Published in IET Computers & Digital Techniques
Received on 7th July 2008
Revised on 27th November 2008
doi: 10.1049/iet-cdt.2008.0095

In Special Issue on Networks on Chip

ISSN 1751-8601

Joint multicast routing and network design optimisation for networks-on-chip

S. Yan   B. Lin

Department of Electrical and Computer Engineering, University of California, San Diego, La Jolla, CA 92093–0407, USA
E-mail: shyan@ucsd.edu

Abstract: In this study, we consider the problem of synthesising custom networks-on-chip (NoC) architectures that are optimised for a given application. Both unicast and multicast traffic flows are considered in the input specification. We formulate the joint multicast routing and network design problem using a rip-up and reroute procedure, where each multicast routing step is formulated as a minimum directed spanning tree problem, and we propose a very efficient algorithm called Ripup-Reroute-and-Router-Merging (RRRM). Our new formulation adopts a rip-up and reroute concept that provides us with a heuristic iterative mechanism to identify increasingly improving solutions. The minimum directed spanning tree formulation efficiently captures the best routing solutions for multicast flows during the topology synthesis procedure. Our design flow integrates floorplanning, and our solutions consider deadlock-free routing. Experimental results compared with our previous proposed algorithms CLUSTER and DECOMPOSE on a variety of NoC benchmarks showed that our new synthesis results are largely improved. RRRM can on average achieve a 9% reduction in power consumption over CLUSTER and a 17% reduction in power consumption over DECOMPOSE with 1786$\times$ and 57$\times$ faster execution times than CLUSTER and DECOMPOSE, respectively. Improvements in performance were also achieved, with an average of 3% reduction in hop counts over CLUSTER and 7% in hop counts over DECOMPOSE on all benchmarks.

1 Introduction

Network-on-chip (NoC) architectures have been proposed as a scalable solution to the global communication challenges in nanoscale systems-on-chip (SoC) designs [1, 2]. The use of NoCs with standardised interfaces facilitates the reuse of previously designed and third-party-provided modules in new designs (e.g. processor cores). Besides design and verification benefits, NoCs have also been advocated to address increasingly daunting clocking, signal integrity and wire delay challenges.

NoC architectures can be designed as regular or custom network topologies. Regular topologies, such as mesh or folded-torus networks, have been successfully employed in a number of tile-based chip-multiprocessor projects, for example [3, 4], which are appropriate because of processor homogeneity and application traffic variability. On the other hand, for custom SoC applications, the design challenges are different in terms of varied module sizes, irregularly spread module locations and different communication data rate requirements. Therefore a custom network architecture optimised to the needs of the application is more appropriate. This synthesis problem is the focus of this paper.

The NoC synthesis problem is challenging for a number of reasons. First, for a large complex SoC design, an optimal solution will likely involve multiple networks since each module will likely communicate only with a small subset of modules. Therefore a single network that spans all nodes is often unnecessary. Part of the synthesis problem is to partition cores to groups, and connect each group to the same router so that they can share the network resources. It is hard to decide which cores should be partitioned into the same group. In general, cores may be grouped together and connect to the same routers even though they are not common sources or destinations of the same group of flows because they may be able to beneficially share common intermediate network resources. Also it is hard to decide the sizes of partitions beforehand, namely whether a design with a few larger routers would be more cost efficient than a design with more smaller routers. Second, besides deciding on connectivity of cores to routers, our synthesis problem must also decide on the connectivity and the physical network topology between routers. The network topologies that are tailored to a specific application as well as optimised to specific design goals are most wanted. Finally, depending on the optimisation goals and the implementation backend, the appropriate cost function may be quite complex. In particular, in this paper, we consider a power minimisation problem that considers both leakage power and dynamic switching power. It is well known that leakage power is becoming increasingly dominating [5, 6]. Therefore it is important to properly account for leakage power when adding routers and network links to the synthesised architecture. Other optimisation goals may include minimising hop counts along with power minimisation.

In this paper, we consider the problem of synthesising custom NoC architectures that support both unicast and multicast traffic flows. In general, there exist a variety of SoC applications. For many applications, support for multicast flows is necessary. Cases include, for example, the passing global states, the management and configuration of the network, and the implementation of cache coherency protocols. The work presented in this paper improves our previous work that was presented in [7, 8]. Our previous NoC synthesis algorithms were based on the formulation of the problem as set partitioning of traffic flows, finding a good network topology for each flow set using a Steiner tree formulation and providing an optimised network implementation for the derived topologies. All possible set partitions of flows are investigated in an intelligent way and a rectilinear Sterner tree problem is solved for each intermediate set partition, which makes those algorithms less efficient for future large applications with hundreds of cores envisioned.

In this paper, we formulate the joint multicast routing and network design problem using a rip-up and reroute procedure, where each multicast routing step is formulated as a minimum directed spanning tree problem. A key idea in our new formulation is a rip-up and reroute concept that has been successfully used in the very large scale integration (VLSI) routing problem [9–11]. The rip-up and reroute concept provides us with a heuristic iterative mechanism to identify increasingly improving solutions. There are two central differences between our on-chip network routing and design problem and the VLSI routing problem. The first is the ability to share network resources in our problem and the second is the difference in cost models. In the latter case, the costs of routers and links are not simple linear costs, and the sharing of network resources further complicates the optimisation process.

In particular, we propose a very efficient algorithm called Ripup-Reroute-and-Router-Merging (RRRM) that synthesises custom NoC architectures for supporting both unicast and multicast traffic flows. The algorithm is based on a rip-up–reroute formulation for routing flows to find a suitable network topology followed by a router merging procedure to optimise network topology. The key part of the algorithm is a rip-up and reroute procedure that routes multicast flows by way of finding the optimum multicast tree on a condensed multicast routing graph (MRG) using the directed minimum spanning tree formulation and the efficient algorithms [12, 13]. Then a router merging procedure follows to further optimise the network topology. In order to obtain the best topology solutions with minimum power consumption, accurate power models for interconnects and routers are derived. The Ripup–Reroute algorithm for routing flows and the Router-Merging algorithm to optimise topologies are based on using these power costs of network links and router ports as evaluation criteria. Our design flow integrates floorplanning and our synthesis process is both performance and power consumption aware. Our solutions also consider several ways of ensuring deadlock-free routing.

As has already been shown in our earlier work, our previous Steiner-tree-based formulation already significantly outperformed regular mesh and optimised mesh topologies. In comparison to our previous work, the performance of our new algorithm was able to achieve a relative reduction of up to 45% in terms of power consumption, up to 21% in terms of hop counts and up to 39% in terms of router area. More important, the execution times of our new algorithm are two–three orders of magnitude faster than the previous algorithms even for very large benchmarks.

The rest of the paper is organised as follows. Section 2 outlines related work. Section 3 presents our design flow, which incorporates floorplanning. Section 4 presents the problem description and our formulation. Sections 5 describes power models and Section 6 describes the details of the RRRM algorithm. Section 7 addresses deadlock considerations. Finally, experimental results and the conclusions are presented in Sections 8 and 9, respectively.

2 Related work

The NoC design problem has received considerable attention in the literature. Towles and Dally [1] and Benini and De Micheli motivated the NoC paradigm. Several existing NoC solutions have addressed the mapping problem to a regular mesh-based NoC architecture [14, 15]. Hu and Marculescu [14] proposed a branch-and-bound algorithm for the mapping of computation cores on to mesh-based NoC architectures. Murali and De Micheli [15] described a fast algorithm for mesh-based NoC architectures that considers different routing functions, delay constraints and bandwidth requirements.

On the problem of designing custom NoC architectures without assuming an existing network architecture, a number of techniques have been proposed [16–21]. Pinto *et al.* [18] presented techniques for the constraint-driven communication architecture synthesis of point-to-point links by using heuristic-based *k*-way merging. Their technique is limited to topologies with specific structures that have only two routers between each source and sink pair. Ogras and Marculescu [16, 17] proposed graph decomposition and long link insertion techniques for application-specific NoC architectures. Srinivasan *et al.* [19, 20] presented NoC synthesis algorithms that consider system-level floorplanning, but their solutions only considered solutions based on a slicing floorplan where router locations are restricted to corners of cores and links run around cores. Murali *et al.* [21] presented an innovative deadlock-free NoC synthesis flow with detailed backend integration that also considers the floorplanning process. The proposed approach is based on the min-cut partitioning of cores to routers. Yan and Lin [7, 8] formulated the custom NoC synthesis problem based on the set partitioning of traffic flows and finding good network topologies using a Steiner tree formulation.

Multicasting in wormhole-switched networks has been explored in the context of chip multiprocessors based on the methods in parallel machines for supporting cache coherency, acknowledgement collection, synchronisation etc [22, 23]. In the NoC works of [24, 25], multicast service can be implemented in their NoC architectures. However, the methods for providing multicast routing and services have not been presented in detail. In [26], a novel multicast scheme in wormhole-switched NoCs using a connection-oriented technique to realise QoS-aware multicasting in a best-effort network was proposed to support SoC applications. In [27], a router architecture supporting unicast and multicast services was proposed using a mechanism for managing broadcast flows so that the communication links in an on-chip network can be shared. In [28], the dual-path multicast algorithm, used in multicomputers, was adapted to wormhole-switched NoCs to support deadlock-free multicast routing.

This paper presents an improved synthesis algorithm over our previous work. The approach is based on flow rip-up–rerouting formulation and router merging scheme that considers both unicast and multicast traffic, which to the best of our knowledge has not been considered in previous custom NoC synthesis formulations other than ours. Our approach considers deadlock-free routing with multicast traffic as well as considers floorplanning in the design flow. Our approach represents a different way of formulating the custom NoC synthesis problem. Given that custom NoC synthesis is still a relatively new problem, we believe that our work provides an interesting direction in this research area.

3 Design flow

Our NoC synthesis design flow is depicted in Fig. 1 and the details were discussed in [8]. The major elements in the design flow are briefly summarised.

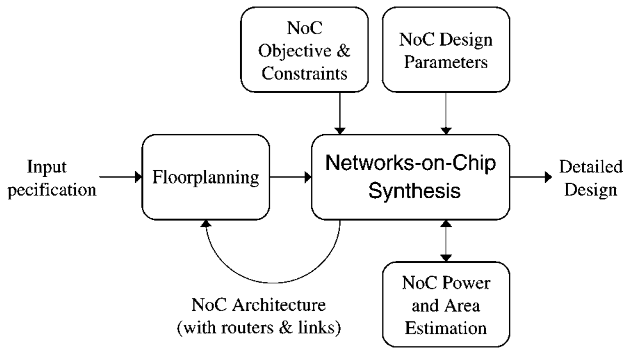
NoC Objective & Constraints
NoC Design Parameters
Input pecification
Floorplanning
Networks-on-Chip Synthesis
Detailed Design
NoC Architecture (with routers & links)
NoC Power and Area Estimation

Figure 1 *Design flow*

1. Input specification: The input specification to our design flow consists of a list of modules and their communications. Modules can correspond to a variety of different types of intellectual property (IP) cores in a variety of sizes and can be either hard or soft macros. Packet-based communication with standard network interfaces is considered and custom NoC architectures are addressed in this paper as a scalable solution. Traffic flows with required data rates between modules are specified as part of the input specification. For our synthesis problem, we consider both unicast and multicast traffic flows. In general, a mixture of network-based communications and conventional wiring may be utilised as appropriate, and not all inter-module communications are necessarily over the on-chip network. Our design flow and input specification allow for both interconnection models.

2. Floorplanning: The floorplanning problem has been extensively studied with many mature solutions (e.g. [29–31]). In our design flow, we have adopted the open source floorplanner Parquet [31]. An initial floorplanning step is performed before NoC synthesis to obtain a placement of modules. This is important because the floorplanning of modules is often influenced by non-network-based interconnections, and the floorplan locations of modules can have a significant influence on the NoC architecture. With the module locations available from the initial floorplanning step, NoC synthesis can better account for wiring delays and power consumptions during the exploration of NoC architectures. During the NoC architecture synthesis, routers are positioned close to the network interface of the IP cores. After NoC synthesis, actual routers and links in the synthesised NoC architecture can be fed back to the floorplanner to update the floorplan. The refined floorplan information can be used to obtain more accurate power and area estimates. After the floorplan has been updated, NoC synthesis can be re-invoked to consider more accurate placement information. As shown experimentally in Section 8, our NoC synthesis algorithms are fast, making it feasible to iterate NoC synthesis with floorplanning.

3. NoC synthesis: Given floorplanning information, the NoC synthesis step then proceeds to synthesise an NoC architecture

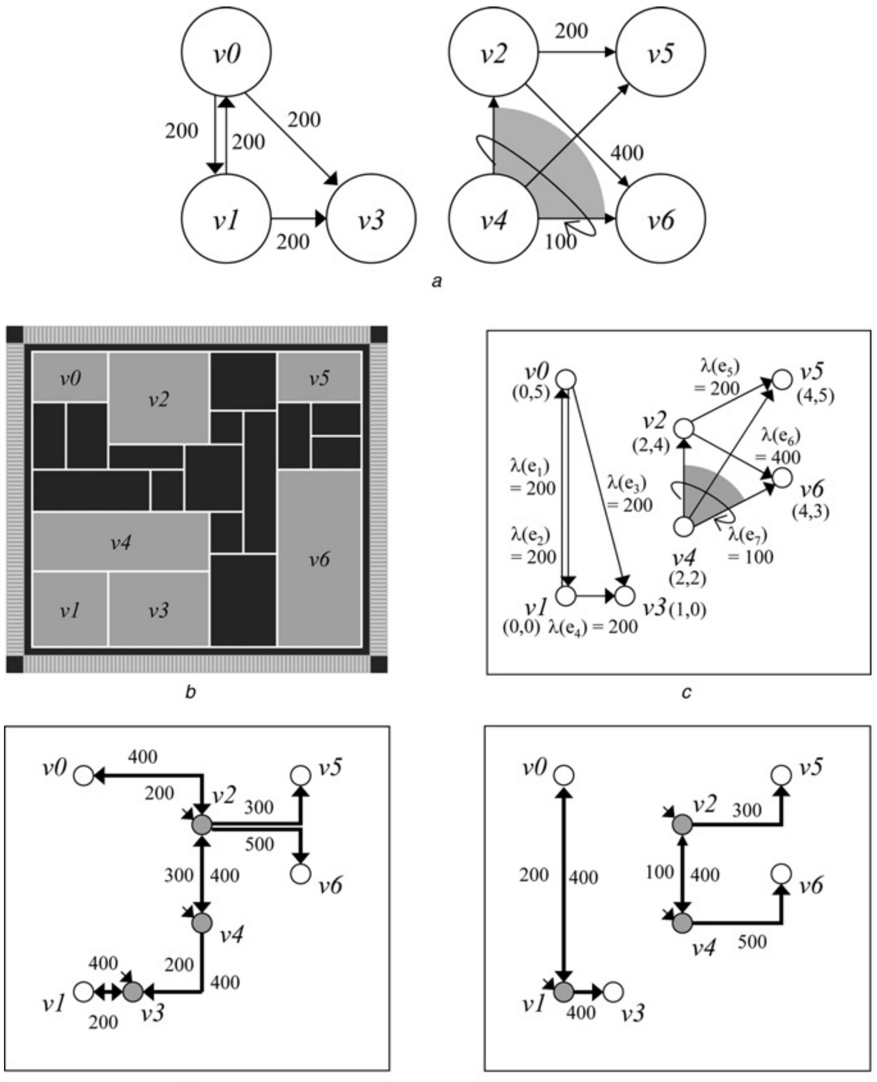

Figure 2 *Illustration of the NoC synthesis problem*

a Example
b Floorplan
c CDG
d One architecture
e Alternative architecture

that is optimised for the given specification and floorplan. Consider Fig. 2*a*, which depicts a small illustrative example. Fig. 2*a* only shows the portion of the input specification that corresponds to the network-attached modules and their traffic flows. The nodes represent modules, edges represent traffic flows, and edge labels represent the data rate requirements for the corresponding flows. Multicast traffic flows are represented with directed hyperedges, which are shown graphically in Fig. 2*a* as a bundle of directed edges in a shaded region. For example, the traffic flow from v_4 to v_2, v_5 and v_6 is a multicast flow. This graph representation is called a communication demand graph (CDG) and is discussed in more detail in Section 4.

An example floorplan is shown in Fig. 2*b*. As noted earlier, modules in a design do not necessarily have to be attached to the on-chip network. Modules can also be connected by conventional wiring, as shown in the unlabelled rectangles in Fig. 2*b*. The CDG with the floorplan positions annotated is illustrated in Figs. 2*c*, and Figs. 2*d* and *e* show two example network topologies.

4. NoC objective and constraints: Our NoC synthesis design flow allows different user-defined objective and constraints. As power dissipation becomes a critical issue in future IC designs due to increased design complexity, we focus in this paper on the problem of minimising network power consumption under performance constraints. Another possible design objective is the minimisation of hop counts for data routing under power consumption constraints. Other possible constraints can be design area, total wire length or some combinations of them.

5. NoC design parameters: In addition to user-defined objectives and constraints, NoC design parameters such as operating voltage, target clock frequency and link widths are provided to the NoC synthesis step as well. If the design allows for different voltages or clock frequencies, or if the IP modules allow for different link widths, then NoC synthesis can be invoked to synthesise solutions for a range of design parameters specified by the user.

6. Detailed design: Finally, the synthesised NoC architecture with the rest of the design specification can be fed to a detailed RTL design flow where design tools like RTL optimisation and detailed place and route are well established.

4 Problem description and formulation

4.1 Problem description

The input to our NoC synthesis problem is a CDG, defined as follows:

Definition 1: A CDG is an annotated directed hypergraph $H(V, E, \pi, \lambda)$, where each node $v_i \in V$ corresponds to a module, and each directed hyperedge $e_k = s \to D \in E$ represents a traffic flow from source $s \in V$ to one or more destinations $D = \{d_1, d_2, \ldots\}$, $D \subseteq V$. The position of each node v_i is given by $\pi(v_i) = (x_i, y_i)$. The data rate requirement for each communication flow e_k is given by $\lambda(e_k)$.

In general, traffic flows can be either unicast or multicast flows. Multicast flows are flows with $|D| > 1$. For example, in Fig. 2*c*, e_7 corresponds to a multicast flow from source v_4 to destinations v_2, v_5 and v_6.

Based on the optimisation goals and cost functions specified by the user, the output of our NoC architecture synthesis problem is an optimised custom network topology with pre-determined routes for specified traffic flows on the network such that the data rate requirements are satisfied. For example, Figs. 2*d* and *e* show two different topologies for the CDG shown in Fig. 2*c*.

Fig. 2*d* shows a network topology where all flows share a common network. In this topology, the pre-determined route for the multicast flow e_7 travels from v_4 to v_2 to first reach v_2, and then it bifurcates at v_2 to reach v_5 and v_6. Fig. 2*e* shows an alternative topology comprising two separate networks. In this topology, the multicast flow e_7 bifurcates in the source node to reach v_6, then it is transferred over the network link between v_4 to v_2 to reach v_2, and then bifurcates to reach v_5. Observe that in both cases, the amount of network resources consumed by the routing of multicast traffic is less than what would be required if the traffic is sent to each destination as a separate unicast flow.

4.2 Problem formulation

In general, the solution space of possible application-specific network architectures is quite large. Depending on the communication demand requirements of the specific application under consideration, the best network architecture may indeed be comprised of multiple networks, among each, many flows sharing the same network resources.

The goal of the proposed work in this paper is to find an optimised network topology such that the communication bandwidth requirements are satisfied and the power consumption of the network is minimised. In order to obtain the best topology solutions with minimum power consumption, accurate power models for interconnects and routers are derived. They are given to the synthesis of design flow as a library and utilised by the synthesis algorithm as evaluation criteria.

The application-specific NoC synthesis problem can be formulated as follows:

Input:

- The CDG $H(V, E, \pi, \lambda)$ of the application.

- The NoC network component library $\Phi(I, J)$, where I provides the power and area models of routers with different sizes, and J provides power models of physical links with different lengths.

- The target clock frequency, which determines the delay constraint for links between routers.

- The floorplanning of the cores.

Output:

- An NoC architecture $T(R, L, C)$, where R denotes the set of routers in the synthesised architecture, L represents the set of links between routers, and a function $C: V \to R$ represents the connectivity of a core to a router.

- A set of ordered paths P, where each $p_{ij} \in P = (r_i, r_j, \ldots, r_k)$, $r_i, \ldots, r_k \in R$, represents a route for a traffic flow $e(v_i, v_k) \in E$.

Objective:

- The minimisation of power consumption for the synthesised NoC architecture.

5 Power models

In nanoscale technologies, minimising power consumption is a very important design goal along with performance maximisation. In this paper, the design goal of the NoC synthesis problem is to construct an optimised interconnection architecture such that the communication requirements are satisfied and the power consumption is minimised.

The total power consumption of the communication architecture includes both leakage power and dynamic switching power of the routers and links. The dynamic switching power is a function of data rate passing through each component and the leakage power is related to the type and characteristics of the components in the NoC architecture.

We will discuss the details of modelling these components in the following sections.

5.1 Modelling routers

It is well known that leakage power is becoming increasingly dominating [6]. In the on-chip network studied in [6], leakage power represented only about 0.6 and 1.8% of the total power consumption at 180 and 100 nm, respectively, but leakage power increased to 25% at 70 nm. High-performance microprocessor studies show even a much larger leakage power component [5]. Therefore it is important to properly account for leakage power when adding routers and channels to the synthesised architecture. However, when considering leakage power, the cost function may need to account for possibly discrete cost increments of links and routers whereas dynamic switching power may be best modelled as a function of cumulative data rates. This non-linear characteristic of the power consumption of the NoC makes it hard to be accurately modelled using mixed-integer linear programming (MILP) or linear programming (LP) formulations.

To evaluate the power of the routers in the synthesised NoC architecture, we use a state-of-the-art NoC power-performance simulator called Orion [6, 32] that can provide detailed power characteristics for different power components of a router for different input/output port configurations. It accurately considers leakage power as well as dynamic switching power. The power per bit values are used as the basis for the entire router dynamic power estimation under different configurations. The leakage power and switching bit energy of some example router configurations with a different number of ports in 70 nm technology are shown Table 1.

5.2 Modelling interconnects

In the NoC architecture, interconnects can be modelled as distributed RC wires. As discussed in Section 3, the target clock frequency is provided to our NoC synthesis design flow as a design parameter. Depending on the network topology, long interconnects may be required to implement network links between routers, which may have wire delays that are larger than the target clock frequency. To achieve the target frequency, repeaters may need to be inserted. Thus, we use the state-of-art repeated on-chip interconnect model [33, 34], where the interconnect is evenly divided into k segments with repeaters inserted between them that are s times as large as a minimum-sized repeater. The delay and power consumption per bit of this interconnect can be modelled using the Elmore model, as in [33, 34]. When minimising power consumption is the objective, the optimum size s_{opt} and number k_{opt} of repeaters that minimise power consumption while satisfying the delay constraint can be determined for the interconnect using the method proposed in [30].

Table 2 Interconnet parameters

Electrical	Physical
$\rho = 2.53\ \mu\Omega$ cm $k_{\text{ILD}} = 2.7$	$w = 500$ nm $s = 500$ nm
$r_h = 46\ \Omega/\text{mm}$ $c_h = 192.5\ fF/$mm	$t = 1100$ nm h = 800 nm

In our experiments, the physical and electrical parameters in 70 nm technology are used and are listed in Table 2. The wires are implemented on the global metal layers and their parameters are extracted from international technology roadmap for semiconductors (ITRS) [35].

In our NoC synthesis design flow, we use the above interconnect model to evaluate optimum power consumption of interconnects with different wire lengths under the given design frequency and delay constraints. These results are provided to the design flow in the form of a library. Since the floorplanning is performed in advance of NoC synthesis, wirelength is known for each on-chip interconnect when evaluating the power consumption.

Table 3 lists the static power and switching bit energy parameters of some example interconnects with different wirelengths in 70 nm technology under 1 GHz frequency constraints.

6 Design algorithms

In this section, we present algorithms for the NoC topology synthesis process. The entire process is a joint multicast routing and network design procedure that consists of the

Table 1 Power consumption of routers using Orion [6]

Ports (in × out)	2 × 2	3 × 2	3 × 3	4 × 3	4 × 4	5 × 4	5 × 5
leakage power (W)	0.0069	0.0099	0.0133	0.0172	0.0216	0.0260	0.0319
switching bit energy (pJ/bit)	0.3225	0.0676	0.5663	0.1080	0.8651	0.9180	1.2189

Table 3 Power consumption of interconnects

Wire length (mm)	1	4	8	12	16
leakage power (W)	0.000496	0.001984	0.003968	0.005952	0.007936
switching bit energy (pJ/bit)	0.6	2.4	4.8	7.2	9.6

inter-related steps of constructing an initial network topology, rip-up and rerouting multicast flows to design the network topology, inserting the corresponding network links and router ports to implement the routing, and merging routers to optimise network topology based on design objectives. In particular, we propose an algorithm called RRRM. The details of the algorithm are discussed in this section.

6.1 Initial network construction

The details of RRRM are described in Algorithm 1. RRRM takes a CDG and an evaluation function as inputs and generates an optimised network architecture as output. It starts with initialising a network topology by a simple router allocation and flow routing scheme. Then it uses a procedure of rip-up and rerouting flows to refine and optimise the network topology. After that, a router merging step is done to further optimise the topology to obtain the best result.

In the initialisation, every flow is routed using its own network. To construct an initial network topology, a router is allocated at each core and placed close to the location of the network interface. These routers are not actual routers that will be included in the network topology. Only those that have traffic either multiplexed from more than two ports to the same port or de-multiplexed from one port to more than two ports at the end of the RRRM procedure will be included. After router allocation, a routing cost graph (RCG) is generated (Algorithm 1 line 2). RCG is a very important graph used in the whole rip-up and reroute procedure of the RRRM algorithm.

Definition 2: The RCG(R, E) is a weighted directed complete graph, where each vertex $r_i \in R$ represents a router and each directed edge $e_{ij} = (r_i, r_j) \in E$ from r_i to r_j corresponds to a connection from r_i to r_j. A weight $w(e_{ij})$ is attached to each edge, which represents the incremental cost of routing a flow f through e_{ij}.

Note that RCG does not represent the actual physical connectivity between different routers and its edge weights change during the whole Ripup–Reroute procedure for different flows. Also, the actual physical connectivity between the routers is established during the Ripup–Reroute procedure, which is explained in the following sections.

Before Ripup–Reroute, initial network topology is constructed using the InitialNetworkConstruction() procedure. Each flow $e_k = (s_k, d_k)$ in the CDG is routed using a direct connection from router r_{s_k} to router r_{d_k}, where r_i is the router that core i connects to, and the path is saved in path(e_k). Multicast flows are routed as a sequence of unicast flows from the source to each of their destinations. The links and router ports are configured and saved. If a connection between routers cannot meet the delay constraints, its corresponding edge weight in RCG is set to infinity. This can be used to guide the rerouting of flows to use other valid links instead of this one in the Ripup–Reroute procedure.

As an example, after initial network construction, the connectivity of routers for the example shown in Fig. 2*a* is shown in Fig. 3*a*.

Algorithm 1 RRRM$(G(V, E, \pi, \lambda), C, L)$

```
Input: $G(V, E, \pi, \lambda)$: CDG, $C$: cost function,
    $L$: library of network components
Output: $T$: synthesized network topology
1: $R$ = InitialRouterAllocation($G$)
2: $RCG$ = ConstructFullyConnectedGraph($R$)
3: ($links, routers$) = InitialNetworkConstruction($G, RCG$)
4: ($links, routers$) = Ripup-Reroute($G, RCG, C, L$)
5: $cost$ = EvaluatePowerConsumption($links, routers$)
6: Router-Merging($cost, links, routers$)
7: $T$ = ObtainBestTopology($links, routers$)
8: return $T$

InitialNetworkConstruction($G, RCG$)
1: for all flow $e_k = (s_k \to D_k) \in E$ do
2:   for all destination $d_{ki} \in D_k$ do
3:     route $e_k$ using a direct connection between $r_{s_k}$ and $r_{d_{ki}}$
4:     if $link(r_{s_k}, r_{d_{ki}})$ exists, update bandwidths of links, routers ports
5:     $path(e_k) = (r_{s_k} \to r_{d_{ki}})$
6:     update routers $r_{s_k}, r_{d_{ki}}, link(r_{s_k}, r_{d_{ki}})$
7:     if $link(r_{s_k}, r_{d_{ki}})$ not meet delay constraint, update $wgt(r_{s_k}, r_{d_{ki}}) = \infty$ in $RCG$
8:   end for
9: end for
```

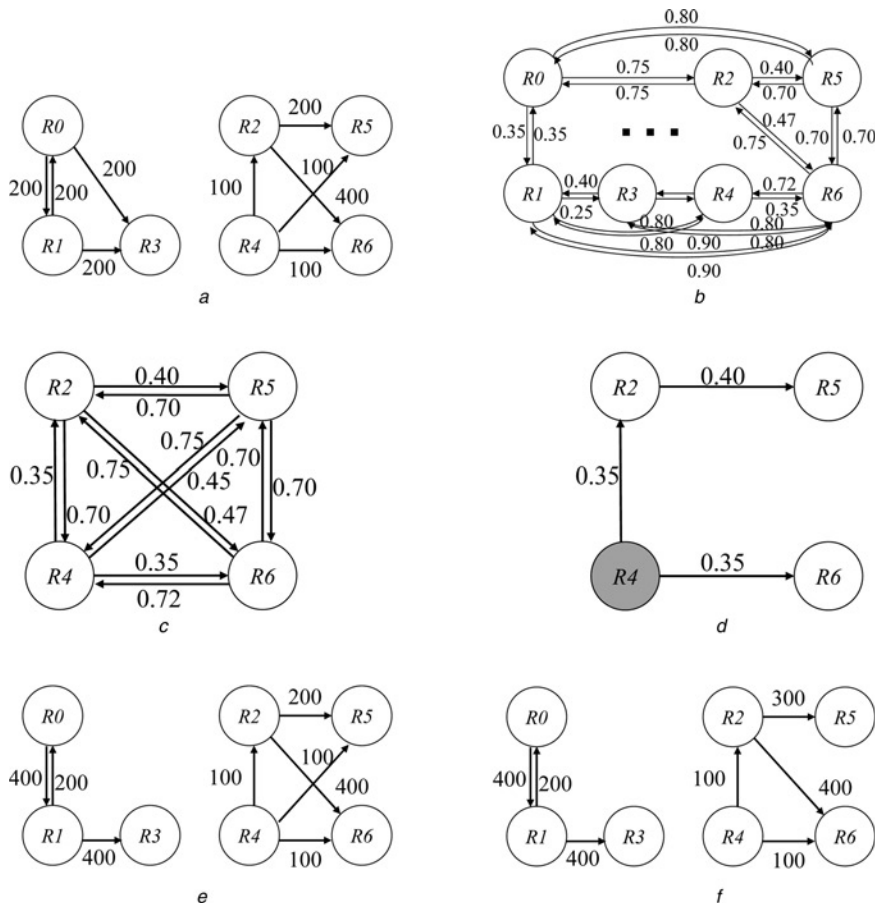

Figure 3 *Illustration of the Ripup–Reroute procedure*

a Initial connectivity
b RCG
c MRG
d MRTree
d Connectivity before reroute e_7
f Connectivity after reroute e_7

6.2 Flow rip-up and rerouting

Once the initial network is constructed and the initial flow routing is done, the key procedure of the – Ripup–Reroute algorithm procedure is invoked to route flows and find an optimised network topology.

The details of Ripup–Reroute are described in Algorithm 3. In the Ripup–Reroute procedure, each multicast routing step is formulated as a minimum directed spanning tree problem. Two important graphs, MRG and multicast routing tree (MRTree), are used to facilitate the rip-up and rerouting procedure. They are defined as follows.

Definition 3: Let f be a multicast flow with source $s \in V$ and one or more destinations $D \subseteq V$. that is, $D = \{d_1, d_2, \ldots, d_{|D|}\}$, each $d_i \in V$.

An MRG is a complete graph $\Gamma(N, A)$ defined for f as follows:

- $N = s \cup D$.
- There is a directed arc between every pair of nodes (i, j) in N. Each arc $a_{i,j} \in A$ corresponds to a shortest path $p_{i,j}$ between the same nodes in the corresponding RCG, $p_{i,j} = e_1 \rightarrow e_2 \rightarrow \cdots \rightarrow e_k$.
- The weight for arc $a_{i,j}$, $w(a_{i,j})$, corresponds to the path weight of the corresponding shortest path $p_{i,j}$ in *RCG*, that is,

$$w(a_{i,j}) = \sum_{e_i \in p} w(e_i)$$

Definition 4: An MRTree is the minimum directed spanning tree for MRG $\Gamma(N, A)$ with $s \in N$ as the root.

When a flow is ripped up and rerouted, its current path is deleted and the links and router ports it occupies are released (line 3). Then based on the current network connectivity and resources occupation, the RCG related to this flow is built and the weights of all edges in RCG are updated (line 4). In particular, for every pair of routers in RCG, the cost of using those routers and the link connecting them is evaluated. This cost depends on the sizes of the routers, the traffic already routed on the routers and the connectivity of the routers to other routers. It also depends on whether an existing physical link will be used or a new physical link needs to be installed. If there are already router ports and links that can support the traffic, the marginal cost of reusing those resources is calculated. Otherwise, the cost of opening new router ports and installing new physical links to support the traffic is calculated. The cost is assigned as an edge weight to the edge connecting the pair of routers in RCG. If the physical links used to connect the routers cannot satisfy the delay constraints, a weight of infinity is assigned to the corresponding edges in RCG.

Once the RCG is constructed, the MRG for the flow is generated from RCG (line 5). MRG is built by including every source and destination router of the flow as its nodes. For each pair of nodes in MRG, the least cost directed path with least power consumption on RCG is found for corresponding routers using Dijkstra's shortest path algorithm, and the cost is assigned as an edge weight to the edge connecting the two nodes in MRG. Then the Chu-Liu/Edmonds algorithm [12, 13] is used to find the rooted directed minimum spanning tree of MRG with the source router as root. A rooted directed spanning tree of a graph is defined as a graph that connects, without any cycle, all n nodes in the graph with $n-1$ arcs such that the sum of the weight of all the arcs is minimised. Each node, except the root, has one and only one incoming arc. This directed minimum spanning tree is obtained as the MRTree, so that the routes of the multicast flow follow the structure of this tree. The details of the Chu-Liu/Edmonds algorithm are summarised in Algorithm 3. The multicast routing for flow f in RCG can be obtained by projecting MRTree back to RCG by expanding the corresponding arcs to paths. A special case is when f is a unicast flow with source s and destination d. In this case, MRG will just consist of two nodes, namely s and d, and one directed arc from s to d. Therefore the routing between s and d in RCG is simply a shortest path between s and d.

After the path is determined, the routers and links on the chosen path are updated.

As an example, Fig. 3*b* shows the RCG for rerouting the multicast flow e_7. For clarity, only part of the edges are shown for RCG. The MRG and MRTree for e_7 are shown in Figs. 3*c* and *d*, respectively. By projecting MRTree back to RCG, the routing path for e_7 is determined, namely e_7 bifurcates in the source router R_4 to reach R_6 and v_6, then it is transferred over the network link between R_4 to R_2 to reach v_2, and then bifurcates to reach R_5 and v_5. The real physical connectivities between routers before and after rip-up and rerouting e_7 are also shown in Figs. 3*e* and *f*. From them, we observe that the links between R_4 and R_5 and their corresponding ports are saved; thus the power consumptions are reduced after rerouting e_7 by utilising the existing network resources for routing other flows.

This Ripup–Reroute process is repeated for all the flows. The results of this procedure depend on the order in which the flows are considered, so the entire procedure can be repeated several times to reduce the dependency of the results on flow ordering. (In the experiments, we have tried several flow ordering strategies such as largest flow first, smallest flow first, random ordering etc., and we found that the ordering of smallest flow first gave the best results. Thus we used this ordering in our experiments. Also, we observed that repeating the whole Ripup–Reroute procedure twice is enough to generate good results.) Once the path of each flow is decided, the size of each router and the links that connect the routers are determined. Routers that have no traffic multiplexing or de-multiplexing are deleted and links are reconnected. The remaining routers and links constitute the network topology. The total implementation cost of all the routers and links in this topology is evaluated and the network topology is obtained.

6.3 Router merging

After the physical network topology has been generated using Ripup–Reroute, a router merging step is used to further optimise the topology to reduce the power consumption cost. The router merging step was first proposed by Srinivasan *et al.* [20]. Their router merging was based on the distance between routers. However, in this paper, we propose a new router merging algorithm for reducing the power consumption of the network and improving the performance. As has been observed, routers that connect to each other can be merged to eliminate router ports and links and thus possibly the corresponding costs. Routers that connect to the same common routers can also be merged to reduce ports and costs. We propose a greedy router merging algorithm, which is shown in Algorithm 4. The algorithm works iteratively by considering all possible mergings of two routers connected to each other. In each iteration, each router's adjacent routers are constructed and sorted by the distance between them in increasing order. They are possible candidate mergings. Then the routers are considered to merge in decreasing order of the number of neighbours they have. For each candidate merging, if the topology from the merging result is valid, the total power consumption of the resulting topology after merging is evaluated using the power models. Routers are merged if they have not merged in this iteration and the cost is improving. After all routers are considered in the current iteration, they are updated by replacing the routers merged with the new one generated. Those routers are reconsidered in the next iteration. The algorithm keeps merging routers until no further improvement can be made. After router merging, the optimised topology is generated and the routing paths of all flows are updated. Since router merging will always reduce the number of routers in the topology, it will not increase the hop counts for all the flows and thus will not worsen the performance of the application. The topology generated after router merging represents the best solution with the minimum power consumption. It is returned as the final solution for our NoC synthesis algorithm.

Algorithm 2 Ripup-Reroute$(G(V,E,\pi,\lambda), RCG, C, L)$

```
Input: $G(V,E,\pi,\lambda)$: CDG, $RCG$: router cost graph,
    $C$: cost function, $L$: library of network components
Output: $links, routers$: links and routers of synthesized network topology
 1: while need another round do
 2:   for all flow $e_k \in E$ in increasing order of $\lambda(e_k)$ do
 3:     delete $path(e_k)$ and release the link and router resources it occupied
 4:     update all edge weights in RCG for flow $e_k$, according to power consumption of the corresponding links and routers
        resources
 5:     $MRG(e_k)$ = ConstructMulticastRoutingGraph($RCG$)
 6:     $MTree(e_k)$ = FindMulticastTree($MRG(e_k)$)
 7:     $path(e_k)$ = Find paths from $s_k$ to $d_{ki} \in D_k$ in $MTree(e_k)$
 8:     Update $link, BW\_avail, routers$ for $path(e_k)$
 9:   end for
10: end while
11: return $links, routers$
```

Algorithm 3 DirectedMinimumSpanningTree$(G(N,A))$

```
1: Discard the arcs entering the root if any; For each node other than the root, select the entering arc with the smallest cost;
   Let the selected $n-1$ arcs be the set $S$.
2: If no cycle formed, $G(N,S)$ is a MST. Otherwise, continue.
3: For each cycle formed, contract the nodes in the cycle into a pseudo-node $(k)$, and modify the cost of each arc which enters
   a node $(j)$ in the cycle from some node $(i)$ outside the cycle according to the following equation.
       $c(i,k) = c(i,j) - (c(x(j),j) - min_j(c(x(j),j)))$
   where $c(x(j),j)$ is the cost of the arc in the cycle which enters $j$.
4: For each pseudo-node, select the entering arc which has the smallest modified cost; Replace the arc which enters the same
   real node in $S$ by the new selected arc.
5: Go to step 2 with the contracted graph.
```

Algorithm 4 Router-Merging(R,T)

```
Input: $R$: routers list $T$: network topology
Output: $R'$: new routers list; $T'$: new network topology
 1: $done = 0$
 2: while $done = 0$ do
 3:   for all $r_i \in R$ do
 4:     $adj(r_i)$ = generate all 1-hop adjacent router list and sort it by their distance to $r_i$ in increasing order
 5:   end for
 6:   sort routers in $R$ by their number of adjacent routers in decreasing order
 7:   for all $r_i \in R$ in this order do
 8:     for all $r_j \in adj(r_i)$ do
 9:       if neither $r_i$ nor $r_j$ is merged in this round then
10:         evaluate the total power consumption cost of merging $r_i$ and $r_j$
11:         merge $r_i$, $r_j$ to $r'$ if merging is valid and total power consumption is improving
12:         delete $r_i$, $r_j$ from $R$ and add $r'$ to $R$
13:       end if
14:     end for
15:   end for
16:   if no merging is done in this iteration then
17:     $done = 1$
18:   end if
19: end while
20: update topology $T'$ and routing path for all flows
21: return $R' = R, T'$
```

As an example, connectivity graphs before and after the Router-Merging procedure for the example of Fig. 2*a* are shown in Figs. 4*a* and *b*. It is shown that after router merging, the network resources are reduced from four routers to three routers and the total power consumption is reduced as well.

6.4 Complexity of the algorithm

For an application with $|V|$ IP cores and $|E|$ flows, the initial network construction step needs $O(|E|)$ time. In the rip-up and reroute procedure, each flow is ripped up and rerouted once. The edge weight calculation for the router cost graph

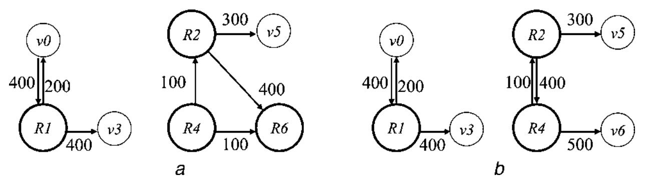

Figure 4 *Illustration of the Router-Merging procedure*

a Before router merging
b After router merging

takes $O(|V|^2)$. For a multicast flow with m destinations, the construction of MRG takes $O((m+1)^2|V|^2)$ by finding the shortest path between each pair of nodes. Then it takes $O(|V|^2)$ to find the rooted directed minimum spanning tree as the multicast tree by using the Chu-Liu/Edmonds algorithm. Hence, the overall complexity of our algorithm is $O(|E||V|^2)$.

7 Deadlock considerations

Deadlock-free routing is an important consideration for the correct operation of custom NoC architectures. In our previous work [7, 8], we have proposed two mechanisms to ensure the deadlock-free operation in our NoC synthesis results. In this paper, we adopt the same mechanisms in our new Noc synthesis algorithm to ensure deadlock-free operation in the deterministic routing problem we consider.

The first method is statically scheduled routing. For our NoC solutions, the required data rates are specified and the routes are fixed. In this setting, data transfers can be statically scheduled along the pre-determined paths with resource reservations to ensure deadlock-free routing [36, 37]. The second method is virtual channels insertion. As shown in [38], a necessary and sufficient condition for deadlock-free routing is the absence of cycles in a channel dependency graph. In particular, we use an extended channel dependency graph construction to find resource dependencies between multicast trees (this extended channel dependency graph construction treats unicast flows as a special case) and break the cycles by splitting a channel into two virtual channels (or by adding another virtual channel if the physical channel has already been split). The added virtual channels are implemented in the corresponding routers. We applied this method into our NoC synthesis procedure and found that virtual channels are rarely needed to resolve deadlocks in practice for custom networks. In all the benchmarks that we tested in Section 8, no deadlocks were found in the synthesised solutions. Therefore we did not need to add any virtual channel.

8 Results

8.1 Experimental setup

We have implemented our proposed algorithm RRRM in C++. As discussed in the design flow outlined in Section 3, we use Parquet [31] for the initial floorplanning step.

In all our experiments, we aim to evaluate the performance of our algorithm RRRM on all benchmarks with the objective of minimising the total power consumption of the synthesised NoC architectures. The total power consumption includes both the leakage power and the dynamic switching power of all network components. As discussed in Section 3, we use a power-performance simulator called Orion [6, 32] to estimate the power consumptions of the router configurations generated. We applied the design parameters of 1 GHz clock frequency, four-flit buffers and 128-bit flits. For the link power parameters, we use the state-of-art on-chip repeated interconnect model [33, 34] to evaluate the optimum powers for links with different lengths under the given delay constraint of 1 ns. Both routers and links are evaluated using 70 nm technology and are provided in a library.

As has already been shown in our earlier work, our previous Steiner-tree-based formulation and the proposed four algorithms already significantly outperformed regular mesh and optimised mesh topologies. Specifically, the two heuristic algorithms CLUSTER and DECOMPOSE could achieve similar results as the other probabilistic algorithms but with faster execution times.

Therefore in the experiments in this paper, in order to evaluate the effectiveness of our new algorithm, we applied RRRM on the same sets of benchmarks used in [7, 8] and compared its synthesis results with the results of CLUSTER and DECOMPOSE. We do not repeat here the comparisons with mesh-based topologies since our new formulation already outperforms our earlier work. In particular, in order to emphasise the benefit and efficiency of our new algorithm on large benchmarks, we picked up those benchmarks with the number of cores larger than 15 and reported their results in this paper. The results show that the algorithm RRRM outperforms CLUSTER and DECOMPOSE in both power consumption and performance with execution times two to three orders of magnitude faster. The details of the results are discussed in the following sections.

The same two groups of benchmarks were used. The first group of benchmarks was used to evaluate the performance of our algorithm on applications with only unicast flows. It consists of a generic multimedia system and several applications of the combinations of four different video processing applications obtained from [39], namely VOPD, MPEG4, PIP and MWD. The names of the benchmarks represent the abbreviations of the names they include, for example V + M means benchmark including VOPD and MWD applications and so on. The second group of benchmarks was used to evaluate the performance of our algorithm on benchmarks with multicast traffic flows. In the absence of published benchmarks with multicast traffic, we generated a set of synthetic benchmarks using the NoC-centric bandwidth version of Rent's rule proposed in [40].

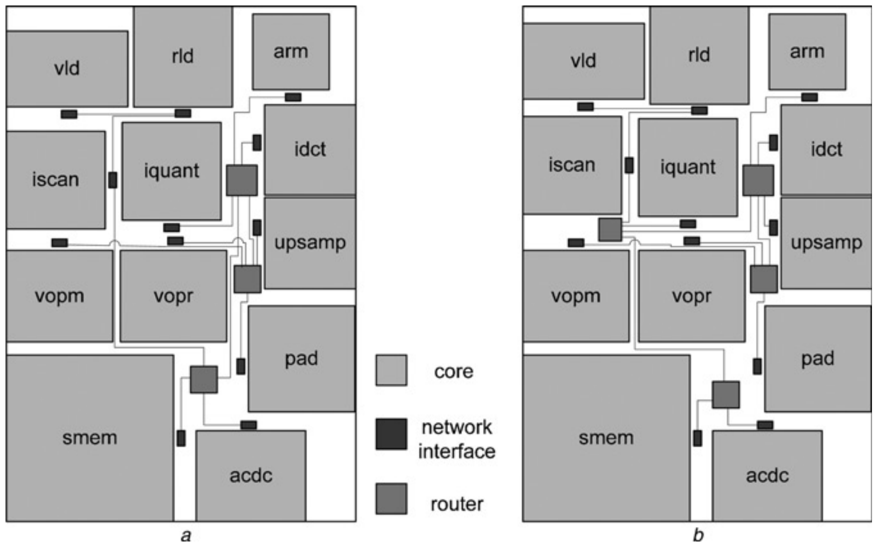

Figure 5 *VOPD custom topology floorplans synthesised by different algorithms*

a Topology by RRRM
b Topology by CLUSTER and DECOMPOSE

They showed that the traffic distribution models of NoC applications should follow a similar Rent's rule distribution as in conventional VLSI netlists. The bandwidth version of Rent's rule was derived showing that the relationship between the external bandwidth B across a boundary and the number of blocks G within a boundary obeys $B = kG^{\beta}$, where k is the average bandwidth for each block and β is Rent's exponent. The benchmark generation procedure proposed in [41] is adopted and modified in accordance with NoC-centric Rent's rule to generate multicast benchmarks. The average bandwidth k for each block and Rent's exponent β are specified by the user. In our experiments, we generated large NoC benchmarks by varying k ranging from 100 to 500 kb/s and varying β from 0.65 to 0.75. We formed multicast traffic with varying group sizes for about 10% of the flows. Thus, our multicast benchmarks cover a large range of applications

Table 4 NoC power and execution time results

Appli.	Label	|V|	|E|	RRRM		Cluster				Decompose			
				Power (W)	Time (s)	Power (W)	Ratio/ RRRM	Time (s)	Ratio/ RRRM	Power (W)	Ratio/ RRRM	Time (s)	Ratio/ RRRM
MMS	B1	25	33	0.138	0.01	0.123	0.89	32	3207	0.132	0.96	0.97	97
V + M	B2	24	27	0.091	0.01	0.086	0.94	15	1547	0.090	0.99	1.30	130
M + P	B3	20	21	0.043	0.01	0.053	1.25	7	745	0.053	1.25	0.62	62
V + M + M	B4	36	40	0.120	0.02	0.120	1.00	73	3651	0.124	1.03	1.00	50
4in1	B5	44	48	0.134	0.02	0.139	1.04	131	6525	0.142	1.06	0.94	47
M1	B6	16	32	0.150	0.03	0.191	1.27	31	1033	0.176	1.17	3	100
M2	B7	20	48	0.257	0.06	0.266	1.03	132	2200	0.291	1.13	7	117
M3	B8	25	58	0.305	0.18	0.347	1.14	235	1306	0.422	1.38	13	72
M4	B9	30	68	0.352	0.63	0.426	1.21	499	792	0.419	1.19	20	32
M5	B10	36	84	0.470	1.52	0.528	1.12	1430	941	0.605	1.29	39	26
M6	B11	42	100	0.534	3.07	0.597	1.12	3123	1017	0.774	1.45	68	22
M7	B12	49	122	0.732	5.82	0.763	1.04	5385	925	0.868	1.19	126	22
M8	B13	56	136	0.876	15.00	0.930	1.06	8808	587	0.960	1.10	202	13
M9	B14	64	164	0.924	35.00	1.052	1.14	18576	531	1.131	1.22	344	10

doi: 10.1049/iet-cdt.2008.0095

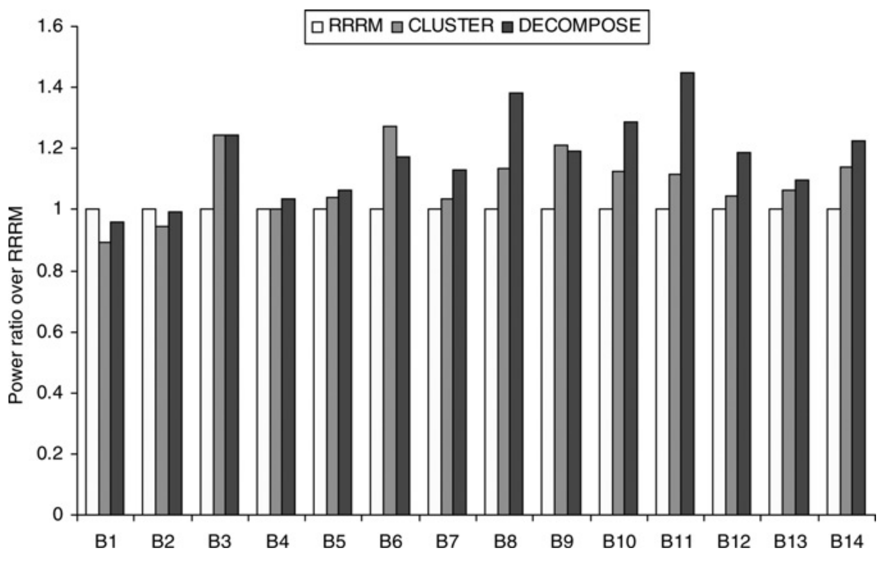

Figure 6 *NoC power comparisons relative to RRRM*

with mixed unicast/multicast flows and varying hop count and data rate distributions.

All experimental results were obtained on a 1.5 GHz Intel P4 processor machine with 512 MB of memory running Linux.

8.2 Comparison of results

The floorplans for the custom topologies synthesised by our tool using different algorithms for one of the benchmark VOPDs are shown in Fig. 5. Fig. 5*a* shows the topology generated by RRRM. It consists of three routers with 0.040 W power consumption. Fig. 5*b* shows the topology generated by CLUSTER and DECOMPOSE. These two algorithms generated the same topology for VOPD consisting of four routers, each having a smaller size. Its total power consumption is 0.042 W. Although the topology generated by RRRM has larger routers, it benefits from reducing one router, leading to lower power for the overall network.

The synthesis results of our algorithm RRRM on all benchmarks at 70 nm in comparison to the results using CLUSTER and DECOMPOSE are shown in Table 4. For all benchmarks, the power results and the execution times of each algorithm, and power ratios and execution time ratios of CLUSTER and DECOMPOSE over RRRM are reported. The power results of all algorithms relative to RRRM are graphically compared in Fig. 6. The results show that RRRM can efficiently synthesise NoC architectures that minimise power consumption as well as achieve good performance. Among all the 14 benchmarks tested, RRRM can achieve better results than CLUSTER and DECOMPOSE for 12 benchmarks. On average,

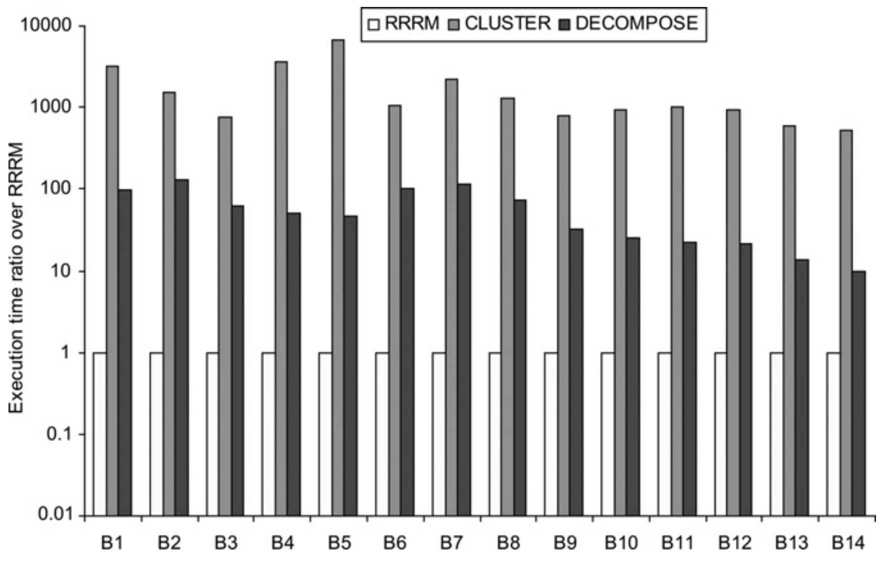

Figure 7 *Execution time comparisons relative to RRRM*

Table 5 NoC HOP count results

Application	Label	RRRM	Cluster		Decompose	
		Avg. Hops	Avg. Hops	Ratio/RRRM	Avg. Hops	Ratio/RRRM
MMS	B1	1.15	1.21	1.05	0.97	0.84
V + M	B2	1.07	1.04	0.97	1.30	1.21
M + P	B3	0.52	0.62	1.19	0.62	1.19
V + M + M	B4	0.90	0.93	1.03	1.00	1.11
4in1	B5	0.83	0.90	1.08	0.94	1.13
M1	B6	1.59	1.52	0.96	1.80	1.13
M2	B7	1.94	1.94	1.00	2.01	1.04
M3	B8	2.36	2.34	0.99	2.30	0.97
M4	B9	1.47	1.46	0.99	1.90	1.29
M5	B10	2.63	2.18	0.83	2.03	0.77
M6	B11	2.54	2.74	1.08	2.21	0.87
M7	B12	2.43	2.51	1.03	2.64	1.09
M8	B13	2.30	2.47	1.07	2.66	1.16
M9	B14	2.66	2.95	1.11	2.93	1.10

RRRM can achieve a 9% reduction in power consumption over CLUSTER and a 17% reduction in power consumption over DECOMPOSE.

Moreover, due to the low complexity of RRRM, it works much faster and more efficiently than CLUSTER and DECOMPOSE. The execution times of all algorithms relative to RRRM are graphically compared in Fig. 7. As can be seen from the results, RRRM can obtain results for all benchmarks under 1 min. Even for the largest benchmarks tested with 64 cores and 164 flows, RRRM can finish within 35 s whereas it takes CLUSTER over 5 h to finish. On average, RRRM is 1786 times faster than CLUSTER and 57 times faster than DECOMPOSE. Its low complexity and very short execution time make RRRM more suitable and efficient for benchmarks with large sizes.

To evaluate the performance of synthesised topologies, the average hop count results for benchmarks from the synthesised topology are reported in Table 5 and the results of all algorithms relative to RRRM are graphically compared in Fig. 8. Hop counts correspond to the number

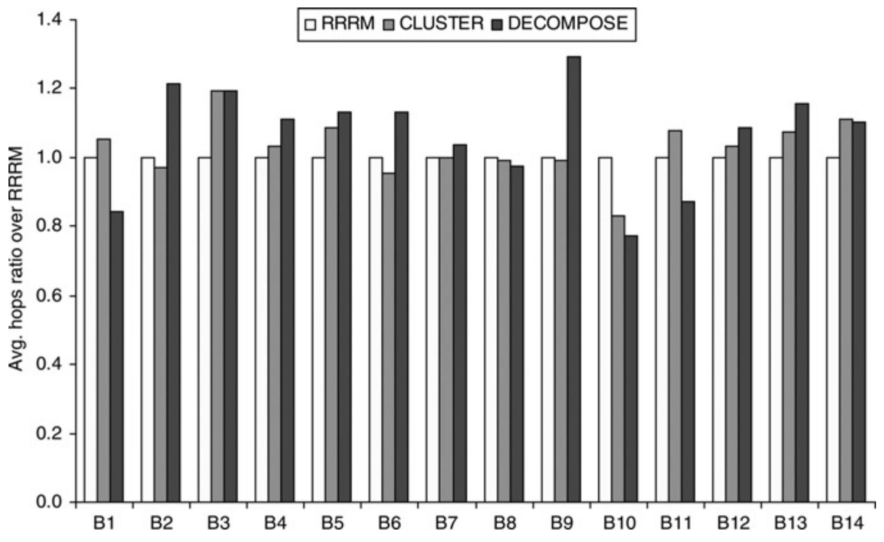

Figure 8 *NoC hop count comparisons relative to RRRM*

Table 6 NoC router area results

Application	Label	RRRM	Cluster		Decompose	
		Area(mm^2)	Area (mm^2)	Ratio/RRRM	Area (mm^2)	Ratio/RRRM
MMS	B1	0.98	0.95	0.97	0.92	0.97
V + M	B2	0.56	0.54	0.98	0.53	0.98
M + P	B3	0.22	0.31	1.39	0.31	1.00
V + M + M	B4	0.72	0.75	1.04	0.75	1.00
4in1	B5	0.78	0.86	1.10	0.86	1.00
M1	B6	0.90	1.14	1.27	1.20	1.06
M2	B7	1.56	1.59	1.02	1.61	1.01
M3	B8	1.78	2.03	1.14	2.03	1.00
M4	B9	1.93	2.68	1.39	2.41	0.90
M5	B10	2.68	3.06	1.14	3.01	0.98
M6	B11	2.93	3.56	1.21	3.36	0.94
M7	B12	3.95	3.92	0.99	4.24	1.08
M8	B13	4.90	4.81	0.98	5.19	1.08
M9	B14	5.05	5.54	1.10	5.61	1.01

of intermediate routers that a packet needs to pass through from the source to the destination. The results show that RRRM can improve the performance of synthesised topologies as well. In particular, the solutions obtained using RRRM can, on average, achieve a 3% reduction in average hop counts over CLUSTER and a 7% reduction in average hop counts over DECOMPOSE.

In a number of benchmarks, some modules have only single incoming flow or single outgoing flow. For example, for the VOPD application, six out of the 12 modules have at most one incoming flow as well as one outgoing flow, and 10 out of the 12 modules have either at most one outgoing flow or one incoming flow. For these benchmarks, the most efficient architectures are actually the ones that provide direct network links between network interfaces for some of its traffic flows without going through intermediate routers. (State-of-the-art router microarchitectures, such as those proposed in [42–44], employ finite buffers and virtual channels. Flow control is used to prevent upstream routers or network interfaces from sending more data when either buffer space or virtual

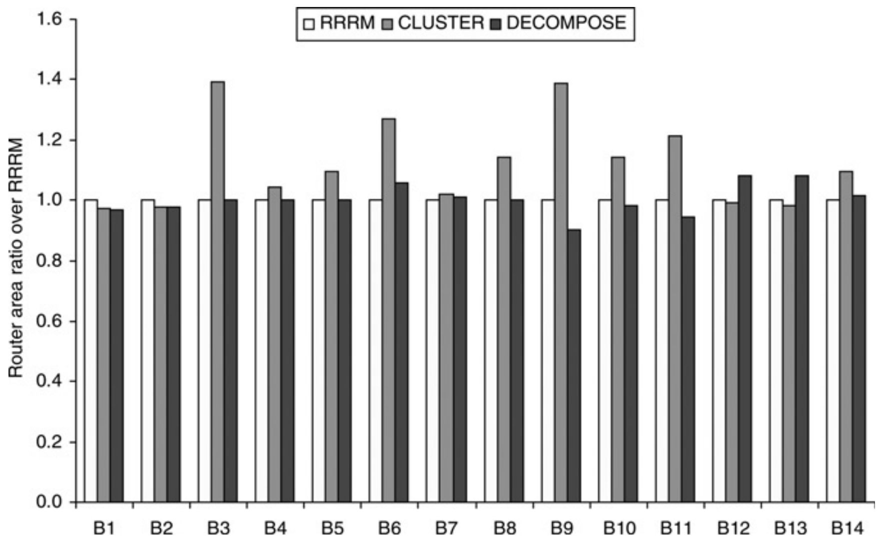

Figure 9 *NoC router area comparisons relative to RRRM*

channel is unavailable. Network interfaces that interoperate with these router microarchitectures must also correspondingly support the same flow control mechanism. This flow control mechanism can be used to control data transfers between network interfaces that are directly connected by a network link. Our synthesis algorithms can also be constrained to produce architectures where flows are required to pass through at least one router.) For these benchmarks, the average hop count may be less than one since not all flows necessarily pass through intermediate routers. Our algorithms are able to arrive at these implementations by correctly capturing these properties of the benchmarks.

Finally, to evaluate the area costs of the synthesised solutions, we also used Orion [6, 32] to estimate the areas of the routers in the synthesised architectures, using the same 70 nm technology used for power estimation. The area cost of a solution corresponds to the sum of the router areas in the solution. The results are presented in Table 6 and their relative results over RRRM are compared in Fig. 9. The total area costs of all solutions produced by RRRM are better than those produced by CLUSTER and DECOMPOSE. In particular, on average, the total area costs produced by RRRM are 12% better than those of CLUSTER and 1% better than those of DECOMPOSE.

9 Conclusions

In this paper, we proposed a very efficient algorithm called RRRM for the custom NoC synthesis problem. Our algorithm takes into consideration both unicast and multicast traffic and our objective is to construct an optimised interconnection architecture such that the communication requirements are satisfied and the power consumption is minimised.

The entire process is formulated as the joint multicast routing and network design problem using a rip-up and reroute procedure. Each multicast routing step is formulated as a minimum directed spanning tree problem. Our new formulation adopts a rip-up and reroute concept, which has been successfully used in the VLSI routing problem, as a good optimisation strategy to identify increasingly improving solutions. The minimum directed spanning tree formulation efficiently captures the best routing solutions for multicast flows during the topology synthesis procedure. We have described several ways to ensure deadlock-free routing of both unicast and multicast flows. Experimental results on a variety of benchmarks using a power consumption cost model show that our algorithm can produce more effective solutions with much faster execution times compared to our previous proposed algorithms CLUSTER and DECOMPOSE on both unicast and multicast applications. Therefore it also significantly outperforms regular mesh and optimised mesh topologies.

10 References

[1] DALLY W.J., TOWLES B.: 'Route packet, not wires: on-chip interconnection networks', *DAC*, 2001

[2] BENINI L., DE MICHELI G.: 'Networks on chips: a new SoC paradigm', *IEEE Comput.*, 2002, **35**, (1), pp. 70–78

[3] TAYLOR M.B., KIM J., MILLER J., *ET AL.*: 'The RAW microprocessor: a computational fabric for software circuits and general-purpose programs', *IEEE Micro*, 2002, **22**, (6), pp. 25–35

[4] SANKARALINGAM K., NAGARAJAN R., LIU H., *ET AL.*: 'Exploiting ILP, TLP, and DLP with the polymorphous TRIPS architecture', *ISCA*, 2003

[5] GROVE A.S.: 'Changing vectors of Moore's law', *International Electron Device Meeting*, December 2002 (Keynote presentation)

[6] CHEN X., PEH L.-S.: 'Leakage power modeling and optimization in interconnection networks', *ISPLED*, 2003

[7] YAN S., LIN B.: 'Application-specific network-on-chip architecture synthesis based on set partitions and Steiner trees', *ASPDAC*, 2008

[8] YAN S., LIN B.: 'Custom networks-on-chip architectures with multicast routing', *IEEE Trans. VLSI Syst.*, 2009, **17**, (3), pp. 342–355

[9] DEES W.A. JR., KARGER P.G.: 'Automated rip-up and reroute techniques', *DAC*, 1982

[10] SHIN H., SANGIOVANNI-VINCENTELLI A.L.: 'A detailed router based on incremental routing modifications: mighty', *IEEE Trans. CAD Integr. Circuits Syst.*, 1987, **6**, (6), pp. 942–955

[11] SHIROTA H., SHIBATANI S., TERAI M.: 'A new rip-up and reroute algorithm for very large scale gate arrays', *ICICC*, 1996

[12] CHU Y.J., LIU T.H.: 'On the shortest arborescence of a directed graph', *Sci. Sin.*, 1965, **14**, pp. 1396–1400

[13] EDMONDS J.: 'Optimum branchings', *Res.Natl Bur. Stand.*, 1967, **71B**, pp. 233–240

[14] HU J., MARCULESCU R.: 'Energy-aware mapping for tile-based NoC architectures under performance constraints', *ASP-DAC*, 2003

[15] MURALI S., DE MICHELI G.: 'Bandwidth constrained mapping of cores onto NoC architectures', *DATE*, 2004

[16] OGRAS U., MARCULESCU R.: 'Energy and performance driven NoC communication architecture synthesis using a decomposition approach', *DATE*, 2005

[17] OGRAS U., MARCULESCU R.: 'Application specific network-on-chip architecture customization via long range link insertion', *ICCAD*, 2005

[18] PINTO A., CARLONI L.P., SANGIOVANNI-VINCENTELLI A.L.: 'Efficient synthesis of networks on chip', *ICCD*, 2003

[19] SRINIVASAN K., CHATHA K.S., KONJEVOD G.: 'Linear-programming-based techniques for synthesis of network-on-chip architectures', *IEEE Trans. VLSI Syst.*, 2006, **14**, (4), pp. 407–420

[20] SRINIVASAN K., CHATHA K.S., KONJEVOD G.: 'Application specific Network-on-Chip design with guaranteed quality approximation algorithms', *ASPDAC*, 2007

[21] MURALI S., MELONI P., ANGIOLINI F., *ET AL.*: 'Designing application-specific networks on chips with floorplan information', *ICCAD*, 2006

[22] LIN X., MCKINLEY P.K., NI L.M.: 'Deadlock-free multicast wormhole routing in 2-D mesh multicomputers', *IEEE Trans. Parallel Distrib. Syst.*, 1994, **5**, (8), pp. 793–804

[23] MALUMBRES M.P., DUATO J., TORRELLAS J.: 'An efficient implementation of tree-based multicast routing fordistributed shared-memory', *IEEE Symp. Parallel and Distributed Processing*, 1996, pp. 186–189

[24] GOOSSENS K., DIELISSEN J., RADULESCU A.: 'The thereal network on chip: Concepts, architectures, and implementations', *IEEE Des. Test Comput.*, 2005, **22**, (5), pp. 414–421

[25] MILLBERG M., NILSSON E., THID R., *ET AL.*: 'Guaranteed bandwidth using looped containers in temporally disjoint networks within the Nostrum network on chip', *DATE*, 2004

[26] LU Z., YIN B., JANTSCH A.: 'Connection-oriented multicasting in wormhole-switched networks on chip', *Emerging VLSI Technol. Archit. (ISVLSI)*, 2006, (2–3), p. 6

[27] SAMMAN F.A., HOLLSTEIN T., GLESNER M.: 'Multicast parallel pipeline router architecture for network-on-chip', *DATE 2008*, 2008

[28] CARARA E.A., MORAES F.G.: 'Deadlock-free multicast routing algorithm for wormhole-switched mesh networks-on-chip', *ISVLSI*, 2008

[29] SHERWANI N.A.: 'Algorithms for VLSI physical design automation' (Kluwer Academic Publishers, Norwell, MA, 1998, 3rd edn.)

[30] HONG X., HUANG G., CAI Y., *ET AL.*: 'Corner block list: an effective and efficient topological representation of non-slicing floorplan', *ICCAD*, 2000

[31] ADYA S.N., MARKOV I.L.: 'Fixed-outline floorplanning: Enabling hierarchical design', *IEEE Trans. VLSI Syst.*, 2003, **11**, (6), pp. 1120–1135

[32] WANG H., *ET AL.*: 'Orion: a power-performance simulator for interconnection networks', *MICRO 35*, 2002

[33] CHEN G., FRIEDMAN E.G.: 'Low-power repeaters driving RC and RLC interconnects with delay and bandwidth constraints', *IEEE Trans. VLSI Syst.*, 2006, **14**, (2), pp. 161–172

[34] ZHANG L., CHEN H., YAO B., *ET AL.*: 'Repeated on-chip interconnect analysis and evaluation of delay, power, and bandwidth metrics under different design goals', *ISQED*, 2007

[35] *The international Technology roadmap for Semiconductors*, 2007

[36] RIJPKEMA E., GOOSSENS K.G.W., RADULESCU A., *ET AL.*: 'Trade-offs in the design of a router with both guaranteed and best-effort services for networks on chip', *DATE*, 2003

[37] ENRIGHT-JERGER N., LIPASTI M., PEH L.-S.: 'Circuit-switched coherence', *IEEE Comput. Archit. Lett.*, 2007, **6**, (1), pp. 5–8

[38] DALLY W.J., SEITZ C.L.: 'Deadlock-free message routing in multiprocessor interconnection networks', *IEEE Trans. Comput.*, 1987, **C-36**, (5), pp. 547–553

[39] BERTOZZI D., JALABERT A., *ET AL.*: 'NoC synthesis flow for customized domain specific multiprocessor systems-on-chip', *IEEE Trans. Parallel Distrib. Syst.*, 2005, **16**, (2), pp. 113–129

[40] GREENFIELD D., BANERJEE A., LEE J.-G., *ET AL.*: 'Implications of Rent's rule for NoC design and its fault-tolerance', *NOCS 2007*, 2007

[41] STROOBANDT D., VERPLAETSE P., VAN CAMPENHOUT J.: 'Generating synthetic benchmark circuits for evaluating CAD tools', *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 2000, **19**, (9), pp. 1011–1022

[42] PEH L.-S., DALLY W.J.: 'A delay model and speculative architecture for pipelined routers'. Seventh Int. Symp. High-Performance Computer Architecture (HPCA), 2001, pp. 255–266

[43] WANG H., PEH L.-S., MALIK S.: 'Power-driven design of router microarchitectures in on-chip networks', *MICRO 36*, 2003

[44] MULLINS R.: 'Minimising dynamic power consumption in on-chip networks'. Int. Symp. System-on-Chip, 2006, pp. 1–4