) are solved in order to obtain a mapping with the best cost:

MCF2:

$$\text{min:} \sum_{i=1}^{|U|}\sum_{j=1}^{|U|}\sum_{k=1}^{|E|} x_{i,j}^k$$

$$\text{s.t.} \sum_{k=1}^{|E|} x_{i,j}^k \;\; \leq \;\; bw_{i,j}, \; \forall i,j \in 1,2,\cdots,|U|$$

$$\sum_{j}^{|Adj_i|}\sum_{k=1}^{|E|} x_{i,Adj_i(j)}^k \; - \sum_{j}^{|Adj_i|}\sum_{k=1}^{|E|} x_{Adj_i(j),i}^k \; = \sum_{k=1}^{|E|} flow^k$$

$$x_{i,j}^k \; \geq \; 0 \;\; \forall i,j,k \qquad (9)$$

where, the objective is to minimize the total communication cost, which is given by the sum of the flow of commodities through all the edges of the NoC graph. To solve these multi-commodity flow equations, we use `lp_solve` [14], a linear programming solver. The `mappingwithsplitting()` routine implements these three phases.

For SoC applications that require low jitter (the time between the delivery of adjacent packets), the traffic between the cores can be split across multiple minimum paths, instead of all paths, so that the packets traveling in the different paths have the same hop delay. To achieve this, we can restrict i,j in Equation 5 for each commodity d^k to lie within the quadrant formed by the source and destination of d^k, i.e. Equation 5 can be rewritten as:

$$\sum_{j}^{|Adj_i|}\sum_{k=1}^{|E|} x_{i,Adj_i(j)}^k \; - \sum_{j}^{|Adj_i|}\sum_{k=1}^{|E|} x_{Adj_i(j),i}^k \; = \sum_{k=1}^{|E|} flow^k,$$

$$\forall i, Adj_i(j) \in Q(d^k) \qquad (10)$$

Splitting the traffic increases the size of the routing tables at each node of the NoC. As each core communicates with only few other cores, even with split traffic routing, the

```
mappingwithsplitting(G(V,E), P(U,F)){
  S(A,B) = makeundirected(G(V,E));
  initialize(S(A,B), P(U,F));
  bestslackcost = MCF1(Placed(W,H));
  assign bestcommcost to maxvalue;
  if (bestslackcost = 0){
    assign true to bwconstsatisfied;
    bestcommcost = MCF2(Placed(W,H));
    assign Placed to Bestmapping;}
  for i = 1 to |U| do{
    for j = i+1 to |U| do{
      assign Placed(W,H) to Ptemp(tW,tH)
       swapping vertices w_i and w_j;
      if (bwconstsatisfied = false){
        find slackcost = MCF1(Ptemp(tW,tH));
        if (slackcost = 0) assign Ptemp to Placed and
         bwconstsatisfied to true;
        else if(slackcost < bestslackcost) assign slack
         cost to bestslackcost,Ptemp to Bestmapping;}
      else{
        find commcost =MCF2(Ptemp(tW,tH));
        if(commcost < bestcommcost) assign Ptemp to
         Bestmapping,commcost to bestcommcost;}}
    assign Bestmapping to Placed;}
  return(bestcommcost,Bestmapping);}
```

number of bits occupied by the routing tables is less than 10% of the total number of bits for the network buffers, a small overhead for the bandwidth savings obtained. The results of the algorithms are given in the next section.

7 Simulation Results

7.1 Experiments with Video Applications

We simulated the NMAP algorithms for core graphs of six video processing applications: *MPEG4 decoder* (mapped onto 14 cores), *Video Object Plane Decoder* (OPD-16 cores), *Picture-In-Picture* application (PIP-8 cores), *Multi-Window Application* (MWA-14 cores), *MWA with Graphics* (MWAG-16 cores) and *Dual Screen Display* (DSD-16 cores), the last four being high-end video applications [15]. We also implemented the PMAP algorithm [12], the greedy mapping algorithm (GMAP - the algorithm for UBC calculation in [8]) and the partial branch-and-bound algorithm (PBB) presented in [8][1] for comparison.

Figure 3 shows the minimum communication cost for the applications with the same bandwidth constraints for all algorithms. As seen from the figure, NMAP and PBB perform well for all applications when compared to the other algorithms. Figure 4 shows the minimum bandwidth needed to satisfy the communication bandwidth demands of the applications with dimension ordered routing for PMAP and GMAP (referred to as DPMAP and DGMAP), single

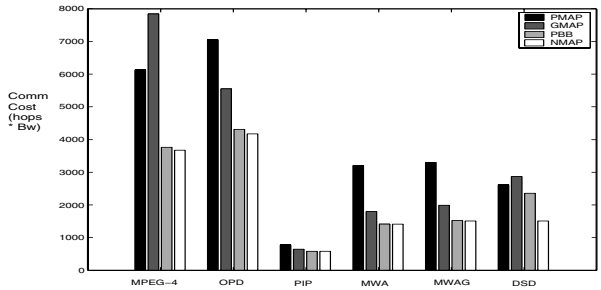


Figure 3. Communication costs for six video applications for the mapping algorithms

Table 1. Cost and BW Ratio

App	cstr	bwr
mpeg4	1.61	2.35
opd	1.35	2.41
pip	1.10	2.00
mwa	1.52	2.07
mwag	1.51	1.80
dsd	1.73	2.13
Avg	1.47	2.13

Table 2. Communication Cost Ratio

no	PBB	NMAP	rat.
25	7540	4892	1.54
35	11204	6959	1.61
45	21820	11820	1.85
55	28741	16987	1.69
65	41667	23649	1.76

minimum-path routing for PMAP, GMAP and NMAP, routing with traffic splitting across minimum paths for NMAP (NMAPTM) and routing with traffic splitting across all paths for NMAP (NMAPTA)[2]. The graph shows significant reduction in bandwidth needs with traffic splitting. The ratio of average cost and bandwidth requirements of PMAP, GMAP and PBB with the cost and bandwidth requirements of NMAP (with split-traffic routing) is given in Table 1. The NMAP algorithm results in an average of 53% savings in bandwidth needs. For same bandwidth constraints, there is 32% reduction in cost for the example applications.

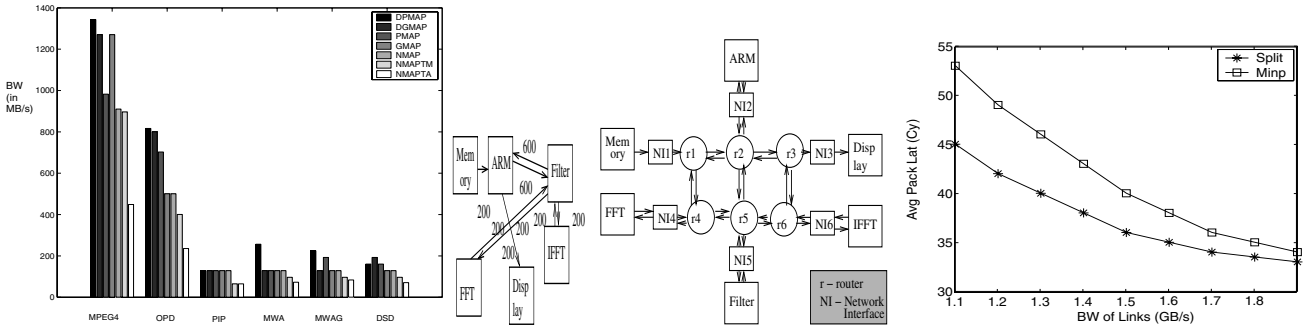
For small number of cores, PBB gives good performance, comparable to NMAP, as large part of the solution space is searched. As the number of cores scale up, NMAP produces mappings that give significant reduction in communication cost when compared to PBB. Random graphs with large number of cores (the number of cores varied from 25 to 65) were generated using the graph package LEDA [16]. Table 2 shows the savings obtained by NMAP algorithm when compared to PBB.

7.2 DSP Filter Design

We applied the NMAP algorithm to a DSP Filter design with six cores (refer Figure 5(a)). The cores are modeled in SystemC and the design is simulated at the transaction level. The resulting core graph is used by the NMAP algorithm to produce a mapping onto the mesh NoC architecture. Once the mapping is obtained, the network components (routers, links, etc) are added to the design. For

[1] We monitored the queue length as explained in [8] so that the PBB algorithm ran for few minutes.

[2] As bandwidth for NMAP and PBB is same for dimension ordered and minimum path for these examples, we only present for min path NMAP.



(a) DSP (b) NoC Impl (c) Packet Latency Vs BW

Figure 4. Bandwidth requirements for the algorithms

Figure 5. NoC Implementation of DSP

Table 3. DSP NoC Design Results

NI area	0.6 mm^2	Pack. size	64B
SW area	1.08 mm^2	minp BW	600MB/s
SW del	7 cy	split BW	200MB/s

this, we use the ×pipes library [9] that has parameterizable SystemC components for the network elements. The NMAP algorithm has an interface to ×pipesCompiler [13], so that the appropriate switches, links and network interfaces are chosen and added to the cores. The resulting NoC design (refer Figure 5(b)) of the DSP is simulated at cycle-accurate and signal-accurate level in SystemC.

The average packet latency with single path and split-traffic routing with varying link bandwidths obtained from the simulation is presented in Figure 5(c). As the traffic is bursty in nature, we have contention even when bandwidth constraints are satisfied. As seen from the figure, the average latency is higher and also increases more sharply with decrease in bandwidth for single path routing. This is because, in single minimum-path routing the congestion on links is higher when compared to the case where the traffic is split across many paths. Moreover, use of wormhole flow control results in a non-linear increase in latency (due to blocking of paths in case of contention, creating a domino-effect) with decreasing link bandwidth. The NoC design parameters are presented in Table 3.

8 Conclusions and Future Work

For efficient design of future SoCs, an automatic mapping of cores onto NoCs is highly desirable. Towards this end, we have presented a fast mapping algorithm satisfying the bandwidth constraints of a mesh NoC, minimizing the average delay. By splitting the traffic in the NoC, we obtain significant bandwidth and cost savings compared to existing realizations. Our approach is validated by cycle-accurate simulation of a DSP design modeled in SystemC to which NoC components are added from the ×pipes library. The approach can be extended to map cores onto various NoC topologies for fast and efficient design space exploration for NoC topology selection.

9 Acknowledgements

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

References

[1] W.Cesario et al., "Component-Based Design Approach for Multicore SoCs", DAC 2002, pp. 789-794, June, 2002.

[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] The Cadence Virtual Component Co-design (VCC), http://www.cadence.com/company/pr/09_25_00vcc.html

[7] 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.

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

[9] M.Dallosso et. al, "×pipes: A Latency Insensitive Parameterized Network-on-chip Architecture For MPSoCs", pp. 536-539, ICCD 2003.

[10] M.Garey, D.Johnson, "Computers and Intractability", W.H Freeman, 1979.

[11] V.Lo et al.,"OREGAMI: Tools for Mapping Parallel Computations to Parallel Architectures",Intl Journal of Parallel Programming, vol. 20, no. 3, 1991, pp. 237-270.

[12] N.Koziris et al.,"An Efficient Algorithm for the Physical Mapping of Clustered Task Graphs onto Multiprocessor Architectures", Proc. of 8th EuroPDP, pp. 406-413, Jan, 2000.

[13] A.Jalabert et.al, "×pipesCompiler: A tool for Instantiating Application Specific Networks on Chip", Proc DATE, 2004.

[14] ftp://ftp.es.ele.tue.nl/pub/lp_solve/

[15] E.G.T.Jaspers, et al.,"Chip-set for Video Display of Multimedia Information", IEEE Trans. on Consumer Electronics, Vol 45, No. 3, pp. 707-716, Aug, 1999.

[16] http://www.algorithmic-solutions.com/