Grant Agreement number: 248972

Project acronym: NaNoC

Project title: “Nanoscale Silicon-Aware Network-on-Chip Design Platform”

Seventh Framework Programme

Funding Scheme: Collaborative project

Theme ICT-2009.3.2 Design of semiconductor components and electronic based miniaturised systems

Start date of project: 01/01/2010 Duration: 36 months

D 3.3 Design space exploration tool chain

Due date of deliverable: M10
Actual submission date: M13

Organization name of lead beneficiary for this deliverable: Lantiq (LTQ)
Work package contributing to the Deliverable: WP3

<table>
<thead>
<tr>
<th>Dissemination Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PU</td>
<td>Public</td>
</tr>
<tr>
<td>PP</td>
<td>Restricted to other programme participants (including the Commission Services)</td>
</tr>
<tr>
<td>RE</td>
<td>Restricted to a group specified by the consortium (including the Commission Services)</td>
</tr>
<tr>
<td>CO</td>
<td>Confidential, only for members of the consortium (including the Commission Services)</td>
</tr>
</tbody>
</table>

APPROVED BY:

<table>
<thead>
<tr>
<th>Partners</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>All partners</td>
<td>Feb 7th, 2011</td>
</tr>
</tbody>
</table>
ABSTRACT

System-on-Chip (SoC) has become a common design technique in the integrated circuits industry as it offers many advantages in terms of cost and performance efficiency. A SoC is a complex, and possibly heterogeneous, system composed by a variable amount of highly integrated components, comprising processors, caches, hardware accelerators, memories, and peripherals. The communication between those components is carried out by means of some sort of communication infrastructure. The performance of SoC designs is heavily influenced by the interconnection infrastructure implemented. While small SoC designs can employ buses as their interconnection, more complex interconnections are required as the number of cores of the design increases.

Networks-on-Chip (NoCs) were proposed as the solution to shared bus scalability issues, and have become a new paradigm for designing scalable communication infrastructures in SoCs. The main advantage of a NoC over a shared bus is its capability to reduce design costs while providing high-performance communications. Although NoCs are quickly growing in popularity, their use is not without challenges. Due to the high heterogeneity of NoC solutions, it is hard to select the optimal NoC configuration for a given design. Moreover, it is difficult to evaluate the performance of these interconnects because of their complexity. This is a potential design risk that raises the need for new tools that are able to guide the designer to the best candidates inside the design space. In order to address this challenge, early Design Space Exploration (DSE) is required to find appropriate system architectures out of many candidate architectures. Usually, early explorations are carried out by means of high level tools that provide performance estimations that the designer will use in order to select the most promising candidates at the earlier stages of the design flow. However, there is a gap between high level performance prediction and the real system performance that grows as the scale of new technologies is reduced, increasing the number of design re-spins caused by inaccurate high level predictions, that result in increased design costs. To minimize the impact of this gap, layout-aware tools must be used in each step of the design process, thus, considering the impact over performance of the physical implications of the interconnect.

This report, shows how to perform an early DSE using the Electronic System Level (ESL) performance evaluation framework SystemQ. We present a DSE tool chain for heterogeneous Multi-Processor SoC (MPSoC) that features a complex NoC as a central interconnect. It implements a high-level Place & Route algorithm capable of predicting the physical characteristics of the NoC in the earlier stages of the design process, thus reducing the number of design re-spins due to inaccurate performance predictions. Additionally, in order to optimize the interaction between the NaNoC partners, the CEF language was employed both for input and output files (see NaNoC task 3.5, deliverable 3.4). Based on the XML standard, CEF was defined by the NaNoC partners as a NoC description language with the aim of improving productivity by removing the translation costs related to the use of different files formats between partners and tools.

The work performed in this deliverable (development of the LDSET tool) was carried out in parallel with the definition of the first version of CEF. To unblock the dependency between both tasks, vendor-specific clauses (available in CEF) have been used, thus allowing the definition of
tags required by the LDSET tool. Notice LDSET is fully compatible with CEF based on the use of those vendor-specific clauses. Later, the clauses needed by LDSET will be analyzed and potentially moved out from the vendor-specific section in CEF (for further details, please read deliverable 3.4).

This deliverable is the outcome of task 3.4, focused on development of design space exploration tools. Additionally, the results from this task will be directly used in task 3.7 and indirectly influence work packages 5 and 7.
1 Introduction

Modern Systems-on-Chip (SoCs) comprise a multitude of functional blocks such as processors, hardware accelerators, peripherals, and on-chip and off-chip memories. A powerful interconnect backbone is required to fulfill the communication needs of these modules. However, an interconnect is just a mean to an end. It should, therefore, require only a little chip area, consume almost no power and it must meet all communication constraints given by the attached modules. Certainly, all of these requirements cannot be met at the same time. Hence, the design of an interconnect is at least a trade-off between performance and chip area.

Today’s SoCs comprise several dozens of modules; next-generation systems might include up to several hundreds. This makes the interconnect design cumbersome because the communication requirements of all modules must be taken into account in order to find the best interconnect design for the overall system. Major concerns for the design of an interconnect are the traffic patterns of the communication between master and slave modules (called terminals). Two main properties are especially important: throughput and latency. Traffic flows with high throughput requirements, i.e. bandwidth, need to be prioritized while not starving low-throughput flows. Low-latency flows must be scheduled quickly by the interconnect independently of their throughput requirement.

Networks-on-Chip (NoCs) are the response to the challenge of designing scalable communication infrastructures for SoCs. The main advantage of a NoC over a shared bus is its capability to reduce design costs while providing high-performance communications. In a NoC, transactions between terminals are encapsulated into packets, and delivered through a shared interconnection network. Each terminal is connected by a Network Interface (NI) to a network link, which is a point-to-point data line connected with other links via switches. A link can connect a switch with another switch or with a NI, in such a way that it exists direct or indirect connection between any two terminals.

Although NoC popularity is quickly growing, their use is not without challenges. Due to the high heterogeneity of NoC solutions, it is hard to select the optimal NoC configuration for a given design. Moreover, it is difficult to evaluate the performance of these interconnects because of their complexity. This is a potential design risk that raises the need of new tools that are able to guide the designer to the best candidates inside the design space. In order to address this challenge, early Design Space Exploration (DSE) is required to find appropriate system architectures out of many candidate architectures. Usually, those early explorations are carried out by means of high level tools that provide performance estimations that the designer will use to select the most promising candidates at the earlier stages of the design flow. However, as the number of components of a SoC increases, the size of the design space and the complexity of the candidates that compose it become too high to be handled manually by the designer.

In order to tackle all of these requirements, Design Space Exploration (DSE) can be used to find promising system architectures within a vast design space. DSE must take the traffic
requirements of the communication patterns into account. Cost functions must be applied to flows in order to minimize overall traffic costs. Furthermore, existing building blocks for the interconnect must be considered. Design constraints such as achievable clock frequencies, buffer sizes, and the bitwidth of data paths need to be incorporated. Putting all parameters together creates an incredibly large design space that needs to be explored. At the end of the exploration an optimal interconnect is available. However, in practice there is not one optimal interconnect because many constraints must be met. Thus, a number of interconnects are created via DSE that need to be evaluated extensively. Hence, in a second step abstract performance evaluation is used to assess the interconnects found via DSE in more detail by focusing on the traffic requirements. Realistic traffic patterns must be applied to stress the system and dependencies among flows should be considered, namely contention at arbiters and congestion at frequently used resources. Additionally, pure high-level approaches for NoC performance predictions have been demonstrated to be inaccurate, misleading designers towards non-optimal, or even non-valid, solutions. Those errors must be corrected at later stages in the design flow, forcing expensive design re-spins that will negatively impact the design cost of the system. In order to reduce the number and impact of design re-spins, the accuracy of high-level performance predictions must be improved by taking into account the impact of the physical implementation over the NoC performance.

The outcome of the evaluations will be the best interconnect for a given number and type of terminals as well as for a given traffic scenario, i.e. a given application. This is typically hard to achieve as the interconnect cannot satisfy all communication requirements to the same extent. However, there could be a best interconnect for, e.g., overall highest throughput or overall lowest latency.

Several works exist that address the design space exploration topic or some of its steps. The work in [7] presents a methodology to calculate possible mappings of the terminals of an SoC that achieve good performance and reduced power consumption, but it is limited to 2D-mesh topologies. In [6], a method to provide an optimal NoC for a system traffic pattern represented as a graph is presented. In [5] a whole tool to perform DSE of embedded systems is presented, while the work in [4] presents a process for performing design space exploration of VLIW architectures. Both works require to have some complex estimations of the system traffic pattern, like traces. Finally, in [1] a full design methodology for automatic generation of heterogeneous NoC-based systems is presented. This work is oriented to multi user systems in which the user behavior with a device is gathered to guide the design process of new generations of similar devices.

In this report we show Lantiq’s Design Space Exploration Tool (LDSET), a tool chain for performing early DSE and performance evaluation using the ESL performance evaluation framework SystemQ [9]. We use a heterogeneous Multi-Processor SoC (MPSoC) as a case study that features a complex NoC as central interconnect backbone. Many parameters of the NoC, such as clock frequency, buffer sizes, and topology will be explored. We emphasize the importance of realistic traffic flows and layout-awareness for our performance evaluations while retaining considerably high abstraction levels for our simulations. Finally, the CEF format is used both for input and output files. CEF is a format developed by the NaNoC partners with the aim of defining any sort of NoC based interconnects. It is based on the XML standard, and thus it is encoded in a machine-readable form that is easy to process and still human-readable. The use of the CEF format facilitates the cooperation between the NaNoC partners and tools, improving the productivity and facilitating the sharing of data and tools. For this reason, it was a main design constraint of LDSET the use of CEF whenever possible, even extending it when needed.

2 LDSET

Lantiq Design Space Exploration Tool (LDSET) is a tool chain to perform the exploration of user defined design spaces. LDSET was designed as two independent parts. First, it implements a versatile and powerful NoC simulator, able to model any topology, in order to simulate the candidates from the design space. Around this backbone, Perl was used to build the required algorithms to perform the DSE, providing to the designer a set of the best candidates of the given
design space. It is important to mention that due to the vast size that a design space may grow into, the tool trades-off precision for performance in many algorithms, providing good candidates instead of the best possible candidate. For example, the first version of the terminal placement algorithm was able to provide optimal candidates, but at the cost of iterating several hours for small spaces (up to 10 terminals), and up to several days for large spaces (close to 100 terminals), while the current version is able to work with one hundred terminal designs in less than 15 minutes.

2.1 NoC Simulator

This part consists of a set of modules developed in SystemQ [9], a SystemC modeling framework used by the NaNoC partners LTQ and IFX (Simula and UPV are sharing developments in systemQ). The design and development of this part was guided by four objectives:

- **Accuracy**: required in order to safely explore the design space, thus minimizing the risk of discarding valid candidates due to inaccurate performance estimations.
- **Speed**: as the number of candidates of the design space increases, the speed of the simulations becomes critical in order to explore the design space in an affordable time.
- **Versatility**: indispensable due to the wide range of possible candidates that may be contained in a single design space.
- **Reusability**: required in order to provide a tool that can be easily adapted to satisfy the needs and/or environments (e.g., other partners needs)

The modules were organized in a hierarchy, as depicted in Figure 1(a). The lower level represents basic building blocks (e.g., crossbar, routing modules). The middle level contains whole network components (e.g., switches, network interfaces) built with the lowest level blocks. The upper level is the most complex. It contains the models of the traffic generators and the top level module, which is able to instantiate a wide range of NoC configurations by using the lower level modules, thus, providing the degree of versatility required to represent the wide variety of candidates that may be contained inside the design space. In this context, the speed objective was achieved in two ways. First, each module was heavily optimized to reduce memory cost and processor usage. Additionally, the ESL employed in the implementation is capable of modifying the behavior of the modules, up to some degree, without forcing to recompile them. In this way, the modules are able to represent any possible configuration after a single compilation process. Even more important, the use of the parameter file paradigm defined in [9] enables the designer to evaluate a huge variety of system configurations without recompiling. In this way, the simulation time is mostly unaffected.
by the design space variety, even when working with a multitude of network architectures and/or topologies. Finally, the accuracy objective is achieved by modeling any component relevant to the system performance, in such a way that the simulation behavior is as close to the real system as possible. Moreover, the high degree of modularity of this approach favors the reusability of the tool, for example, by allowing designers to change some basic blocks in order to modify the network architecture without affecting the more complex modules. Additionally, we use advanced traffic generators that can be programmed to inject a wide range of synthetic traffic patterns as well as use traffic traces, thus alleviating the impact of design re-spins produced by inaccurate traffic estimations.

2.2 Design Space Exploration

Any DSE process is composed of 4 main stages: design space definition, initialization, exploration and output generation. Figure 1(b) shows the main parts of LDSET and the data flow that links them. As depicted, the above mentioned stages are present, although the frontiers between each part are blurred, as candidate generation and evaluation are partially mixed. Although a general view is depicted in this section, each stage will be analyzed in detail in its own subsection. The first stage consists of the definition of the design space exploration (its boundaries, targets, and characteristics) as well as the user commands to the tool. The outcome of this stage is the design space. In the initialization stage, the design space is prepared for the exploration process. This stage carries out part of the candidate generation. In particular, it generates the base candidates, that are like templates that define the immutable characteristics of candidates. The exploration stage contains the interaction between the NoC simulator and LDSET. It carries out simulations and analyzes results. Also, this stage instantiates detailed candidates starting from the templates defined in the previous stage. Finally, in the output generation stage, the user is provided with performance summaries, candidate lists and detailed descriptions for each candidate.

2.2.1 Design space definition

The first step is the definition of the design space constraints and parameters. It comprises three main classes: design definition, design constraints and performance objectives. In this context, LDSET builds the design space by expanding the design definition with the design constraints, and provides the best candidates that meet the performance objectives. Figure 2 shows an example of design space definition in CEF format. The design definition class includes all parameters that allow the designer to define the design space, like the number of terminals, the type of each terminal, the different topology types that should be considered (i.e. 2D mesh, rings, trees, . . . ) or the traffic flows that define the design communication pattern. An example of design definition in CEF format can be seen in the configuration tag of Figure 2. Inside this class, the definition of the traffic flows is the most complex and most important task as it defines the rules to guide the terminal mapping for different topologies in the design space. The traffic flows also impact the results of the DSE tool since the accuracy of the simulations is heavily dependent on the way traffic generators work.

In the literature three major traffic generation approaches exist. First, synthetic traffic patterns are used to send generic messages to predefined or random destinations at a given rate [8] but this is too general for a SoC design. Second, benchmarks and traces can be used but hardware architectures and application constraints must match the target design. Third, mathematical models could be used, however it is difficult to ensure that they are representative for given application constraints. Our approach to address this issue is to use fully programmable traffic generators, that allow the designer to specify the traffic requirements of each terminal as accurate as possible, for example, providing support to several traffic flows per terminal (each one with different parameters, like communication frequency or size), or by distinguishing between transaction types. The CEF format does not include the definition of the traffic flows in the same tag as the rest of the design definition, defining instead a special tag to specify traffic flows: the communication_flows tag. Figure 12 shows how to configure one of our traffic generators in CEF format. In this example
a single traffic generator is configured to manage a communication flow to a single target. The flow is composed by two sub-flows: generating write transactions for 40% of the simulation time, and read transactions for only 20% of the simulation time. For the rest of the time (20%) the communication flow is idle.

The design constraints class is composed of all the parameters that define the boundaries of the design space, e.g. frequency ranges, buffer sizes or number of virtual channels. Finally, the performance objectives class defines the evaluation rules that will distinguish between optimal and suboptimal candidates, like throughput or latency requirements. The evaluation rules are critical to the outcome of the DSE process. For this reason, LDSET implements a reduced but powerful language to define evaluation rules. It support the maximum and minimum operators, logic operators (AND, OR and NOT), and numerical comparisons. Figure 2 shows an example of design constraints and performance objectives in the CEF format, under the tags constraints and evaluation_rules, respectively.

Although the DSE process should run autonomously, we believe that the designer must have a certain degree of control over the DSE decisions. In our approach design constraints and performance objectives are clustered in groups that will be processed individually. In this way, the designer can sort the groups in order to determine which constraints are evaluated first and thereby defining a path through the design space. This is the role of the groups tag in Figure 2.

### 2.2.2 Design space initialization

This stage consists of the generation of topologies, as well as the terminal-to-core placement. This stage is critical to the performance and accuracy of the tool. Although it is possible to employ
<comm_flows>
  <usecase>
    <name>Default Mode</name>
    <id>0</id>
    <initiators>
      <initiator>
        <name>INITIATOR</name>
        <id>0</id>
        <targets>
          <target>
            <name>TARGET</name>
            <id>30</id>
            <flows>
              <flow>
                <type>WR</type>
                <load>0.40</load>
                <data_length>4</data_length>
              </flow>
              <flow>
                <type>RD</type>
                <load>0.20</load>
                <data_length>8</data_length>
              </flow>
            </flows>
          </target>
        </targets>
      </initiator>
    </initiators>
  </usecase>
</comm_flows>

Figure 3: Example of traffic flow description in CEF.

Figure 4: LDSET topology generation flow.

methods that will ensure that the best possible candidate for any given design space is chosen, these methods are too complex, requiring high amounts of memory and time to process medium sized spaces (20 or 30 terminals). For this reason, the current implementation employs an advanced heuristic in order to provide good placements at the same time that is able to cope with huge design spaces, being able to calculate several topologies for a 100 terminals design in a matter of minutes.

Figure 4 shows the data flow of the topology generation process. In the first step, each traffic flow is assigned a weight according to the traffic flow communication frequency and the amount of data of each transaction. In this step the differentiation between read and write transactions is
very important. Figure 5 shows the optimal placement of a flow in a unidirectional ring according to the transaction type. In case of a write transaction the distance from the initiator to the target is critical, while the distance from the target to the initiator is irrelevant. On the other hand, as read transactions are composed of a short request message from the initiator to the target and a longer response message from target to initiator, the distance from the target to the initiator is the one that becomes critical.

The second step calculates the possible topologies that may accommodate the terminals of the design according to the design space definition. Although the currently supported topologies are rings (both unidirectional and bidirectional), 2D-meshes and 3D-meshes, this limitation is due to the lack of topology modules in the tool, not in the NoC simulator. As LDSET was designed to be easily upgradeable, it is possible to easily add new topologies to the tool with a minimal impact on the rest of its features. Then, for each possible topology terminals are placed according to the weight of the flows, in which they are involved, with the aim to minimize the distance (in hops) of the flows with higher weights at the cost of flows with lower weights. In this step, the initial set of candidates is generated according to the design space definition. Each candidate from the initial set will be later expanded into a new subset of candidates by applying the design space constraints.

This step is critical to the DSE tool speed, as if irrelevant candidates are allowed in the initial set, the later stages of the DSE may waste a considerable amount of time in candidates that will finally be discarded later on. Figure 6 shows the time spent calculating valid placements for an optimal set of candidates topologies and for a suboptimal one, for an SoC composed of different number of terminals. The optimal set of topologies is composed of those topologies that will have good placements for the design traffic flows, while the suboptimal set was calculated by a brute force approach, and includes all possible topologies defined in the design space. As shown in the figure, the execution time is greatly reduced when irrelevant candidates are removed as soon as possible.

### 2.2.3 Design space exploration

Due to the high number of possible candidates that may form a design space, advanced techniques to reduce the number of candidates are required in order to keep the performance of the DSE tool within acceptable levels. A common approach in this case is the use of genetic algorithms [7]. A good example of genetic algorithm is the NSGA-II algorithm [2], that enables the exploration of huge sets of multi-parametric spaces at a relatively low cost, while keeping a good accuracy. Those algorithms are carried out in several iterations, in which the best candidates of the previous iterations evolve into a new generation of candidates during a pseudo-random mutation process. Although it is not guaranteed that the best candidate will be found, this method will provide good-enough results while significantly reducing the number of candidates evaluated, thus becoming a good trade-off solution between accuracy and speed.

The simulation engine is the core of the DSE, playing a vital role in the DSE tool accuracy and speed. Although the tool currently implements two heuristics, a brute force approach and a genetic algorithm one, it is designed as a framework for simulation management where different heuristics may be added. The brute force approach is suitable for small designs, and it explores all
Figure 6: Placement time for optimal and suboptimal sets of topologies for different system sizes.

candidates inside the design space, guaranteeing that the best candidate inside the design space is found. As the size of the design space grows, it becomes more and more costly to employ the brute force approach, hence, then the need of the genetic heuristic. This latter heuristic starts by choosing a pseudorandom set of candidates, which are simulated and evaluated. The best candidates are chosen as survivors, and form the first generation of candidates. At this point, the genetic algorithm will mutate the first generation into a new set of candidates that will be simulated and evaluated, forming the second generation with the best candidates among the first generation and the new candidates. This process is repeated until the whole design space defined is explored or until the maximum number of generations defined by the user is reached. It must be noted that this heuristic is a performance-accuracy trade-off, as it will not obtain the best candidate of the design space, but the best candidate among all the evaluated ones.

Additionally, the simulation engine is highly parallelized, being capable of calculating new simulation scenarios, while at the same time performing and evaluating simulations in different processes. Moreover, simulations are performed in parallel, exploiting the capabilities provided by modern clusters of computers.

The simulation engine flow is depicted in Figure 7. It starts by executing a variable number of subtasks, that are responsible of executing the simulations. These tasks are the source of the parallel execution of LDSET, as can be executed on different machines. They keep monitoring the pool looking for new simulations to execute. When a simulation is found, the task executes the simulation and notifies LDSET about the finalization of the simulation. After launching the subtasks, the simulations engine initializes the subspaces and generates candidates according to the selected heuristic. Then, each candidate is translated into a simulation scenario that is added to the pool. The engine waits until all the simulations required by the heuristic finalize. At this point, the simulation manager is called to evaluate the candidates according to the defined evaluation rules. Finally, the heuristic is called again to decide if the DSE has finished or new simulations are required.

2.2.4 Output generation

The last step is the generation of the DSE output. Although less critical to the DSE performance, this decision is important to the global design process of any SoC. Re-spins may happen during an
SoC design process. Although DSE tools may help to reduce the number of re-spins by guiding the designer to the best candidate, in case the candidate becomes unsuitable for the SoC requirements in the later stages of the system design process, the designer must restart the whole DSE process with a different design space definition. The negative impact of the restart can be alleviated by providing a set of best candidates, instead of a single one when possible. Thus, giving the designer the possibility to quickly change the candidate for the rest of the SoC design process.

In our approach the designer may choose to have a single or a set of best candidates by setting the performance objectives of the DSE. If those objectives are very tight, the DSE tool output will consist of a reduced set of candidates, even a single candidate in some cases. If those objectives are relaxed it will output a larger set of candidates, organized by network architectures and/or topologies, and classified according to their suitability for latency or throughput sensitive scenarios.

The resulting set of candidates can be written out in the CEF format, so that all topology and parameterization information chosen by LDSET as promising is available for further processing by other tools.

### 2.2.5 Example

As an example, we show the performance results obtained by our DSE tool for an SoC composed of 16 processors that communicate with their local memories using read and write transactions. Additionally, four shared memories exist that are occasionally accessed by all processors. The design space is composed of several NoC topologies that are capable to interconnect 36 terminals. The design allows several buffer sizes in the ports of both network nodes and Network Interfaces (NIs), together with several possible numbers of virtual channels per port (both in the nodes and in the NIs). We also allow the tool to attach up to four terminals per node.

Figure 8 depicts the DSE results grouped by topology. The figure shows the suitability for latency or throughput sensitive scenarios normalized to the best candidate for each metric. For the sake of simplicity we show here only a couple of candidates for each topology. For latency sensitive scenarios, we can classify the candidates in two groups, the ones that provide the best average latency and the ones that provide the lowest maximum latency. While the former candidates are best suited for overall latency, the latter are better for systems with a tight upper bound on communication latency. As illustrated in the figure, mesh configurations show a very similar average latency, while the $3\times4\times3$ mesh presents the lowest maximum latency, making this topology the best choice for latency sensitive scenarios. This topology also shows the best throughput of the whole design space. This topology requires 36 nodes, while the other options, like the $3\times3\times2$ mesh and the bidirectional ring with 18 nodes require half this number of nodes. This will be an important fact when the implementation cost and the power consumption of the design are
evaluated in the later stages of the DSE.

3 Place & Route

As mentioned, in order to provide accurate performance estimations, the NoC simulator should consider the physical characteristics of each topology. Our approach to address this topic was
to add high-level place & route capabilities to LDSET. To carry out this task, a tight collaboration between LTQ and UNIFE was necessary. The backbone of the collaboration was five month internship of an UNIFE student in LTQ. During this internship, UNIFE contributed with their experience developing layout-aware simulation frameworks for NoCs, while LTQ contributed with their expertise developing fast simulation frameworks in SystemQ and the computing power required for such a development.

One of the most relevant physical properties of a NoC from a performance viewpoint is the operating clock frequency, as a too low operating frequency will probably lead to a non-valid candidate, regardless of any other advantages it may present. In a NoC there are two main causes of NoC low frequency: switches and links. Usually, when the critical path of the NoC is in the switches, the straightforward solution is either to use an alternative topology with smaller (and thus faster) switches, or to use another switch architecture or technology library able to improve it. In both cases LDSET has a little room for improvement: the tool already implements support for alternative topologies/architectures, and the characteristics of the technology library is an input parameter of the tool. For this reason, we focused on the link frequency issue.

A network link may become the critical path if it is too long. In order to correctly evaluate the length of a link it is necessary to synthesize the system and perform its place & route. However, this task is quite time consuming, thus being ill-suited for a DSE tool. The solution we engineered consist of a high-level place & route algorithm, able to estimate the length of a link according to several parameters of the design space and of each candidate. In an heterogeneous system each switch and terminal may have a different area, shape or size. It is important to consider the impact of the geometry of each component once placed into the NoC, as links will be wired around them. Also, the buffering strategy employed has a direct impact over link length, as link length is less constrained when switches may implement buffers between themselves and the links. The topology of each candidate is required in order to perform the layout of the candidate’s network. Additionally, the algorithm requires to know the maximum length that a link operating at 1 GHz can reach, with and without buffers between links and switches. Then the target frequency of each candidate is required. Finally, we added the possibility of defining fixed terminals, whose position in the layout is fixed by the designer and cannot be modified by the tool. In this case, the algorithm will take care of them, thus adapting the rest of the layout to those placement constraints. Figure 9 shows the CEF code required to configure the placement of a single terminal.

Figure 10 has been obtained by reading in a link model derived from a 65nm technology. In Fig 10(a),(b) and (c) the unit length has been changed, and therefore the cores are separated 1 mm, 1.5 mm and 3 mm far apart, respectively. The target frequency is 1 GHz in all cases. As can be observed, in order to achieve the target speed, the tool inserts an increasing number of repeater stages as the interswitch spacing increases. With this data, the algorithm generates a projection of the topology in a 2D-plane, calculating an approximate value for the length of each link. Then, it calculates the number of repeater stages required at each link in order to achieve the target frequency.

In order to introduce this feature in LDSET, both parts of the tool were modified. First, the NoC simulator was upgraded, adding models of multi-cycle links, in order to represent the additional latency introduced in links when repeater stages are needed for breaking long timing paths. Such models are backannotated from real physical synthesis runs with mainstream industrial design toolflows. Second, the topology generation flow represented in Figure 4 was updated to the one shown in Figure 11(a). As depicted, the place & route step is optional, meaning that the user can select whenever it has to be executed or if it can be skipped.

The place & route step consists of two parts. First, the communication requirements of the design are considered in order to build a logical placement, similar to the one carried out in the original LDSET. Then in the second part, the geometry of the topology and all the terminals is considered, calculating the layout of the topology. The placement heuristic of each topology is different, but the underlying method is the same. It is modeled as 2D matrix fill algorithm. The model requires the number of nodes that would be allocated in the X and Y dimensions. Next, it forms a matrix of nodes that represents the layout. In this matrix a node is internally represented
Figure 10: Place and route at 1, 1.5 and 3 mm inter-switch spacing at a target speed of 1 GHz (IP Core size not drawn to scale).
as depicted in Figure 11(b). As can be seen, each node is composed of 9 cells. The center cell is reserved for the switch (label as N), while the cells that are side by side to the node are routing channels (label as R). Finally, the most external cells are reserved for terminals (label as T). While a single switch can be placed in the center cell, terminal cells can allocate up to 2 terminals. Notice that each node size is determined according to the terminals defined inside it, so the columns and rows of the matrix are as tall/ wide as the tallest/widest node they contain.

Once the matrix of nodes is built, connections between the nodes must be defined, thus building the network topology. The connections maybe short or long. Short connections are defined between adjacent nodes, while long connections are defined between non-adjacent nodes. In case that the nodes do not share the same row or column, a link with a turn is automatically added. Finally, NIs and channels between NIs and nodes are automatically added. There are some special limitations in the case that fixed terminals are present:

1. Fixed terminals must be placed with one side on the border of the floorplan
2. Fixed terminals must be inside the floorplan
3. It is not possible to attach more than 4 fixed terminals to the same node
4. It is not possible to attach more than two fixed terminals of the same side of the floorplan to the same node

4 Conclusions

In modern SoC designs, it is required to have a DSE tool in order to reduce the design costs down to acceptable levels. In this report we have shown how early design space exploration and performance evaluation can be used to find an optimal system architecture for a NoC-based SoC with reasonable effort. We introduced LDSET, our DSE tool that allows the designer to specify traffic and design constraints. Its output is a number of system architectures ordered by suitability for a given metric like overall maximum latency. Moreover, whenever LDSET needs to read/write data it employs the CEF format: a standard format used between the NaNoC partners used to facilitate the collaboration. Finally, we presented our approach to address the gap between high-level performance estimations and real system performance, consisting of a high-level place & route algorithm, able to predict the impact of the system layout and frequency on system performance.
5 Future work

The LDSET tool as presented provides a great step forward to solving the problem of network-on-chip design and application. However, there exist several directions, in which the tool can be further developed and excelled in the future. The future work on extending the capabilities of the LDSET tool may be composed of various aspects. One of the most major extensions has to be the inclusion of a support for multiple clock domains. At this moment the tool assumes the whole system to be synchronously running at one and the same clock frequency. In application domains like wireless (WLS) handheld devices, where power consumption is of a major concern, the globally asynchronous, locally synchronous (GALS) design paradigm is widely used. It means that different sections of the chip run at different clocks and while portions of the system may run synchronously, the system as a whole is asynchronous. For the moment, the requirements formulated in the local clock domains have to be translated by the engineer to requirements based on a common clock domain to be able to apply the LDSET tool to WLS applications.

Additionally, the inclusion of different use-cases can also prove to be useful. A system may be used in different circumstances, some of which may be orthogonal to each other. This means that they may not occur at the same time. Especially with multi-function mobile devices, a lot of different use-cases may exist with significant communication requirements, but never at the same time (e.g. video capture and video playback on a smartphone). Therefore, these use-cases exist separately by themselves and their flows may not affect each other. On such occasions, the tool will have to optimize the architecture in order to make it compliant to all the different requirements. At the end the architecture has to be able to sustain all the traffic as generated by the separate usages.

Finally, the inclusion of more different topologies besides the current supported ones will render the tool even more flexible and applicable to real life industry problems. The code provides already the necessary modularity to add support for more topologies, so it is just a matter of further design effort to increase this list. At least some more topologies that close the gap in multi-hop latency and number of connections between rings and meshes would be very interesting for power-sensitive mobile applications, where the power consumption of a mesh may be too high, but the latency and performance of a ring may be too low. Ideally, LDSET would be able to support irregular structures, where the topology is built to be a custom fit for the system in question, but this probably is too great an explosion of explorable design space unless very powerful pruning heuristics can be found for that problem.

References


<comm_flows>
  <usecase>
    <name>Default Mode</name>
    <id>0</id>
    <initiators>
      <initiator>
        <name>P0</name>
        <id>0</id>
      </initiator>
    </initiators>
    <targets>
      <target>
        <name>LM0</name>
        <id>30</id>
        <flows>
          <flow>
            <type>WR</type>
            <load>0.40</load>
            <data_length>4</data_length>
          </flow>
          <flow>
            <type>RD</type>
            <load>0.20</load>
            <data_length>8</data_length>
          </flow>
        </flows>
      </target>
    </targets>
  </usecase>
</comm_flows>

Figure 12: Example of input file flow description

6 Appendix: Differences with CEF format

As exposed, LDSET was developed in parallel with the CEF format. For this reason, several parts of the CEF format were undefined when LDSET was developed, while others were modified during the development process of CEF. This section is a list of the main differences between the current CEF format and the one supported by LDSET.

In the CEF schema every component in the NoC is a block: nodes, terminals, repeaters and network interfaces (NIs). Each block is defined by a unique numerical id, which is used to identify the block when needed. In the input file, the only accepted blocks are terminals. The rest of the NoC components and their relationships will be an outcome of the DSE. So, the input file is a subset of the CEF schema, that allows to define terminals, the flows that relate the terminals, and the design space constraints. Some of those constraints, like the floorplan dimensions, were already in the CEF schema, while some others were added to expand the capabilities of CEF. Notice that some of the parameters defined in the CEF file are currently not used in LDSET input files. Some of those are output parameters for the tool (i.e., core placement coordinates, routes across the network), while others are just out of the scope of LDSET. Anyway, LDSET will recognize valid CEF parameters and silently ignore the ones that are not required as input parameters.

The first difference is the flow definition. As the current implementation of LDSET does not support use cases, only the first usecase block will be considered. Also, due to the kind of traffic generators employed in the current SystemQ modules, the whole flow block is replaced with the
one showed in Figure 12. As can be seen, there are three tags in a flow block: the type of the flow (RD means read, WR means write), the load (a double between 0 and 1) and the data length of the transaction (in words). Note that the sum of all the loads of a single initiator cannot be higher than 1.

Other set of differences between CEF and the input file are found in the definition of terminal blocks. First, the bit width of each terminal must be defined inside a `bitwidth` block. Second, the allowed types are only 100 for masters and 101 for terminals. Third, the `fixed` block should be added inside the `bottomleftcorner_position` tags, this tag is a boolean and when set to true the geometrical placement algorithm will not move this terminal.

Another difference between the original CEF format and the tool input format is the `design_space_exploration` block. This block contains the definition of the design space boundaries and the targets of the design space exploration. Figure 2 shows an example of this block.

```
<distributedrouting>
  <tables>
    <table>
      <switch_id>70</switch_id>
      <destination_id>0</destination_id>
      <input_port>0</input_port>
      <output_ports>
        <output_port>
          <id>0</id>
          <route_weight>0</route_weight>
        </output_port>
        ...
      </output_ports>
    </table>
    ...
  </tables>
</distributedrouting>
```

```
<distributedrouting>
  <switch>
    <name>Node_0</name>
    <id>70</id>
  </switch>
  <routingproperties>
    <defaultport>0</defaultport>
    <routingtable>
      <line>
        <id>1</id>
        <port>4</port>
      </line>
      ...
    </routingtable>
  </routingproperties>
</distributedrouting>
```

Figure 13: Routing description in CEF (left) and LDSET (right)

Finally, among all the routing schemas supported in CEF, current LDSET implementation supports only the distributed routing one. Also, the routing table description is completely different from the one described in the CEF format. Figure 13 shows the LDSET routing table format and the new CEF format.