Optimizing Heterogeneous NoC Design

Yaniv Ben-Itzhak[1] Israel Cidon[2] Avinoam Kolodny[2]
Electrical Engineering Department
Technion – Israel Institute of Technology
Haifa, Israel
[1]yanivbi@tx.technion.ac.il [2]{cidon, kolodny}@ee.technion.ac.il

ABSTRACT
We develop a novel design methodology that optimizes capacity of each link in a NoC and the numbers of virtual channels (VCs) at each router port for a given set of flows and latency constraints. In order to lower computation costs associated with a simulated annealing search in the design space, we utilize an approximate analysis of the NoC performance thus replacing the need for a NoC simulation. Therefore, computation time and resources are dramatically reduced. The area saving achieved by our heterogeneous NoC design is demonstrated by several use-cases. The heterogeneous NoC design process is applied to SoCs running multimedia benchmarks, and to Chip-Multi-Processor (CMP) running PARSEC benchmark programs.

Categories and Subject Descriptors
B.8.2 [**Performance and Reliability**]: Performance Analysis and Design Aids Language

General Terms
Algorithms, Performance, Design

Keywords
NoC design, Heterogeneous NoC

1. INTRODUCTION
SoC and CMP designs use Networks-on-Chip (NoC) to support a variety of inter module communications bandwidth and latency requirements. The NoC traffic requirements are usually heterogeneous, both in terms of module-to-module bandwidth and delays. In the case of SoC, there is typically a priory specification describing the traffic and timing requirements at design time. In CMPs, the traffic requirements depend on particular executed software. However, certain parts of the NoC, depending on network topology and chip layout, always tend to be heavily loaded. Figure 1 presents two well-known examples which demonstrate that heterogeneous traffic loads are common: uniform traffic pattern over a mesh NoC [10, 22] (Figure 1(a)) and CMP with a tiled cache in the middle architecture (which employs banked DNUCA connected by a NoC [3]) (Figure 1(b)). These two examples exhibit higher loads over NoC routers in the center as compared to NoC routers in the periphery.

The problem of heterogeneous traffic loads is usually solved by a load distribution method, such as dynamic routing [19, 21, 20]. However, these solutions require additional logic, which consumes area and power. Therefore, fixed shortest-path routing is typically preferred in order to minimize NoC area and communication power dissipation [8, 23, 9, 25, 18]. With simple routing schemes, such as dimension-order routing, different links in the NoC must carry significantly different data rates and number of flows. Hence, heterogeneous NoCs have been proposed [13, 15, 12, 22, 2, 17].

As the traffic requirements are heterogeneous, one should also expect that the optimal area/power NoC to support these requirements will also be heterogeneous in terms of link capacities and a number of virtual channels (VCs) for each unidirectional port. The allocation of capacity and the number of VCs should be affected by different latency requirements for different data flows while considering the sharing of network resources by different flows. Furthermore, the use of such heterogeneous NoC resources results in fewer restrictions on mapping optimizations as compared with homogeneous NoCs, where capacities and virtual channels are uniform for all links and routers. Hence, our approach facilitates more efficient design.

This paper explores optimal link capacity and VCs allocation for heterogeneous NoCs under end-to-end latency constraints. Some of previous works focused on either non-uniform number of VCs allocation (with uniform link capacities) [13, 15] or non-uniform link capacities allocation (with uniform number of VCs) [12]. Others focused on NoCs composed of a small set of predefined routers with fixed bandwidth and a number of VCs for all ports [22, 2, 17]. We argue that efficient NoC design should consist of heterogeneous routers, such that each unidirectional port of each router is automatically generated with its own required capacity and number of VCs. Unlike [13, 15] which are based on heuristics, we use optimization method in order to obtain the optimal NoC design.

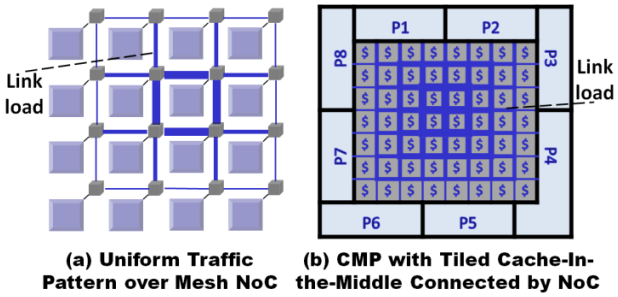

Figure 1. Heterogeneous traffic loads examples (The link thickness corresponds to its load). (a) Uniform traffic pattern; (b) CMP with tiled cache in the middle. Both result in higher loads at center as compared to periphery of the NoC.

SLIP'12, June 3, 2012, San Francisco, CA, USA.

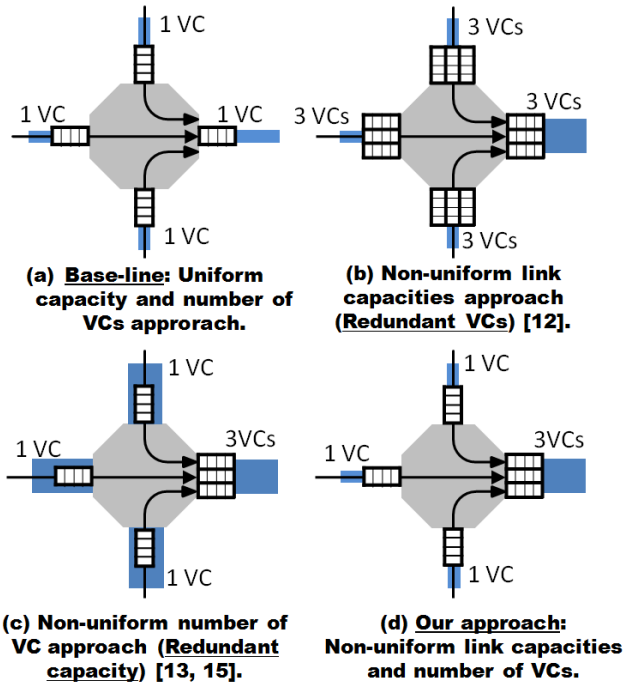

Figure 2. Reasoning for using our heterogeneous NoC design (The link thickness corresponds to its capacity). (a) Uniform approach (Homogeneous NoC); (b) Non-uniform capacity approach; (c) Non-uniform VCs approach; (d) Our approach.

Figure 2(a) presents a simple design example in which three equal rate flows are transmitted over a shared link. This design uses uniform capacity and a uniform number of VCs for all links. Previous works have used two resource allocation strategies to achieve better performance. First, the non-uniform link capacities approach [12] is presented in Figure 2(b). This approach uses three VCs in every link, even though it is required only for the shared link. Second, the non-uniform number of VCs approach [13, 15] is presented in Figure 2(c). This approach uses high capacity for all links, even though it is required only for the shared link. Therefore, both of these existing approaches allocate excess resources which increase the area and the power consumption of the NoC. Our novel heterogeneous NoC design methodology allocates non-uniform links capacities as well as a non-uniform number of VCs (Figure 2(d)). Therefore, unlike previous approaches, the allocation of unnecessary VCs and capacity over ports is avoided in the new approach thus saving a considerable amount of area and power.

We substantiate our approach by presenting the correlation between link capacities and number of VCs in optimized designs (Section 4.1). Hence, one cannot attain optimal NoC design by using previous methods separately (e.g., a non-uniform link-capacities design followed by a non-uniform number of VCs design).

We use a heterogeneous NoC router architecture, which is an extension of the XShare NoC router architecture presented in [11]. We extended this architecture such that it supports different link capacities and different number of VCs for each unidirectional port. To the best of our knowledge, this is the first description of such heterogeneous NoC routers. Section 2.1 provides the detailed architecture of our heterogeneous NoC routes. Since ORION2 [14] supports only homogeneous NoC routers, we modify it in order to evaluate the area of a given heterogeneous NoC router (Section 2.2). We also point out the benefits of heterogeneous NoC routers (Section 2.3).

We present an allocation scheme which minimizes the total area of VCs and the capacity of links required to meet the end-to-end latency constraints of each flow. Our allocation scheme is based on simulated annealing, customized to support the use of constraints and the optimization of two parameters (i.e. link capacities and number of VCs). Furthermore, we use the network latencies as constraints rather than using the end-to-end latency constraints. Consider the case of an allocation with saturated sources such that the corresponding end-to-end latencies are equal to infinite. Whenever the optimizer compares two saturated NoC allocations, it cannot get useful indication as to which allocation is better. Therefore, in order to improve the optimization progress, we calculate the network latency constraints based on the given end-to-end latency constraints. In this way, the optimizer can also get a useful indication even for saturated NoC allocations. More details about the design process can be found in section 3.2.

In order to reduce the computational time and resources, we employ an approximate analysis of the NoC delay [4] in the inner optimization loop. Usually, the design process heavily relies on extensive performance simulations, where each intermediate NoC configuration is tested in terms of meeting the requirements in a long 'change and test' search sequence. The use of detailed simulations makes the task of searching for efficient link capacities and virtual channels allocation computationally intensive and does not scale well with the size of the problem. Therefore, the use of costly simulations [5] is left only for the final verification and fine-tuning of the system.

In section 4, we present several heterogeneous NoC designs and demonstrate the area savings achieved by it. We present a synthetic heterogeneous NoC example in section 4.1. This example demonstrates the correlation between link capacities and number of VCs and justifies the optimization of both in order to achieve an optimal design. It also demonstrates that there is a possible trade-off between the total capacity of links and the total number of VCs, which actually offers more than one optimal heterogeneous NoC design. We apply the heterogeneous NoC design methodology to several examples of SoCs running multimedia benchmarks in section 4.2 and demonstrate the area savings of heterogeneous NoC design compared to homogeneous design. Furthermore, we demonstrate that heterogeneous NoC design is also justified for Chip-Multi-Processor (CMP) in section 4.3, by presenting an optimal design for CMP running PARSEC benchmark programs.

2. HETEROGENEOUS NOC ROUTER

Heterogeneous NoC consists of routers which support different link capacities and different numbers of VCs for each unidirectional link. To the best of our knowledge, there is neither a detailed implementation of heterogeneous NoC architecture nor a method for calculating the area of such routers. Therefore, we define a new architecture for heterogeneous NoC routers which is an extension of the architecture of XShare NoC router [11] (Section 2.1). Furthermore, we present the area evaluation of such NoC router which is a modification of ORION2 [14] (Section 2.2). We also point out the benefits of heterogeneous NoCs (Section 2.3).

2.1 Architecture of Heterogeneous NoC Router

Our heterogeneous NoC router is an extension of XShare architecture [11]. XShare is a homogenous NoC router architecture which allows two flits to be concurrently transferred over a single channel. Therefore, the size of each flit is at most half of the link width. The two flits are combined together and sent over the link as a single bigger flit. The input buffer is modified in order to divide the incoming combined flit into the two original flits. This modification is accomplished by adding MUX/DEMUX logic to the input buffer. Next, the in-port arbiter chooses which of its VCs can bid for the out-port, according to the arbitration policy of the NoC router (e.g., round-robin or winner-takes-all). Each out-port requires additional logic in order to support concurrent transfer of two flits. Each one of the out-port arbiters chooses one of the in-ports to be served. Afterwards, the out-port combines the two chosen flits into a bigger flit and sends it over the outgress link and so on and so forth.

As opposed to XShare, our NoC router architecture is heterogeneous. Therefore, we extend the XShare architecture to support non-uniform link capacities and non-uniform number of VCs per unidirectional port. Figure 3 depicts the architecture of our heterogeneous NoC router. In order to support different number of VCs for each unidirectional port, we use V_i:1 in-port arbiter for each in-port i (with V_i VCs). All VCs have the same number of buffers. In order to support different link capacities, we use different link-widths while keeping the NoC router`s frequency fixed. The link-width of link o, $LinkWidth_o$, is determined by the number of flits which can be concurrently transmitted over the link, n_0.

$$LinkWidth_o = n_o \cdot FlitSize \qquad (1)$$

Therefore, we use n_o simultaneous out-port arbiters for each outgress link o. The concurrent transmitted flits can be either of different ingress VCs or of the same VC. It depends on the number of VCs over the outgress link o and on the current traffic pattern towards the outgress link. The capacity of link o is:

$$Capacity_0 = LinkWidth_o \cdot Frequency \qquad (2)$$

According to equations (1) and (2), the capacity granularity of the allocation process is determined by the product of the NoC`s frequency and the flit size. In order to support the connectivity of ingress and outgress ports, the crossbar consists of different ingress and outgress link widths. It allows combining flits from several ingress ports with low capacity into an outgress port with high capacity. Inversely, it also allows concurrent transfer of flits from an ingress port with high capacity to several outgress ports with low capacity.

2.2 Area Modeling

In this section, we discuss our area estimation for the above-described heterogeneous NoC router architecture. Previous works on power and area modeling for NoC routers mainly focused on homogeneous NoC routers. In particular, ORION2 [14] only supports homogeneous NoC router. Therefore, we modify ORION2 to support the area evaluation of the heterogeneous NoC architecture presented above. Our starting point is the equations of ORION2. Next, we modify them to support different in-port and out-port arbiter sizes, and heterogeneous crossbar. Furthermore, we add area estimation for the required additional MUX/DEMUX logic of the input and the output ports for supporting simultaneous transmission of flits over a link [11].

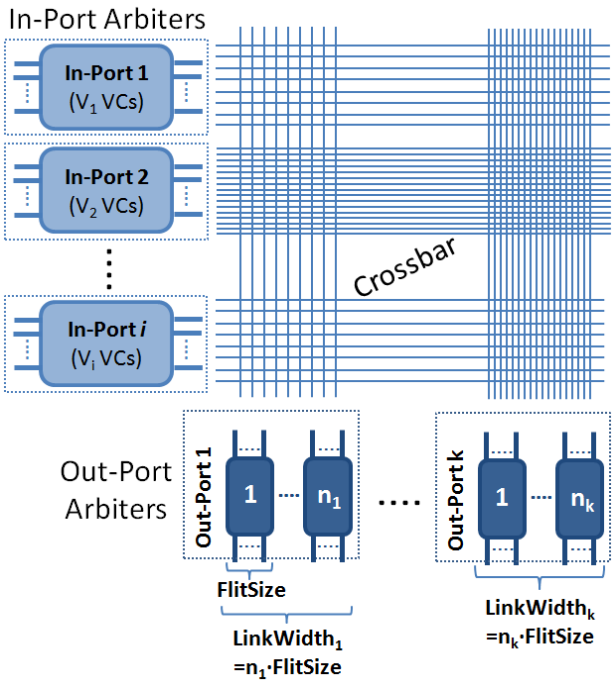

Figure 3. Our heterogeneous NoC architecture.

Our work targets total NoC area minimization for given source-destination flows. In general, NoC power consumption is proportional to the NoC routers` area and load. Since, the load of the NoC routers depends on the given flow pattern, the power consumption in our work is mostly correlated to the NoC routers` area. Hence, the minimization of NoC area also leads to reduction of the NoC power consumption.

2.3 Benefits of Heterogeneous NoC Router

Heterogeneous NoC routers require additional MUX/DEMUX logic and simultaneous out-port arbiters in order to support a variable number of VCs and variable link capacities, respectively (Section 2.1). Our area evaluations show that these additional logics are almost negligible. The area difference between a homogeneous NoC router and the most possibly heterogeneous NoC router is only 3.5% given an equal sum of link capacity and a total number of VCs. Hence, by using heterogeneous NoC design, a given area budget of a NoC router can be utilized in a much better way, which results in better performance per area compared to homogeneous NoCs. Moreover, the use of heterogeneous routers can reduce the total NoC area without violating any performance constraints. This is accomplished by avoiding the allocation of unnecessary VCs and capacity over certain ports which would be allocated in a homogeneous design because of performance restriction of other ports. In section 4, we demonstrate the area savings which can be achieved by our novel heterogeneous NoC design.

3. DESIGN PROCESS

Our goal is to minimize the NoC area by an optimal allocation of link capacities and a number of VCs for each unidirectional port

of each router under end-to-end latency constraints of each flow. In this work we focus on minimizing the total NoC area, which in turn also reduces the NoC power consumption.

The NoC consists of heterogeneous NoC routers as described in section 2.1. The total NoC area (both links and routers) is calculated by taking into account the capacity and the number of VCs of each unidirectional link (Section 2.2).

3.1 Optimization Problem Definition

For given source-destination flows, each flow *f* with its packet generation rate, $\lambda(f)$, and its end-to-end latency constraint, $t_{e2e}^{constraint}(f)$. We minimize the total NoC area by allocating capacity and number of VCs for each unidirectional port. The parameters and variables we use are listed in TABLE I.

TABLE I. PARAMETERS AND VARIABLES DEFINITIONS

Notation	Definition
$t_{e2e}(f)$	The end-to-end latency of flow *f*.
$t_{e2e}^{constraint}(f)$	The end-to-end latency constraint of flow *f*. (The time from the generation of the packet until its arrival to the destination).
$t_{network}^{constraint}(f)$	The network latency constraint of flow *f*. (The time from the injection of the packet into the NoC until its arrival to the destination).
$\lambda(f)$	The packet generation rate of flow *f*.

The definition of the optimization problem is:

$$Minimize: Total\ NoC\ Area$$
$$Subject\ to: t_{e2e}(f) \le t_{e2e}^{constraint}(f); \forall f \in flows \quad (3)$$

3.2 The Optimization Method

The design space for allocating link capacities and number of VCs for each unidirectional port is extremely large. Therefore, we use the simulated annealing algorithm [16], which targets large search space and copes with local minima. Moreover, simulated annealing algorithm is also suitable for extreme large problem instances (e.g. 1000+ tiles). Generally, simulated annealing algorithm minimizes a general function without constraints. Therefore, we modify the basic algorithm to support the use of constraints by updating the current allocation with a new lower area allocation only if it meets the end-to-end latency constraints of all flows. Furthermore, we minimize the total NoC area by allocating both capacity and the number of VCs. Therefore, at each iteration we use a candidate allocation generator function. This function randomly changes either the capacity or the number of VCs of an arbitrary port as long as this change yields a legal allocation (i.e. non-negative capacity and positive number of VCs for all links). As mentioned above, the capacity is changed according to its granularity which depends on the product of the NoC`s frequency and the flit size. The number of VCs is changed by one at each iteration.

There are several techniques to support faster convergence for large scale NoCs (e.g. 1000+ tiles). One can use progressive capacity and number of VCs granularities, such that the granularities are relatively high (i.e., high change of capacity/number of VCs) at the first iterations of the optimization and get lower as the optimization is progressing. Moreover, several random unidirectional ports can be changed at each iteration. The number of changed ports per iteration is getting lower as the optimization progress. By using such techniques, the optimization progress would be faster and support optimal NoC design even for extreme large problem instances.

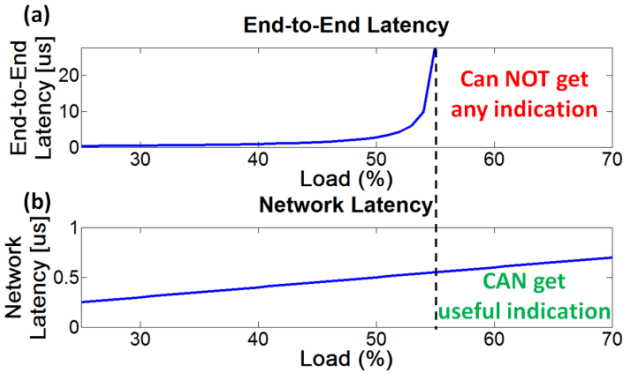

Figure 4. Rationale for using network latencies constraints instead of end-to-end latencies constraints.

3.3 End-to-End Latency Constraints Support

The optimization process dramatically slows down when the network approaches saturation. Consider the case shown in Figure 4(a), in which the current resource allocation causes saturation. The optimization algorithm compares the infinite end-to-end latency of the current NoC allocation to the end-to-end latency of a new saturated NoC allocation (by modifying either the capacity or the number of VCs of an arbitrary link). Since the network is saturated, the optimization most times does not get a useful indication as to which allocation is better. However, in contrast to the end-to-end latency, the network latency is always finite. In particular, network latencies are meaningful even when the network is saturated (Figure 4(b)). Hence, the using of network latency constraints (instead of end-to-end latency constraints) provides meaningful information to the optimization process even at saturated network cases.

To that end, for each flow *f*, we calculate the required network latencies constraints based on the given end-to-end latency constraints. We decompose the end-to-end latency constraint into the source queuing time and the corresponding network latency constraint by solving the following equation:

$$\frac{\lambda(f)\cdot t_{network}^{constraint}(f)^2}{2\left(1-\lambda(f)\cdot t_{network}^{constraint}(f)\right)} + t_{network}^{constraint}(f) = t_{e2e}^{constraint}(f) \quad (4)$$

We approximate the source queuing time by using the queuing latency of M/D/1 queue model (the first expression of equation (4)) [12]. The service time of the source queue is equal to the corresponding network latency constraint, $t_{network}^{constraint}(f)$.

4. RESULTS

In this section, we present several heterogeneous NoC designs. In particular, we demonstrate the area savings that can be achieved by our novel heterogeneous NoC design without violating any performance constraints. We assume 45nm process technology and evaluate the total NoC area (both links and routers) of a given NoC design according to the description in section 2.2.

We execute the modified simulated annealing algorithm in two phases. In the first phase of the optimization process, we use the delay analysis methodology for the heterogeneous NoCs presented in [4] in order to evaluate the flows` latencies of an arbitrary allocation. The usage of the analytical delay model dramatically saves run-time and resources during the design process. Next, we use the optimized allocation achieved in the first phase as an initial allocation for the second phase. In this phase, we use HNOCS (Heterogeneous NoC Simulator) in order to validate and refine the optimized allocation. HNOCS is an OMNeT++ based heterogeneous NoC simulator [24, 1, 5]. It supports any heterogeneous NoC configuration in terms of any link capacity and any number of VCs. HNOCS executes wormhole switching with VCs and deterministic XY routing.

The following heterogeneous NoCs are designed with capacity granularity of one GBps, defined by NoC frequency of 250 MHz and flit size of four bytes (Section 2.1). One can round the link capacities of the following optimized heterogeneous NoC designs in order to support design which uses higher capacity changes.

4.1 Synthetic Example

We present a synthetic heterogeneous NoC design example, which demonstrates the area savings achieved by heterogeneous NoC design. Furthermore, we demonstrate that heterogeneous NoC design can offer more than one optimal design, and also the correlation between the links` capacities and number of VCs.

Figure 5 presents a 2x4 NoC with two flows. The packet generation rates of the flows are six and eight GBps. We execute our heterogeneous NoC design process with end-to-end latency constraint of 0.4 µsec for each flow.

Figure 5(a) presents the minimal homogeneous NoC, which meets the flows` end-to-end latency constraints, and its area (Considering only active links. The full homogenous NoC area is much higher: 83776µm^2). The capacity and the number of VCs (19 GBps and two VCs) of all links are determined by the link from core 1 to core 2, since it is common for both flows.

The heterogeneous NoC design has two different options, which result in almost the same area. In both options, the area is approximately 25% less than the area used by the homogeneous NoC design (active links only). On one hand, while the first option (Figure 5(b)) allocates a single VC, the second option (Figure 5(c)) allocates two VCs for the link between core 1 and 2. On the other hand, the total capacity allocated by the first option is higher compared to the second option. In fact, these two options demonstrate a trade-off and correlation between total number of VCs and capacity (79GBps and 5VCs versus 77GBps and 6VCs).

The first option (Figure 5(b)) incurs longer path-acquisition latency over the link between core 1 and 2, since it allocates a single VC over it. Therefore, in order to meet the end-to-end latency constraints, the links should have higher link capacities compared to the second option (Figure 5(c)). This is necessary to decrease the transfer latency of the flows and compensate for the path-acquisition latency. The second option, on the contrary, results in instantaneous path-acquisition latency over the link between core 1 and 2 (the link has two VCs). Therefore, it allows the use of links with lower capacities compared to the first option while maintaining the same end-to-end latency constraints.

4.2 Heterogeneous NoC-Based SoC Design

In this section, we present a heterogeneous NoC design for SoC. We use the SoC applications presented in [6]. We execute our heterogeneous NoC design process with end-to-end latency constraint of 0.4 µsec for each flow. We manually map the SoC application into the NoC, and the results are averaged for several mappings. Figure 6 presents a comparison between the total NoC area consumed by our heterogeneous NoC design and the total NoC area consumed by a homogeneous NoC design.

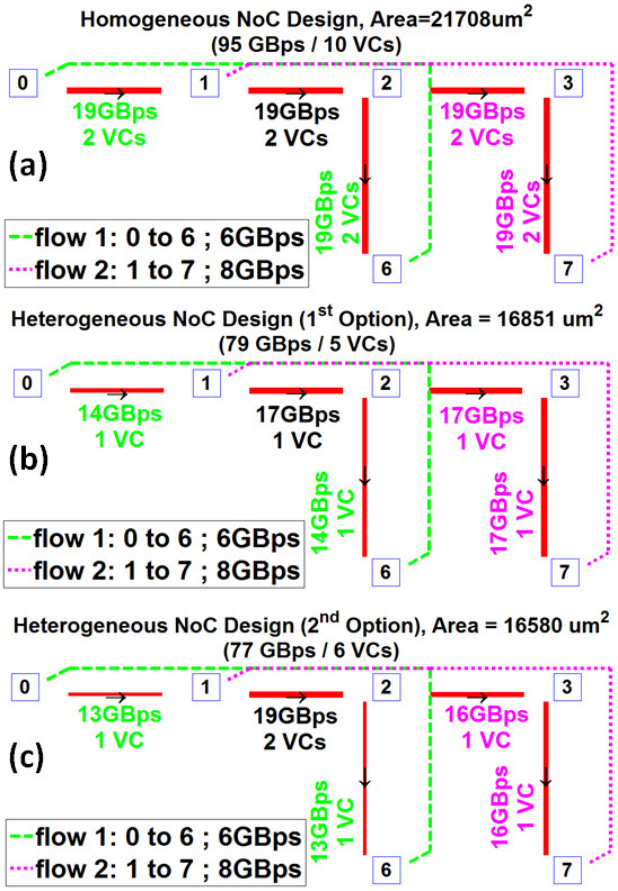

Figure 5. Different options for NoC design. (a) Homogeneous NoC design; (b) Heterogeneous NoC design (first option); (c) Heterogeneous NoC design (second option). Both heterogeneous NoC designs offer area savings of 25%.

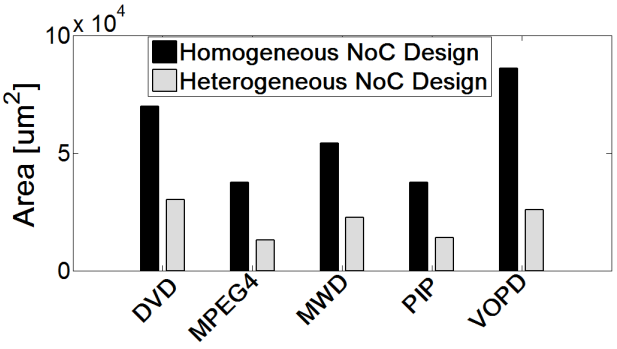

Figure 6. Area comparison of NoC-based SoC designs for the SoC applications presented in [6]. The heterogeneous NoC design offers area savings 55% to 70%.

The heterogeneous NoC design results in area savings of 55% to 70% compared to homogeneous NoC design. In homogeneous NoC design, the capacity and the number of VCs of each unidirectional port are determined by the link with the tighter constraint. Therefore, unnecessary capacities and VCs resources are allocated in homogeneous NoC design. However, in heterogeneous NoC design, the capacity and number of VCs for each unidirectional port is allocated according to its own constraint. Hence, unnecessary resource allocation is avoided, which allows major area (and power consumption) savings.

4.3 Heterogeneous NoC-Based CMP Design

In this section, we demonstrate that heterogeneous NoC design is also justified for CMPs. We present a heterogeneous NoC design for tiled CMP running PARSEC benchmarks [7]. To that end, we use HNOCS in the inner loop of the optimization. HNOCS is extended to provide functionality of core with L1 cache, L2 shared cache and DRAM controller. Figure 7 presents a 4x4 NoC-based CMP which consists of these modules. We assume 64KB L1 cache and 4MB L2 shared caches, both with 64 bytes cache lines and associativity of 64 lines per set. We apply L2 cache access traces of the PARSEC benchmarks to this model.

Since the benchmarks consist of several different phases, we use the benchmarks` run-time as constraints for our optimization rather than using the end-to-end latencies of each memory transaction. The run-time constraint for each benchmark is determined by the run-time over a base-line homogeneous NoC, which consists of links with capacity of 22 GBps and two VCs.

First, we design different optimized heterogeneous NoCs for different random thread allocation cases of PARSEC benchmarks. To that end, we modified the simulated annealing algorithm. For each candidate allocation, the optimization evaluates the run-time of each benchmark. Next, based on these optimized heterogeneous NoCs, we design a general optimized heterogeneous NoC which supports the execution of any thread allocation of any benchmark over the CMP.

Figure 7 presents this optimized heterogeneous NoC. It can be seen that there are mainly three different types of links. The run-time is mostly determined by the time to handle a miss read. Thus, the links with highest capacity are the links connecting the DRAM controllers and the L2 caches. These links (marked in dotted blue) have 22 GBps and two VCs. In order to allow fast block replacements during miss handles in the L2 caches, the links which connect the L2 caches to the DRAM controllers (dashed green) have 12 GBps and two VCs. All other links (solid red) which connect the cores to the L2 caches, are actually handle much lower load. Thus, they have three GBps and a single VC. Our optimized heterogeneous NoC area consumes only 20% of the homogeneous base-line NoC area.

4.4 Design Progress and Scalability

In this section, we discuss about the design progress by presenting the NoC area versus the number of iterations of the heterogeneous NoC design process. We describe the generation of a modified initial homogeneous NoC allocation to the optimization method, which decreases the required number of iterations of the design. Next, we demonstrate how to achieve scalability by using the progressive techniques presented in section 3.2.

In order to reduce the required iterations of our design method, we use a modified homogeneous NoC as our initial allocation for the optimization method, such that number of VCs of each unidirectional port is equal to the maximum number of flows that

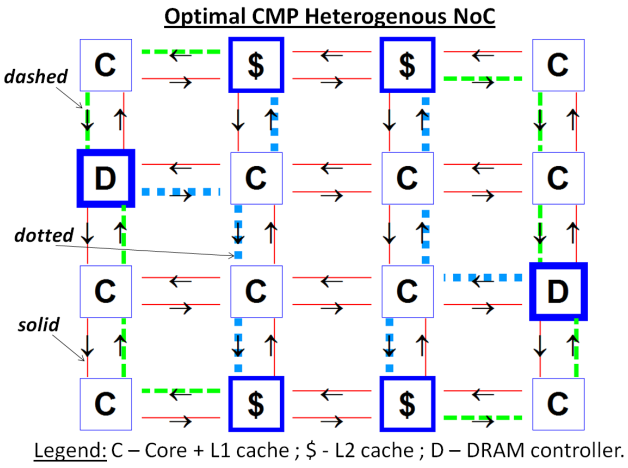

Figure 7. The optimized heterogeneous NoC for CMP executing any thread allocation of any PARSEC benchmarks [7]. Dotted (Blue) links connects DRAM controllers to L2 caches and have capacity of 22 GBps and two VCs. Dashed (Green) links connects L2 caches to DRAM controllers and have capacity of 12 GBps and two VCs. All other links (Solid red) have three GBps and single VC.

transmitted over it. For instance, there is only a single flow over the ports that connect cores 0 to 1, 2 to 6 and 3 to 7, in the synthetic NoC example presented in section 4.1 (Figure 5). Therefore, the initial NoC of our optimization method is similar to the homogeneous NoC (Figure 5(a)) except that the aforementioned ports have a single VC rather than two VCs.

Figure 8 presents the design progress versus number of iterations for the synthetic example (Section 4.1). It can be seen that the NoC area is exponentially reduced with the number of iterations. We achieve area reduction of 12% by using the modified initial homogenous NoC allocation. After only 30 iterations we get area reduction of 18% and after 90 iterations we get final area reduction of 25%. Hence, we achieve 72% of the total area savings (area reduction of 18% out of 25%), after only 33% of the required iterations.

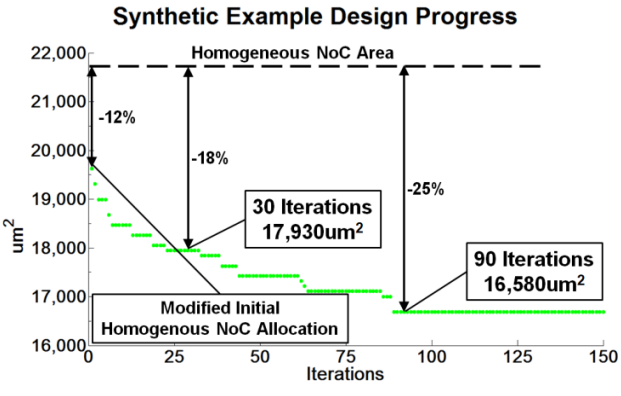

Figure 8. The design progress of the synthetic example design which presented in section 4.1 by NoC area versus number of iterations. Area reductions of 12% by using the modified initial homogenous NoC allocation, 18% after 30 iterations, and 25% after 90 iterations.

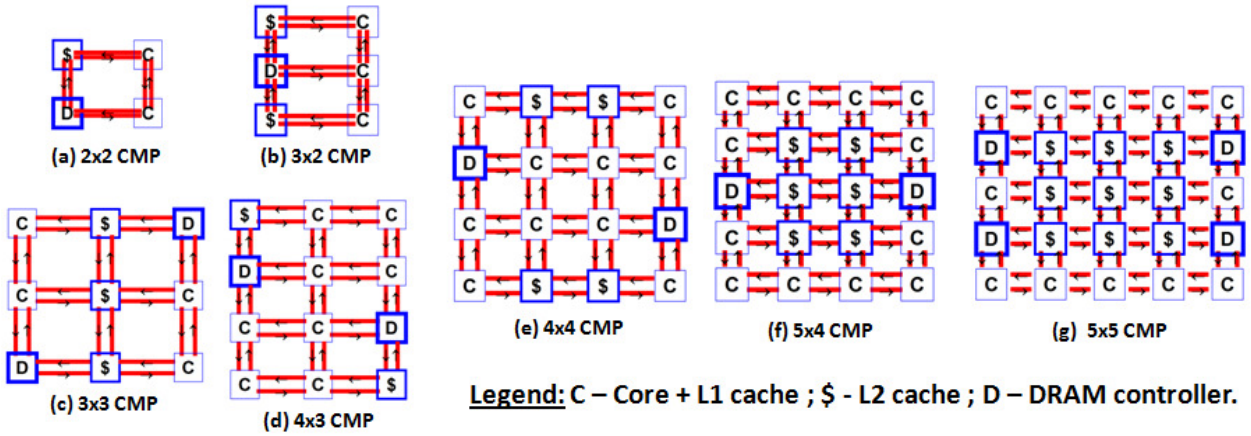

Figure 9. The CMP NoC designs presented in Figure 10.

We demonstrate the scalability of our NoC design by using the two progressive techniques presented in section 3.2: *progressive capacity granularity* (start with high capacity steps, which get lower with the optimization progress) and *progressive number of links per iteration* (start to change a high number of links per iteration, which gets lower with the optimization progress). We apply the CMP NoC design method presented in section 4.3, to the CMP topologies presented in Figure 9. The CMPs consists of four up to 25 tiles. Figure 10 presents the number of required iterations needed in order to achieve the optimal CMP NoC design for optimization without any progressive technique, with *progressive capacity granularity* technique, and with *progressive number of links per iteration* technique. The progressive techniques might require more iterations for small scale NoCs (e.g., 3x2 CMP with *progressive number of links per iteration* technique). However, for large scale NoCs, the progressive techniques offer significant reduction of the total required iterations. Hence, these kinds of techniques allow the design of large scale NoCs with a reasonable number of iterations.

Figure 11 presents the design progress with *progressive number of links per iteration* technique versus number of iterations for the heterogeneous NoC-based CMP design presented in section 4.3. We present both the area reduction and the number of changed links versus number of iterations. The heterogeneous NoC-based CMP design achieves area reduction of 60% (compared to the modified homogeneous NoC) after 33% of the required iterations, and final area reduction of 80%.

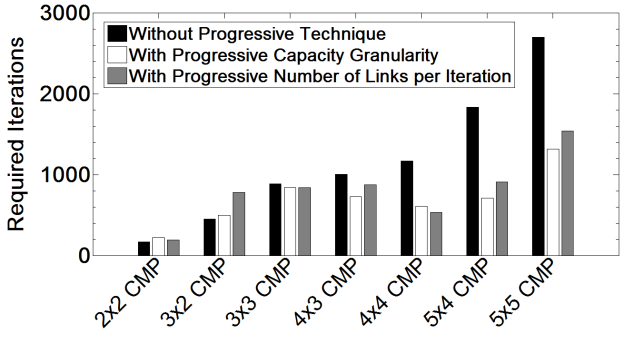

Figure 10. The required number of iterations needed to achieve optimal CMP NoC design (of the CMPs presented in Figure 9), for optimization without progressive technique, with *progressive capacity granularity* technique, and with *progressive number of links per iteration* technique.

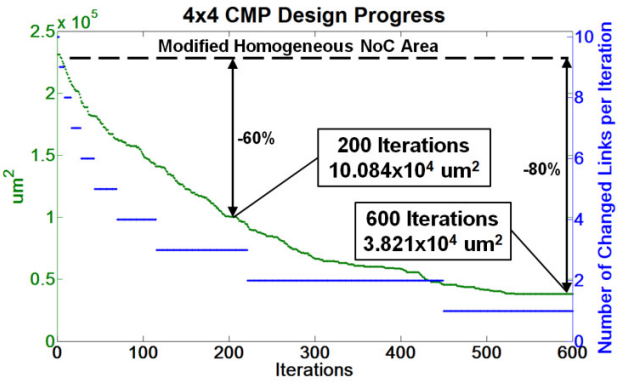

Figure 11. The heterogeneous NoC-based 4x4 CMP design (presented in section 4.3) using *progressive number of links per iteration* technique. NoC area and number of chanaged links per iteration versus number of iterations.

5. SUMMARY

A novel design methodology of heterogeneous NoC with different link capacities and different number of VCs has been presented. To that end, a heterogeneous NoC router architecture and a proper area evaluation have been proposed. In order to minimize the NoC area, an optimization procedure based on modified simulated annealing has been presented. Furthermore, a novel technique for an efficient support of end-to-end latency constraints in simulated annealing has been used in the optimization procedure. Scalability issues of our optimization method have been discussed and several techniques have been proposed. Heterogeneous NoC example which demonstrates a trade-off (and correlation) between link capacities and a number of VCs has been evaluated. The area savings achieved by our design have been demonstrated by SoCs running multimedia benchmarks, and by CMPs running PARSEC benchmarks. The design progress has been demonstrated for a synthetic NoC example and for a CMP design.

6. REFERENCES

[1] http://webee.technion.ac.il/matrics/software.html.

[2] A. Bakhoda, J. Kim, and T. Aamodt. Throughput-effective on-chip networks for manycore accelerators. In *Proceedings of the 2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 421–432. IEEE Computer Society, 2010.

[3] B. Beckmann and D. Wood. Managing wire delay in large chip-multiprocessor caches. In *Proceedings of the 37th annual IEEE/ACM International Symposium on Microarchitecture*, pages 319–330. IEEE Computer Society, 2004.

[4] Y. Ben-Itzhak, I. Cidon, and A. Kolodny. Delay analysis of wormhole based heterogeneous NoC. In *Proceedings of the fifth ACM/IEEE international symposium on Networks-on-Chip (NOCS 2011)*, 2011.

[5] Y. Ben-Itzhak, E. Zahavi, I. Cidon, and A. Kolodny. HNOCS: Modular Open-Source Simulator for Heterogeneous NoCs. To be published in: *SAMOS 2012,: International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XII)*. IEEE, 2012.

[6] D. Bertozzi, A. Jalabert, S. Murali, R. Tamhankar, S. Stergiou, L. Benini, and G. De Micheli. NoC synthesis flow for customized domain specific multiprocessor systems-on-chip. *IEEE Transactions on Parallel and Distributed Systems*, 16(2):113–129, 2005.

[7] C. Bienia, S. Kumar, J. Singh, and K. Li. The PARSEC benchmark suite: Characterization and architectural implications. In *Proceedings of the 17th international conference on Parallel architectures and compilation techniques*, pages 72–81. ACM, 2008.

[8] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny. QNoC: QoS architecture and design process for network on chip. *Journal of Systems Architecture*, 50(2-3):105–128, 2004.

[9] M. Dall'Osso, G. Biccari, L. Giovannini, D. Bertozzi, and L. Benini. Xpipes: a latency insensitive parameterized network-on-chip architecture for multiprocessor SoCs. In *Computer Design, 2003. Proceedings. 21st International Conference on*, pages 536–539. IEEE, 2003.

[10] W. Dally and B. Towles. *Principles and practices of interconnection networks*. Morgan Kaufmann, 2004.

[11] R. Das, S. Eachempati, A. Mishra, V. Narayanan, and C. Das. Design and evaluation of a hierarchical on-chip interconnect for next-generation CMPs. In *High Performance Computer Architecture, 2009. HPCA 2009. IEEE 15th International Symposium on*, pages 175–186. IEEE.

[12] Z. Guz, I. Walter, E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny. Network delays and link capacities in application-specific wormhole NoCs. *VLSI Design*, 2007.

[13] T. Huang, U. Ogras, and R. Marculescu. Virtual channels planning for networks-on-chip. In *Quality Electronic Design, 2007. ISQED'07. 8th International Symposium on*, pages 879–884. IEEE, 2007.

[14] A. Kahng, B. Li, L. Peh, and K. Samadi. Orion 2.0: A fast and accurate noc power and area model for early-stage design space exploration. In *Design, Automation & Test in Europe Conference & Exhibition, 2009. DATE'09.*, pages 423–428. IEEE, 2009.

[15] A. Kahng, B. Lin, K. Samadi, and R. Ramanujam. Trace-driven optimization of networks-on-chip configurations. In *Design Automation Conference (DAC), 2010 47th ACM/IEEE*, pages 437–442. IEEE, 2010.

[16] S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by simulated annealing. *science*, 220(4598):671, 1983.

[17] M. Kreutz, C. Marcon, L. Carro, F. Wagner, and A. Susin. Design space exploration comparing homogeneous and heterogeneous network-on-chip architectures. In *Proceedings of the 18th annual symposium on Integrated circuits and system design*, pages 190–195. ACM, 2005.

[18] D. Lattard, E. Beigné, F. Clermidy, Y. Durand, R. Lemaire, P. Vivet, and F. Berens. A reconfigurable baseband platform based on an asynchronous network-on-chip. *Solid-State Circuits, IEEE Journal of*, 43(1):223–235, 2008.

[19] M. Li, Q. Zeng, and W. Jone. DyXY: a proximity congestion-aware deadlock-free dynamic routing method for network on chip. In *Proceedings of the 43rd annual Design Automation Conference*, pages 849–852. ACM, 2006.

[20] P. Lotfi-Kamran, M. Daneshtalab, C. Lucas, and Z. Navabi. BARP-a dynamic routing protocol for balanced distribution of traffic in NoCs. In *Proceedings of the conference on Design, automation and test in Europe*, pages 1408–1413. ACM, 2008.

[21] T. Mak, P. Sedcole, P. Cheung, W. Luk, and K. Lam. A hybrid analog-digital routing network for NoC dynamic routing. In *Proceedings of the First International Symposium on Networks-on-Chip*, pages 173–182. IEEE Computer Society, 2007.

[22] A. Mishra, N. Vijaykrishnan, and C. Das. A case for heterogeneous on-chip interconnects for CMPs. In *Proceeding of the 38th annual international symposium on Computer architecture*, pages 389–400. ACM, 2011.

[23] F. Moraes, N. Calazans, A. Mello, L. Moller, and L. Ost. HERMES: an infrastructure for low area overhead packet-switching networks on chip. *Integration, the VLSI Journal*, 38(1):69–93, 2004.

[24] A. Varga et al. The OMNeT++ discrete event simulation system. In *Proceedings of the European Simulation Multiconference (ESM'2001)*, pages 319–324, 2001.

[25] C. Zeferino and A. Susin. SoCIN: a parametric and scalable network-on-chip. In *Integrated Circuits and Systems Design, 2003. SBCCI 2003. Proceedings. 16th Symposium on*, pages 169–174. IEEE, 2003.