MPSoC Architecture-Aware Automatic NoC Topology Design

Rachid Dafali, Jean-Philippe Diguet

MPSoC Architecture-Aware Automatic NoC Topology Design

Rachid Dafali and Jean-Philippe Diguet

European University of Brittany - UBS/CNRS/Lab-STICC dept.
BP 92116, F-56321 Lorient Cedex, FRANCE
rachid.dafali@univ-ubs.fr

Abstract. This paper presents a methodology for the automatic definition of NoC topology according to application and architecture requirements. The proposed solution, which has been implemented as a new step of our NoC design flow, results from the analysis of real concerns and demands from designers. The main contribution lies in the fact that we customize the method according to multiprocessor architecture models and associated memory organizations. A real-life H264 example has been used to compare synthesis results for NoCs generated by our tool, with automatic topology selection with well-known efficient topologies, which wer manually specified. Results clearly show the efficiency of our approach.

Keywords: Application Specific Network, Network-on-Chip, Topology Generation, Mapping.

1 Introduction

Networks-on-Chip (NoC) have recently emerged as a new generation of communication infrastructure to support communication on SoCs providing a solution to connect different IP-cores through an effective, modular, and scalable communication network.

The choice of the most suitable network topology for a particular set of applications is an important phase in designing an efficient NoC. Wrong topologies choices can dramatically affect network performances. Consequently, the topology not only impacts the efficiency of the network in terms of bandwidth and speed, but also has a major influence on the area of the network and its energy consumption.

Area cost is one of the most frequently cited drawbacks addressed to NoC solutions and the purpose of this paper is precisely to describe a new methodology that aims to minimize the final NoC area by means of a relevant topology selection. This design step is implemented within our existing NoC CAD tool in order to choose a network topology according to application constraints and architecture model.

This methodology is based on the heterogeneity of application domains and exploits the designer's knowledge on the model of his architecture including processors and memory organization to guide the topology selection. This aspect is usually ignored although it is of great importance. Actually, in real-life conditions the architecture model usually exists before the NoC has been designed for two major reasons. First, it is directly related to the application domain and secondly it depends on previous design

reuse. As a result, our method depends on the initial architecture model and then aims to minimize the final cost by topology refinements according to application constraints.

This paper is organized as follows: In section 2, a survey of the related work is done. The methodology and the design are explained in sections 3 and 4, section 5 describes results obtained with a real-life video encoder. Finally, section 6 concludes the paper and gives an overview of our future work.

2 State of the Art

To integrate and automate the choice of topology in the NoC design flow, current methods [7, 9, 11] select a generic topology among a predefined set. This approach is based on the placement of IPs on various topologies, followed by a comparison of performances to identify the best topology. The relevancy of this approach depends on many factors, such as the criteria for comparison, the number and the models of topologies compared, and the placement algorithm used.

The Stanford solution presented in [11] consists of a tool called SUNMAP, which is used for the automatic selection of the best topology and the NoC code generation. This tool places the IP blocks on several standard topologies (2D mesh, Torus, Hypercube, Butterfly, and Stage Clos); it then selects the best architecture by estimating model performances in terms of bandwidth and area. The mapping is performed through a heuristic approach formulated in [10], with an objective function, the goal of which is to minimize average communication time, as well as surface and energy consumption, while meeting bandwidth constraints.

Similarly, the simulation tool SIMOO from University of Rio Grande do Sul automatically chooses a topology among several by analyzing the communication of a given application [9]. This tool first extracts the application communication behaviour as a dynamic communication graph from high-level communication primitives and simulation results. Then, it maps this graph into different previously characterized architecture templates. Finally, it analyses and selects the architecture that best fits the application communication performance constraints.

Moreover, a similar approach is presented in [7]. This paper proposes a methodology that aims at minimizing the power consumption by selecting the optimum network topology for a specific application.

Another method for designing the best network topology is based on a hierarchical partitioning [3, 12]. In this approach, the network is divided into two or more partitions and each partition is modeled by a dedicated topology. The different partitions are then combined to form a single network topology. The efficiency of this approach depends on the choice of model for a given partition and the choice of the algorithm partitioning.

The approach in [12] customizes a specific topology for a given application by decomposing the communication requirements into a set of generic communication primitives, such as gossiping and broadcasting. Each primitive is swapped by a specific representation graph in the library. This graph structure is the pattern that the decomposition algorithm searches for when processing the input application graph. After the decomposition step, the communication primitives are replaced by their optimal implementations and finally glued together to synthesize the customized architecture.

The personalization of the network topology for a given application remains the best option to achieve the highest degree of flexibility in the NoC paradigm. This customization can be achieved by exploiting the specific communication application to design the network that meets the designer's requirements for performance, surface and energy consumption.

Our view differs from previous works, as they are directly guided by the architecture model and strongly focalized on the real design bottleneck that motivates the bandwidth requirement and consequently the need of a NoC solution, namely the concurrent memory accesses. However, our methodology is adapted to the way local, shared, and global memories are accessed. Thus, we consider that the NoC topology design flow must be based on two strongly related main stages. The first one starts with the exploration and analysis of application domain requirements. It results in Communication Dependency Graphs (CDG) with bandwidth requirements. Our tool already performs these tasks. The second one defined as an interactive step, which is necessary to integrate designer knowledges and experience. This is the specification of the architecture model in terms of memory accesses and address computation mechanisms. Area optimization is then performed along with topology design steps including clustering of non communicating IPs, load balancing and IP placement. It results in a topology that fits application and architecture requirements. This second stage is the main contribution presented in this paper.

3 Design Methodology

The heterogeneity of application domains in embedded system-on-chip makes it impossible to develop a single method for NoC topology design because the varying nature of communications means various different schemes from one domain to another. The aim of our approach is to exploit the awareness of the design about the architecture model to guide the topology selection. Consequently, we propose a new method that extracts the application communication behaviour from high-level communication primitives, analyzes and identifies the class of multiprocessor architecture and designs the topology with an automatic method customized for the class that has been chosen.

This section describes the various multiprocessor architecture classes identified and explains the topology design method developed for each one.

3.1 Multiprocessor Architecture Class Identification

To classify the multiprocessor architecture, we use Raina taxonomy [5] that extends Flynn's [4] classification by considering that multiprocessor architectures are based on two elemental features: the address space and the physical memory organization. These are the basic properties that represent a memory system and its relations with processors. Raina completed this terminology by designing the address space to be either shared or disjoint and the physical memory organization to be either shared or distributed. Therefore, this taxonomy organizes memory systems into three different classes described in Fig 1. The first class is shared address-space shared memory (SASM), which corresponds to the conventional shared memory systems. The second

class is disjoint address-space distributed memory (DADM), which corresponds to the conventional distributed memory systems. The third class is shared address-space and distributed memory (SADM).

We have chosen this terminology primarily because the the memory's access policy has a major impact on the NoC performance. Secondly, it is based on criteria that differentiate application fields. For example, the architecture of a video coding application fits the SASM class because all processors communicate with shared memories through the network. However, the architecture of a turbo-communication application belongs to the DADM class because the messages circulate between processors and the local memories are not connected to the network.

The choice of the architecture model and the component specification is made by the designer during a first interactive step implemented in the μSpider II CAD tool. Then, the associated method is applied according to the designer's choice. These methods are detailed in the following subsections.

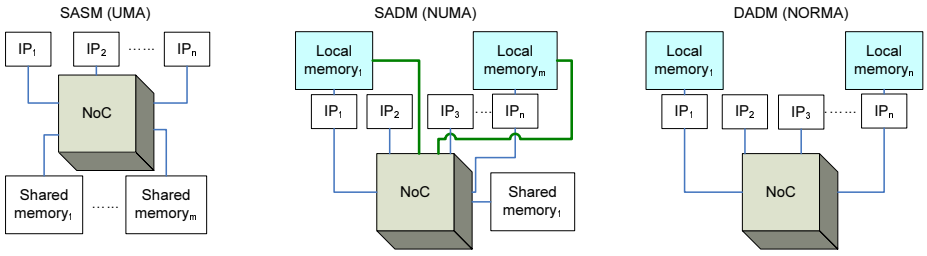

Fig. 1. Raina multiprocessor architecture classification.

3.2 SASM Topology Design Method

The SASM class is based on the Uniform Memory Access (UMA) model. In UMA architecture, all the processors share the physical memory uniformly. Usually, the SASM architecture is used to speed up the execution of a single large program in time critical applications.

The Algorithm 1 describes the method used to create a dedicated topology for a SASM class, this algorithm includes three steps:

1. First, choose the structure to map the shared memories (*SMs*). Thus, if the *SMs* are less than or equal to three, they are connected with point-to-point links. Otherwise they are mapped onto a mesh 2D topology. The *SMs* location on the mesh 2D topology is guided according to the number of common *SMs* interlocutors and the communication bandwidth required for these common interlocutors. For example, in Fig.2, a SM_4 has a PU_5 as a common interlocutor with SM_1 and SM_3 and PU_5 communications require more bandwidth with the SM_1, this implies that the SM_4 is placed closer to the SM_1.
2. In the second step, place the processing units (PUs) onto the topology. The strategy used consists in attaching to each shared memory node a star topology that connects

Algorithm 1 Topology Design for SASM class

First step: Shared memories (*SM*) mapping.
if Number of $SM \leq 3$ **then**
SM are connected by point-to-point links.
else
SM are mapped on mesh 2D topology.
end if
Second step: Processing Unit (*PU*) mapping.
if PU_x communicates with SM_y **then**
Map the PU_x on SM_y star topology.
end if
Third step: Each *PU* must belong to only one star-group
repeat
Selection of PU_x which demands the smallest bandwith.
Computation of the PU_x bandwith impact (Eq. 1) on each star-group.
Add the PU_x. to the star-group which incurs the minimum impact.
until all *PU* are not connected

the *PUs* communicating with this shared memory. This implies that a PU_x can belong to several star-groups. As illustrated in Fig.2, PU_5 belongs to three star-groups: SM_1, SM_3 and SM_4. we choose the star structure bacause is scalable, easy to set up and to expand.

3. Finally, to address the problem of the PU_x belonging to several star groups, compute the PU_x's bandwidth impact on the star-groups and then keep the PU_x in the group where it introduces the minimum amount of impact on.

   The PU's bandwidth impact is calculated using this formula:

$$Impact_{PU_x^G} = \left(\frac{(\sum_{i=0}^{N_{PU}} BW_{PU_i} * \delta_{i(G)}) - BW_{PU_x^G}}{\sum_{i=0}^{N_{PU}} BW_{PU_i} * \delta_{i(G)}} \right) * 100 \quad (1)$$

   where :

   - $Impact_{PU_x^G}$ is the impact of PU_x attached to G group.
   - N_{PU} is the total number of PUs.
   - BW_{PU_i} is the bandwidth required between the *PU* and the *SM* belonging to the star-group.
   - $\delta_{i(G)} = \begin{cases} 1 \; if \; PU_i \in G \\ 0 \; otherwise \end{cases}$

   Consequently, in the example shown in Fig.2, the PU_5 is placed definitively in the SM_1 star-group because it incurs the minimum bandwidth impact compared to the impact in the other star-groups.

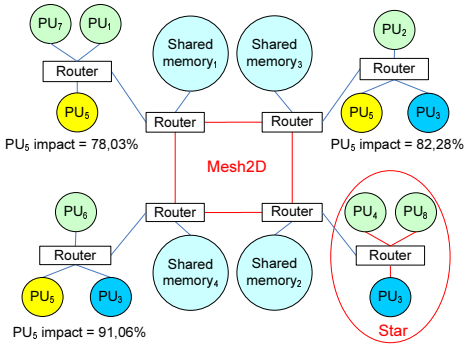

Fig. 2. SASM design topology method.

3.3 SADM Topology Design Method

The SADM class is based on a Non-Uniform Memory Access (NUMA) model. NUMA model is a memory design used in multiprocessor architecture, where the processor can access its own local memory faster than non-local memory. The non-local memory can be a local memory to another processor or a shared memory. To automatically generate a dedicated topology for the SADM class, we have also implemented a specific algorithm similar to the approach used for the SASM class.

However, the difference is located in the third step of the algorithm. Where the processors are connected to the star-groups of their local memories, even if they have the higher impact on these star-groups. Thus, we keep the specification provided by the NUMA model, while guaranteeing that the processor can access its own local memory rapidly.

3.4 DADM Topology Design Method

The memory access in the DADM class is based on the No Remote Memory Access (NORMA) model. NORMA is a memory organization used in multiprocessor architectures with distributed memories, where a processor cannot access non-local memories. To deal with this model, we use an algorithm composed of three steps to design a dedicated topology for the DADM class:

1. First step: create non-communication *PU* groups. As shown in Fig.3, the processors are grouped in 4 clusters represented by different colors and in each group the processors do not communicate.
2. Secondly, structure all non-communication *PU* groups in pairs; a pair links two nodes of a group through a router; a node represents a processor or another pair. The arrangement of the pairs are created according to the communication bandwidth required for the common interlocutors of the pair elements. In Fig.3, we illustrate the pair arrangement in the example by structuring the group represented by the yellow color.
3. Finally, connect the top router of each group on mesh 2D topology.

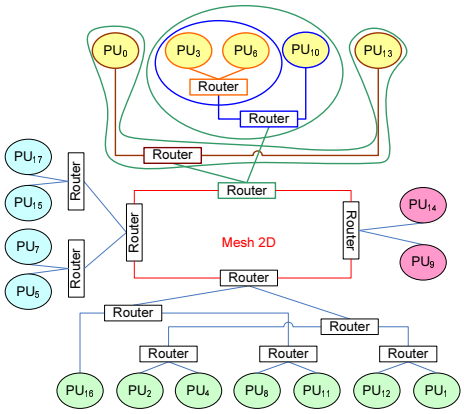

Fig. 3. DADM design topology method.

4 From Application to NoC Design

The design of NoC implies a complex set of tasks, because the NoC design flow includes several stages with different choices and complex functions. Moreover, the arrangement of these stages, their interactions, and any incompatibilities affect a large number of parameters. To address this complexity of design, we have design μSpider II CAD tool. This tool performs design space exploration and code generation. Design space exploration is implemented in an interactive way based on designer choices for arbiter, routing policies and topology selections. Then automatic procedures are available for time-consuming and error prone tasks such as Time Division Multiplexing (TDM), FIFO sizing and path allocation for guaranteed traffic management.

In the continuity of μSpider II CAD tool development [1], we replaced the choice of a generic topology with an automatic generation of specific topology based on the methodology developed in section 3. The CAD environment flow design consists of four parts:

1. The first step simulates the application and extracts the communication specification such as bandwidth, latency, and communication sequences. These specifications represent the input point of the NoC design flow. They allow for an accurate description of the performances to be delivered by the NoC. To obtain the maximum benefits from this information, we developed a method for extracting constraints based on CDG. For instance, these graphs can be generated from a specification with the cadence E language. The CDG describes the the communication scheduling. This scheduling organizes communications according to their order of execution and it can automatically extract tedious information such as:
   (a) Parallel communications.
   (b) Mutual exclusion between communications.
   (c) Synchronisation constraints.

2. Thanks to the scheduling of communications in the CDG graphs. The second step explores these graphs to calculate the real bandwidth and latency, provides parameters to generate automatically a specific topology using the methodology explained in section 3 and determine the mutual exclusive communications.
3. The third step deals with derivation of local latency and bandwidth constraints for each unidirectional communication, and computes the minimum TDM table size required for implementing guaranteed traffic communications and a minimum bandwidth for all best effort communications. Then, it computes and allocates time slots and path to each guaranteed traffic communication.
4. The last step is the VHDL code generator, some additional C APIs are also provided for interfacing NoC components with IP cores, currently the library provides interfaces compliant with the OPB bus.

5 Experimental Results

5.1 Experiments with Complex Video Decoder Application

To evaluate the potential of the topology design methodology implemented in μSpider II CAD tool for real applications, we have implemented a H.264 video decoder, provided by our industrial partner. The SoC is composed of 35 cores (28 masters and 7 slaves). The video compression standards are defined by means of profiles and levels that correspond to the configuration of the codec. A profile is usually a set of algorithms and a level corresponds to a complexity degree (e.g. pixel resolution, decoding speed, data-rate ...). H.264 standard has 14 different levels and 3 profiles: baseline (videoconferencing, wireless network ...), X (video streaming) and main (high quality and performances). Each profile can be combined with any of the different levels (e.g. resolution levels).

The estimation of bandwidth by masters and slaves have been done in the case of a high definition television (HDTV) video mode (1080i/1920/30Hz) which corresponds to the main profile and it is the maximum size running on a simulation tool over a 8ms cycle frame 5 macro-block time. The designers of our industry partner manually estimated the bandwidth requirements to $3,2 Gbits/s$ but our CDG-based analysis tool showed that these features were over-estimated. While taking into account mutual-exclusion and dependency constraints we finally saved around 20% (max: 65%) of bandwidth as shown exhaustively for all IP in Fig.4.

The H.264 video decoder application is distributed over multiple processors via data partitioning. The multiprocessor architecture platform contains 28 processors and 7 shared memories. All the processors can read and write data from/to all shared memories. Therefore, this multiprocessor architecture belongs to the SASM class as specified in our methodology explained in section 3, because all the processors share the physical memory uniformly.

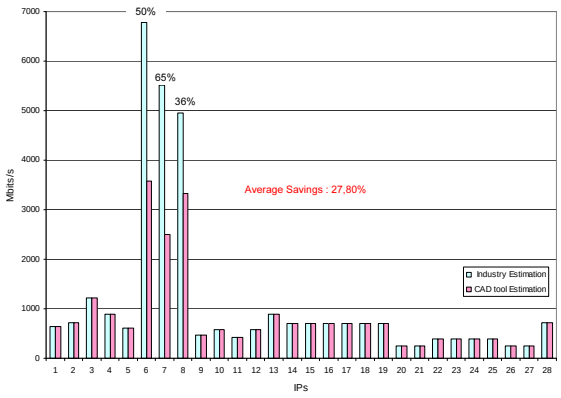

Fig. 4. H.264 bandwidth gains with mutual exclusion & dependency analysis.

5.2 Test Conditions

The experimental NoC generated by our tool has been parameterized as follows: bit width 32; Routing: street sign; End to end flow control; Round robin arbiter; and single guaranteed traffic (GT) channel with a 32 slot TDMA table. We generate four NoCs for the H.264 video decoder application with the same parameters and bandwidth and latency constraints in order to obtain the same performances. We have generated 4 NoCs with our tool. The first NoC is based on a specific topology automatically generated. And 3 other NoCs are based on the most popular NoC regular topologies, which are also used in tools addressing topology selection:

- A Mesh 2D topology connects the nodes in mesh. It is so far the most popular and exploited architecture in network on chip design. The main reason is that the routing algorithms are simple to introduce, it is scalable particularly in view of its location on silicon, and provides in theory high-speed communication, but is expensive to implement on silicon because of its large number of port for routing and connection.
- The WK-recursive topology [2] is a structure for recursively scalable networks. It offers many interesting properties such as a high degree of scalability, and symmetry, which correspond very well to a modular design of distributed systems connecting a large number of processors.
- The Spidergon topology [6] developed by STMicroelectronics is a network topology in which all nodes are arranged in ring form and each node is connected to its clockwise and its counter-clockwise neighbour. Also, each node is connected directly to its diagonal counterpart in the network. This structure allows for a simple routing scheme to minimize the number of nodes that a data packet must cross before reaching its destination. The main advantage of this topology over a conventional 2D mesh is that the network latency can be significantly reduced for the same number of nodes. The number of links required and the silicon implementation cost of the Spidergon topology are also better than the 2D mesh.

5.3 Evaluation Experiments

We compared the synthesis results obtained by the NoC based on the custom topology and the ones based on generic topologies for the same video encoder application and the same performances. On average, the area required by the generic topologies is 30% more than the area occupied by the custom topology. The gain we have obtained for the NoCs (Table1) in the worst case (Spidergon) are equal to 17%, 22%, 14% for logic, registers and memory respectively and 46%, 37%, 25% in the best case (WK-Recursive). Finally, Table 3 shows that the gains obtained for the links and routers are more signifi-

Table 1. NoC costs of various topologies.

	Synthesis results (Gain)		
	Logic (LC)	Registers (LC)	Memory (bits)
Dedicated	50512	47808	58442
Mesh 2D	69880 (27,7%)	64390 (25,8%)	70322 (16,9%)
WK-Recursive	94774 (46,7%)	76136 (37,2%)	78242 (25,3%)
Spidergon	61056 (17,3%)	61663 (22,5%)	68474 (14,7%)

cant (37%, 55%, 53% for Spidergon) than the ones obtained for the full NoC. This gap is due to the area occupied by the network interfaces (NIs) (Table 2), which do not vary as much from one scheme to another. The NI area depends on the size of buffers, which depends on the bandwidth and latency constraints and not on the kind of topology used.

Table 2. Network Interface costs of various topologies.

	Synthesis results (Gain)		
	Logic (LC)	Registers (LC)	Memory (bits)
Dedicated	33412	36578	49862
Mesh 2D	33640 (0,67%)	36599 (0,06%)	49862 (0%)
WK-Recursive	33751 (1%)	36801 (0,6%)	49862 (0%)
Spidergon	33688 (0,82%)	36583 (0,02%)	49862 (0%)

6 Conclusion and Future Work

In this paper, we have developed a new topology-based design methodology for NoC applications. The proposed methodology automatically generates a custom topology according to Raina multiprocessor architecture classification. This methodology has been integrated in our CAD tool to replace the generic topology with the automatic generation of a specific topology. Using our tool to design a specific topology for a H264 video decoder application has proven the efficiency of our approach. The results

Table 3. Router and Link costs of various topologies.

	Synthesis results (Gain)		
	Logic (LC)	Registers (LC)	Memory (bits)
Dedicated	17100	11230	8580
Mesh 2D	36240 (52,8%)	27791 (59,6%)	20460 (58,1%)
WK-Recursive	61023 (72%)	39335 (71,5%)	28380 (69,8%)
Spidergon	27368 (37,5%)	25080 (55,2%)	18612 (53,9%)

from custom topology compared to generic topologies show a 30% improvement in the average area.

We have noticed that TDMA-based network interface area is close to half of the total cost independent of NoC topology, and this is the point that has motivated our current research direction dedicated to interface optimization. We are adding a last optimization step to the topology design flow, this work focuses on local optimizations to remove resources (link, router) from the dedicated topology, that have a low utilization rate. We also plan to test our method for applications that belong to the DADM class.

References

[1] Evain, S., Dafali, R., Diguet, J-Ph., Eustache, Y., Juin, E.: μSpider cad tool: Case Study of NoC IP Generation for FPGA. In: DASIP, (2007)

[2] Della Vecchia, G., Sanges, C.: A recursively scalable network vlsi implementation. In: Future Gener. Comput. Syst. Elsevier. pp 235–243 (1988)

[3] Ahonen, T., Sigenza-Tortosa, D. A., Bin, H., Nurmi, J.: Topology optimization for application-specific networks-on-chip. In: Int. work. on System level interconnect prediction (SLIP'04). pp 53–60 (2004)

[4] Flynn, M.J.: Some computer organisations and their effectiveness. In:IEEE Trans. on Computers, (1972)

[5] Raina, S.:Virtual Shared Memory: A Survey of Techniques and Systems. (1992)

[6] Coppola, M., Locatelli, R., Maruccia, G., Pieralisi, L., Scandurra, A.: Spidergon a novel on chip communication network. In: Int. Symp. on System on Chip. pp 15–22 (2004)

[7] Elmiligi, H. Morgan, A. A., El-Kharashi, M. W., Gebali, F.: A Topology-based Design Methodology for Networks-on-Chip Applications. In: Int. Design and Test Workshop. pp 61–65 (2007)

[8] Hu, J., Marculescu, R.: Energy- and performance-aware mapping for regular NoC architectures. In: IEEE Transaction on CAD of Integrated Circuits and Systems. (2005)

[9] Kreutz, M. E., Carro, L., Zeferino, C. A., Susin, A. A.: Communication architectures for system-on-chip. In: 14th Symp. on Integrated Circuits and Systems Design. pp 14–19 (2001)

[10] Murali, S., De Micheli, G.: Bandwidth-constrained mapping of cores onto NoC architectures. In: DATE (2004)

[11] Murali, S., De Micheli, G.: SUNMAP: a tool for automatic topology selection and generation for NoCs. In DAC. pp 914–919 (2004)

[12] Ogras, U. Y., Marculescu, R.: Energy- and performance-driven NoC communication architecture synthesis using a decomposition approach. In: DATE. pp 352–357 (2005)