POLITECNICO DI TORINO
Repository ISTITUZIONALE

Turbo NOC: a framework for the design of Network-on-Chip-basedturbo decoder architectures

(Article begins on next page)

21 May 2024

Turbo NOC: a framework for the design of Network On Chip based turbo decoder architectures

Maurizio Martina, *Member IEEE*, Guido Masera, *Senior Member IEEE*

Abstract—This work proposes a general framework for the design and simulation of network on chip based turbo decoder architectures. Several parameters in the design space are investigated, namely the network topology, the parallelism degree, the rate at which messages are sent by processing nodes over the network and the routing strategy. The main results of this analysis are: i) the most suited topologies to achieve high throughput with a limited complexity overhead are generalized de-Bruijn and generalized Kautz topologies; ii) depending on the throughput requirements different parallelism degrees, message injection rates and routing algorithms can be used to minimize the network area overhead.

Index Terms—Turbo Decoder, Network on Chip, VLSI

I. INTRODUCTION

In the last years wireless communication systems coped with the problem of delivering reliable information while granting high throughput. This problem has often been faced resorting to channel codes able to correct errors even at low signal to noise ratios. As pointed out in Table I in [1], several standards for wireless communications adopt binary or double binary turbo codes [2], [3] and exploit their excellent error correction capability. However, due to the high computational complexity required to decode turbo codes, optimized architectures (e.g. [4], [5], [6], [7], [8]) have been usually employed. Moreover, several works addressed the parallelization of turbo decoder architectures to achieve higher throughput [9], [10]. In particular, many works concentrate on avoiding, or reducing, the collision phenomenon that arises with parallel architectures (e.g. [11], [12], [13], [14]).

Although throughput and area have been the dominant metrics driving the optimization of turbo decoders, recently, the need for flexible systems able to support different operative modes, or even different standards, has changed the perspective. In particular, the so called software defined radio (SDR) paradigm made flexibility a fundamental property [15] of future receivers, which will be requested to support a wide range of heterogeneous standards. Some recent works (e.g. [1], [16], [17]) deal with the implementation of Application-Specific Instruction-set Processor (ASIP) architectures for turbo decoders. In order to obtain architectures that achieve both high throughput and flexibility multi-ASIP is an effective solution. Thus, together with flexible and high throughput processing elements, a multi-ASIP architecture must feature also a flexible and high throughput interconnection backbone. To that purpose, the Network-On-Chip (NOC) approach has

been proposed to interconnect processing elements in turbo decoder architectures designed to support multiple standards [18], [19], [20], [21], [22], [23]. Current approaches to multi-standard turbo decoders pragmatically aim at implementing a "just enough flexibility", by supporting interleavers and throughput requirements specified in some of the current standards. On the contrary, this paper addresses a "full flexibility", capable of supporting high throughput implementation of any interleaver. In fact, even if parallel collision-free turbo decoder architectures exist, it is difficult to obtain a flexible architecture that is able to manage in a collision-free manner different turbo codes. In fact, some codes lead to parallel collision-free architectures only for specific values of the parallelism degree. As an example with WiMax turbo codes the collision-free feature can be obtained with proper selection of the decoder parallelism degree as a function of the data-frame size [24]. As a consequence, given a parallelism degree and a set of turbo codes, the NOC approach is a suitable solution to support the required flexibility. In addition, NOC based turbo decoder architectures have the intrinsic feature of adaptively reducing the communication bandwidth by the inhibition of unnecessary extrinsic information exchange. This can be obtained by exploiting bit-level reliability-based criteria where unnecessary iterations for reliable bits are avoided [25].

In [18], [19], [20] ring, chordal ring and random graph topologies are investigated whereas in [21] previous works are extended to mesh and toroidal topologies. Furthermore, in [22] butterfly and Benes topologies are studied, and in [23] binary de-Bruijn topologies are considered. However, none of these works presents a unified framework to design a NOC based turbo decoder, showing possible complexity/performance trade-offs. This work aims at filling this gap by providing a comprehensive study of NOC based turbo decoders, conducted by means of a dedicated NOC simulator and a list of obtained results, showing the complexity/performance trade-offs offered by different topologies, routing algorithms, node and ASIP architectures. Moreover, from previous studies it is not completely clear if the area overhead of NOC based turbo decoders is acceptable. This work tries to further investigate this aspect by giving real post synthesis results for a 130 nm standard cell technology.

The paper is structured as follows: in section II the requirements and characteristics of a parallel turbo decoder architecture are analyzed, whereas in section III NOC based approach is introduced. Section IV summarizes the topologies considered in previous works and introduces generalized de-Bruijn and generalized Kautz topologies as promising solutions for NOC based turbo decoder architectures. In section V three main routing algorithms are introduced, whereas in

section VI the Turbo NOC framework is described. Section VII describes the architecture of the different routing algorithms considered in this work, section VIII presents the experimental results and section IX draws some conclusions.

II. System requirement analysis

A parallel turbo decoder can be modeled as P processing elements that need to read from and write to P memories. Each processing element, often referred to as soft-in-soft-out (SISO) module, performs the BCJR algorithm [26], whereas the memories are used for exchanging the extrinsic information λ among the SISOs. The decoding process is iterative and usually each SISO performs sequentially the BCJR algorithm for the two constituent codes used at the encoder side; for further details on the SISO module the reader can refer to [27]. As a consequence, each iteration is made of two half iterations referred to as interleaving and de-interleaving. During one half iteration the extrinsic information produced by SISO i at time j ($\lambda_{i,j}$) is sent to the memory k at the location t, where $k = k(i, j)$ and $t = t(i, j)$ are functions of i and j derived from the permutation law (Π or interleaver) employed at the encoder side. Thus, the time required to complete the decoding is directly related to the number of clock cycles necessary to complete a half iteration. Without loss of generality, we can express the number of cycles required to complete a half iteration (hi) as

$$N_{cyc}^{hi} = \frac{N}{P \cdot R} + IL \qquad (1)$$

where N is the total number of trellis steps in a data frame, N/P is the number of trellis steps processed by each SISO, R is the SISO output rate, namely the number of trellis steps processed by a SISO in a clock cycle, and IL is the interconnection structure latency. Thus, the decoder throughput expressed as the number of decoded bits over the time required to complete the decoding process is

$$T = \frac{d \cdot N \cdot f_{clk}}{2I \cdot N_{cyc}^{hi}} = \frac{d \cdot N \cdot f_{clk}}{2I \cdot \left(\frac{N}{P \cdot R} + IL\right)} \qquad (2)$$

where f_{clk} is the clock frequency, I is the number of iterations, $d = 1$ for binary codes and $d = 2$ for double binary codes. When the interconnection structure latency is negligible with respect to the number of cycles required by the SISO, we obtain

$$T \approx \frac{d \cdot P \cdot R}{2I} \cdot f_{clk} \qquad (3)$$

Thus, to achieve a target throughput $\hat{T}$ and satisfactory error rate performance, a proper number $\hat{I}$ of iterations should be used. The minimum P (P_m) to satisfy $\hat{T}$ with $\hat{I}$ iterations can be estimated from (3) for some ASIP architectures available in the literature. If we consider $\hat{I} = 5$, as in [1], [17], P ranges between 5 and 37 to achieve $\hat{T} = 200$ Mb/s (see Table I). It is worth pointing out that the $C = (R \cdot d)^{-1}$ values in Table I represent the average numbers of cycles required by the SISO to update the soft information of one bit (see Table VI in [1] and Table I in [17]). Moreover, C strongly depends on the internal architecture of the SISO and in general tends to increase with the code complexity. As a consequence, several

conditions can further increase P, namely 1) interconnection structures with larger IL; 2) higher $(R \cdot d)^{-1}$ values; 3) higher $\hat{T}$; 4) higher $\hat{I}$; 5) lower clock frequency. Thus, we consider as relevant for investigation a slightly wider range for P: $P \in \{8, 16, 32, 64\}$.

Table I
PARALLELISM DEGREE REQUIRED TO OBTAIN $\hat{T} = 200$ MB/S FOR $\hat{I} = 5$ WITH SOME ASIP ARCHITECTURES AVAILABLE IN THE LITERATURE

Architecture	Technology [nm]	f_{clk} [MHz]	$C = (R \cdot d)^{-1}$	P_m
[1]	65	400	2.35	6
[17]	90	400	1.75	5
[28]	90	335	6.5	20
[28]	180	180	6.5	37

III. Network based approach

The NOC approach [29] has been proposed as a general methodology to interconnect heterogeneous intellectual properties (IP) in complex systems on chip (inter-IP interconnection). Recent works deal with methodologies to design application specific NOCs (e.g. [30]) where the NOC is tailored around a particular application or group of applications. In this scenario, turbo decoder architectures are a common IP required in physical layer chips for modern communication standards. In this work, as in some previous papers, e.g. [18], [21], [23] we concentrate on the problem of interconnecting the main building blocks of a parallel turbo decoder, namely we focus on the intra-IP interconnection problem [31], and we do not deal with the general problem of connecting the turbo decoder IP to other receiver modules through an inter-IP interconnection network. To that purpose, it is worth pointing out that statistical characterization of communication patterns, which is one of the most relevant aspects in the design of application specific NOCs, is not required in turbo decoders, as communication patterns depend on Π. As a consequence, given a set turbo codes with the corresponding Π laws, the intra-IP communication patterns are deterministic. Thus, the challenge of NOC based turbo decoder architectures is to find one or more sets of parameters that match throughput constraints for all supported standards with a reduced complexity overhead. This set of parameters includes R, P, the topology and the routing algorithm. Two interesting results, developed for a generic application in [32], can be used as general guidelines to design NOC based turbo decoder architectures. i) For short messages and maximum workload ($R = 1$) NOC is an effective solution compared to bus based implementations when P grows. ii) NOC is still an effective solution compared to a bus based implementation even when $R < 1$, given that $P > 25$. Moreover, as highlighted in section II and in [17], to achieve a target throughput of several hundreds of Mb/s a realistic value for P is of some tens. Thus, as shown in [21] interconnection structures that permit concurrent interleaving are mandatory for this scenario, making NOC a viable solution.

A NOC based turbo decoder architecture relies on P nodes connected through a proper topology where the extrinsic information is sent over the network according to a certain
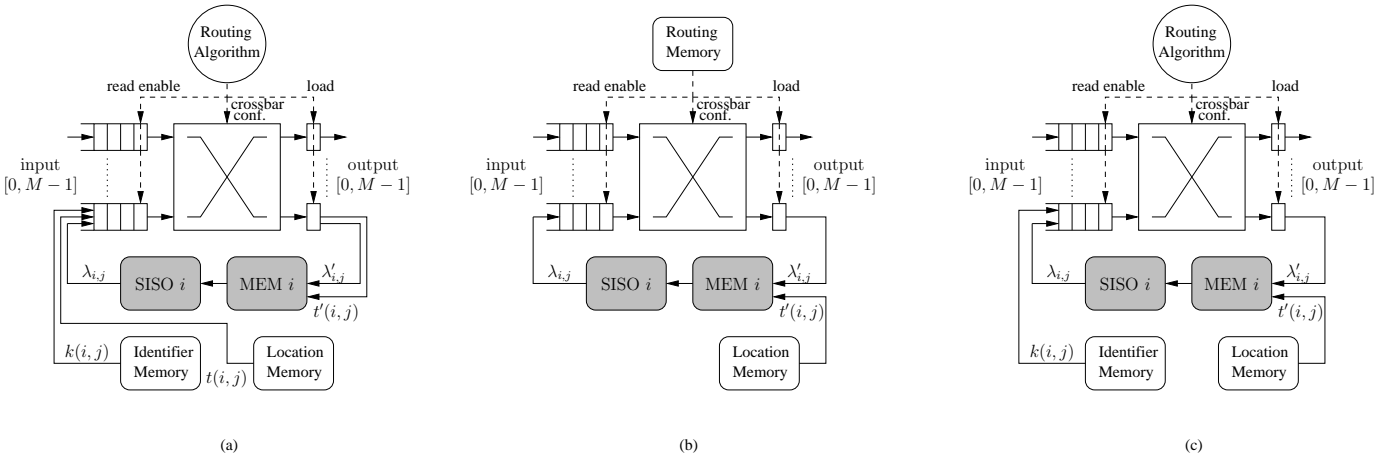
Figure 1. Node block scheme: (a) FA architecture, destination identifier and memory location are sent over the network; (b) AP architecture, routing algorithm is precalculated and stored in a routing memory; (c) PP architecture, hybrid solution

routing algorithm. We assume that each node has a certain number of input and output ports (M). As highlighted in [21], node architectures based on input-queuing lead to a fairly low complexity. As a consequence we assume that each node has a FIFO for each input, a crossbar to connect each input FIFO to a proper output and an output register, as shown in Fig. 1. Furthermore, each node has a local SISO (SISO i) that sends extrinsic information over the network through the $M - 1$ labeled input port and a local memory (MEM i) that receives extrinsic information from the network through the $M - 1$ labeled output port.

Three possible node architectures, shown in Fig. 1, have been considered in this work, namely the *Fully Adaptive node architecture* (FA) and two simplified versions, *All Precalculated node architecture* (AP) and *Partially Precalculated node architecture* (PP).

a) Fully adaptive node architecture: In each half iteration a SISO sends N/P messages where every message is made of a payload containing the extrinsic information $\lambda_{i,j}$ and the location of the memory where the extrinsic information will be written ($t(i, j)$), and a header containing the identifier of the destination node ($k(i, j)$). As a consequence, the node should contain a memory to store $k(i, j)$ (Identifier Memory), a memory to store $t(i, j)$ (Location Memory) and a routing algorithm to properly route messages through the network (see Fig. 1 (a)).

b) All-precalculated node architecture: Since the permutation law defined by the interleaver is known a-priori, the path followed by a message during an interleaving (or de-interleaving) half iteration can be precalculated and stored as a routing information into a routing memory for each node. This approach reduces the data width of FIFOs, crossbars and registers as neither $k(i, j)$ nor $t(i, j)$ are sent over the network. The location where received messages ($\lambda'_{i,j}$) will be stored ($t'(i, j)$) can be also precalculated and stored into a Location Memory (see Fig. 1 (b)).

c) Partially-precalculated node architecture: Since the routing memory foot-print can be relevant, a hybrid solution is obtained by precalculating and storing only $t'(i, j)$, whereas the routing is managed by a routing algorithm (see Fig. 1 (c)).

This solution does not require a Routing Memory and employs a smaller payload with respect to the FA node architecture. On the other hand, the FA node architecture directly supports adaptive bandwidth reduction techniques, whereas, neither the AP nor the PP node architectures do. As an example the FA node architecture can be employed to implement the bit-level reliability-based criterion proposed in [25], where unnecessary iterations for reliable bits are avoided. In such a case, the extrinsic information related to reliable bits is not sent on the network leading to bandwidth reduction. As a consequence, this bandwidth reduction leads to increase the throughput of the NOC based turbo decoder.

IV. TOPOLOGIES

As highlighted in [21] and [23] the choice of topologies and routing algorithms impacts both on throughput and complexity. As a consequence, given a certain parallelism degree P, topologies with a small node degree ($D = M - 1$) as rings ($D = 2$) keep the network complexity overhead limited. On the other hand, topologies with a higher node degree as toroidal meshes ($D = 4$) can increase the throughput. Since interleavers tend to spread the extrinsic information almost uniformly among the P memories, we consider fixed degree topologies where every node has the same degree D. Among fixed degree topologies we included rings and toroidal mesh networks as in [18] and [21]. Moreover, de-Bruijn topologies are known to show shorter worst case distance among nodes than toroidal meshes: therefore, they are good candidates to reduce the latency of the network in turbo decoder architectures, as investigated in [23]. A de-Bruijn topology is made of nodes labeled by an array of n elements, each element is taken from an alphabet $\mathcal{A}$ with m symbols. Each node is connected to the nodes whose labels are obtained by left-shifting the node-label array and by placing in the rightmost position a symbol from $\mathcal{A}$. As a consequence, each node is connected to m nodes ($D = m$) and the number of nodes in the network is $P = m^n$. Thus, in general, de-Bruijn topologies for given P and D values not always exist. This limitation can be overcome by using generalized de-Bruijn topologies [33],
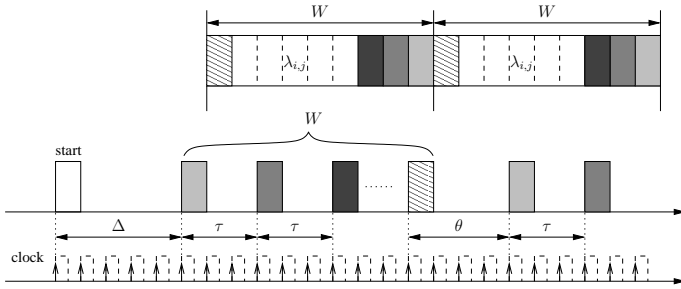
Figure 3. SISO architecture parameters: graphical representation of the timing for a generic SISO architecture that sends the extrinsic information according to the backward recursion order (Recursion order: BRO)

where there is a connection from node i to node j if

$$j = (i \cdot D + k) \bmod P \quad k = 0, 1, \ldots, D - 1 \qquad (4)$$

A further limitation of de-Bruijn and generalized de-Bruijn topologies are self loops that are present in some nodes, e.g. node 0 ("0000") in Fig. 2.

This limitation is overcome by Kautz topologies where nodes are labeled as in de-Bruijn topologies but avoiding sequences with equal symbols in consecutive positions of the node-label array (Kautz sequences). Then, node connections are obtained as for de-Bruijn topologies, where the symbol placed in the rightmost position of the node-label array is taken from $\mathcal{A}$, subject to the constraint that the obtained node-label array is a Kautz sequence. As a consequence, each node is connected to $m - 1$ nodes ($D = m - 1$) and the number of nodes in the network is $P = m \cdot (m - 1)^{n-1}$. Thus, as for de-Bruijn topologies, Kautz topologies for assigned P and D values not always exist. This problem is eliminated by using generalized Kautz topologies [34], where there is a connection from node i to node j if

$$j = (-i \cdot D - k) \bmod P \quad k = 1, \ldots, D \qquad (5)$$

Unfortunately, some generalized Kautz topologies have self loops: as an example with $P = 16$ and $D = 2$ node 10 ("1010") has a self loop (see Fig. 2). However, said $n_{sl}^x(P, D)$ the number of self loops introduced by a generalized de-Bruijn (B) or generalized Kautz (K) topology, we have that $n_{sl}^K(P, D) \leq n_{sl}^B(P, D)$ (see Fig. 2). As will be discussed in section V self loops can be managed in different ways depending on the routing algorithm.

Moreover, we included in our investigation honeycomb networks that, as suggested in [35], are alternatives to toroidal meshes that reduce nodes degree to $D = 3$. Thus, we have that rings, honeycombs and toroidal meshes are represented as undirected graphs, whereas de-Bruijn (generalized de-Bruijn) and Kautz (generalized Kautz) correspond to directed graphs. As an example in Fig. 2 we show the aforementioned topologies for $P = 16$.

V. Routing Algorithms

Since in turbo decoder architectures the achieved throughput is a key objective, we should try to deliver messages following the shortest available path. Furthermore the NOC must guarantee that all messages are delivered to the destination, namely

dropping of messages to avoid dead-locks is not allowed as it could impair the decoder correction capability. As highlighted in [23] shortest-path based routing algorithms are suited to achieve high throughput and grant message delivery. In the following we will consider both single-shortest-path (SSP) and all-local-shortest-path (ASP) based routing algorithms. In SSP algorithms only one shortest-path from each node i to each node k is considered. On the other hand, ASP algorithms rely on the fact that in a topology two nodes i and k may be connected by more shortest-paths. At each node i, the actual routing choice toward node k must be made by selecting one destination node directly connected to i and belonging to a set $\mathcal{N}^{i,k}$ defined as the set of all nodes adjacent to i and placed on a shortest path between i and k. As an example, in the toroidal mesh topology in Fig. 2 there are six distinct shortest paths that connect node 0 ("0000") to node 6 ("0110"). Two of these paths go through node 4 ("0100"), two of them through node 1 ("0001") and two through node 3 ("0011"). For the connection between nodes 0 and 6, SSP algorithm selects a unique path and the choice of the first intermediate connection among nodes 1, 3 and 4 depends the graph visiting order. On the contrary, ASP algorithm explores all shortest paths between 0 and 6 and this leads to consider all three connections from node 0 to nodes 1, 3 and 4 ($\mathcal{N}^{0,6} = \{1, 3, 4\}$), see bold lines in Fig. 2.

A further choice in the design of the NOC is how input FIFOs have to be served. Three distinct strategies have been considered in this work and indicated as SSP Round-Robin (SSP-RR), SSP FIFO-length (SSP-FL) and ASP FIFO-length with traffic-spreading (ASP-FT). These alternatives are briefly described in the following, while further details on routing algorithms for NOC can be found in [36]. The SSP-RR approach is based on a circular serving policy coupled to the SSP approach. FIFO-length strategies are weighted RR with weights equal to the number of elements stored into each input FIFO. The SSP-FL approach serves the input FIFOs based on the number of elements contained in each input FIFO: the longest FIFO is served first and the shortest one is served last. The ASP-FT approach is based on the input FIFO length serving policy, as for SSP-FL, but it is more complex and can be described as follows. Let's define $\mathcal{I}_j^{i,l}$ as the set of input ports in a node $l \in \mathcal{N}^{i,k}$ that can receive a message from node i at time j. At time j the number of elements contained in the input FIFO associated to port $p \in \mathcal{I}_j^{i,l}$ with $l \in \mathcal{N}^{i,k}$ is $L_{j,p}^l$. According to Algorithm 1, the ASP-FT routing algorithm chooses $\hat{l} \in \mathcal{N}^{i,k}$ and $\hat{p} \in \mathcal{I}_j^{i,l}$ so that

$$L_{j,\hat{p}}^{\hat{l}} = L_{min} = \min_{p,l} \{L_{j,p}^l\} \qquad (6)$$

The couples $\hat{l}$, $\hat{p}$ that satisfy (6) belong to the set $\mathcal{J}_{j,\hat{p}}^{i,\hat{l}}$. To choose only one couple in $\mathcal{J}_{j,\hat{p}}^{i,\hat{l}}$ we operate a traffic spreading based selection, namely our objective is to spread the traffic as much as possible over the network. To that purpose we use a set of counters (Q), where each counter $Q_{j,\hat{p}}^{i,\hat{l}}$ is incremented each time a message is sent from node i to node $\hat{l}$ through input port $\hat{p}$. Then, we select the couple $\tilde{l}$, $\tilde{p} \in \mathcal{J}_{j,\hat{p}}^{i,\hat{l}}$ that is
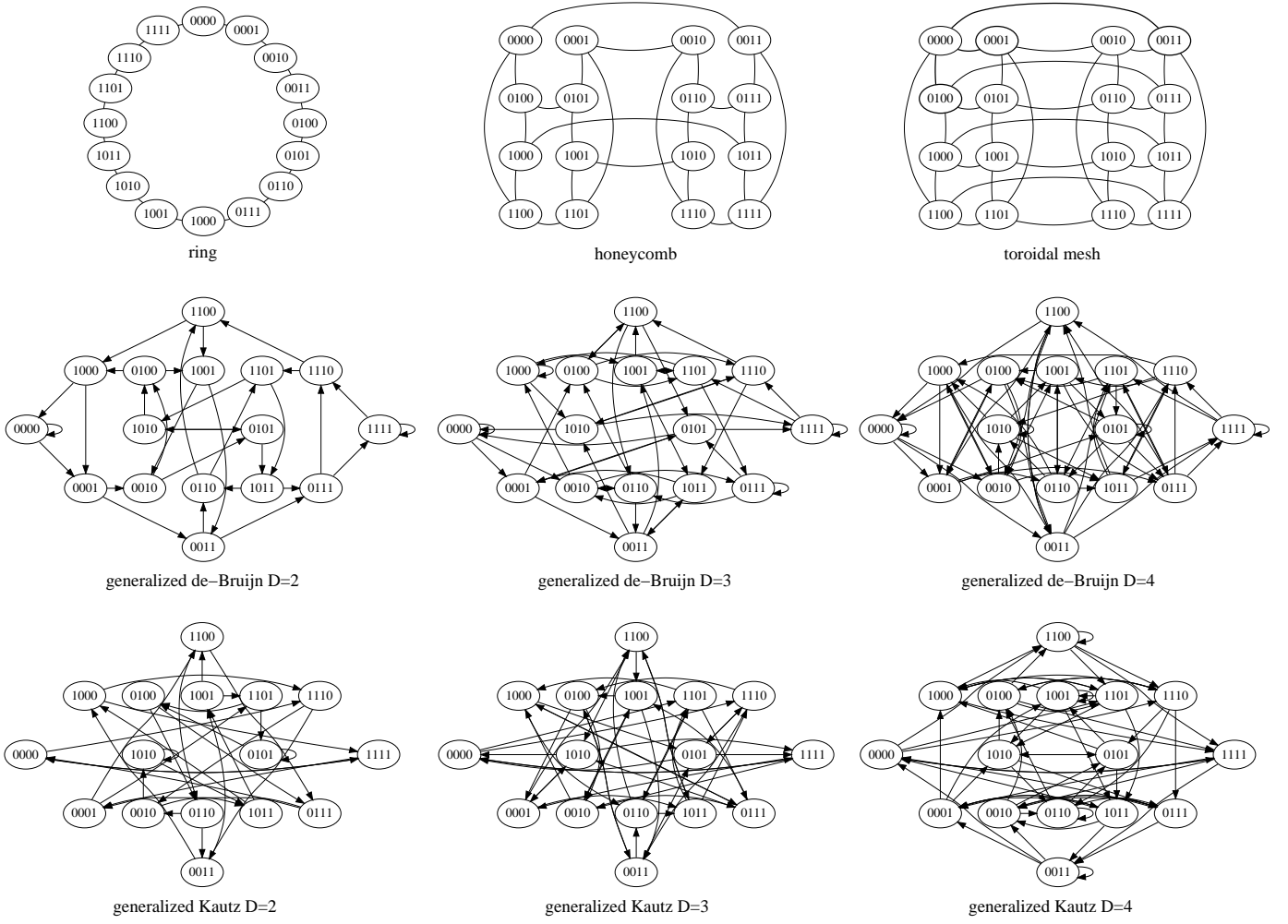
Figure 2. Example of the considered topologies for $P = 16$, bold lines refer to the example detailed in section V

associated to the least used path

$$Q_{j,\tilde{p}}^{i,\tilde{l}} = Q_{min} = \min_{\hat{p},\hat{l}}\{Q_{j,\hat{p}}^{i,\hat{l}}\} \qquad (7)$$

It is worth pointing out that, shortest-path based routing algorithms do not prevent output ports contention, that is a situation where two or more inputs need to send data to the same output port. Said $\mathcal{I}_{j,b}^n$ the set of inputs in node n that at time j need to send data to output port b, the contention problem can be faced by properly choosing an input $a \in \mathcal{I}_{j,b}^n$ allowed to send its data to output port b. The remaining inputs belonging to $\mathcal{I}_{j,b}^n - \{a\}$ can be managed in different ways. In this work we consider the following two approaches: i) storing $a' \in \mathcal{I}_{j,b}^n - \{a\}$ into the corresponding input FIFO so that we delay a colliding message, in the following we will refer to this approach as delay-colliding-message (DCM); ii) if possible, sending $a' \in \mathcal{I}_{j,b}^n - \{a\}$ to another output port $b' \neq b$, send-colliding-message (SCM). The DCM approach aims at reducing the number of hops to deliver a message to its destination, whereas the SCM approach aims at reducing the maximum depth of the input FIFOs. It is worth pointing out that topologies with self loops cause a waste of resources with DCM. In fact, since we are considering dead-lock free

---

Algorithm 1 ASP-FT routing algorithm

Require: $Q_{j,p}^{i,l} \leftarrow Q_{j-1,p}^{i,l}$ and $Q_{0,p}^{i,l} \leftarrow 0$
1: $L_{min} \leftarrow \infty$
2: $Q_{min} \leftarrow \infty$
3: **for all** $l \in \mathcal{N}^{i,k}$ **do**
4:     build $\mathcal{I}_j^{i,l}$
5:     **for all** $p \in \mathcal{I}_j^{i,l}$ **do**
6:        get $L_{j,p}^l$
7:        **if** $L_{j,p}^l \leq L_{min}$ **then**
8:           **if** $Q_{j,p}^{i,l} < Q_{min}$ **then**
9:              $Q_{min} \leftarrow Q_{j,p}^{i,l}$
10:            $L_{min} \leftarrow L_{j,p}^l$
11:            $\tilde{l} \leftarrow l$
12:            $\tilde{p} \leftarrow p$
13:          **end if**
14:        **end if**
15:     **end for**
16: **end for**
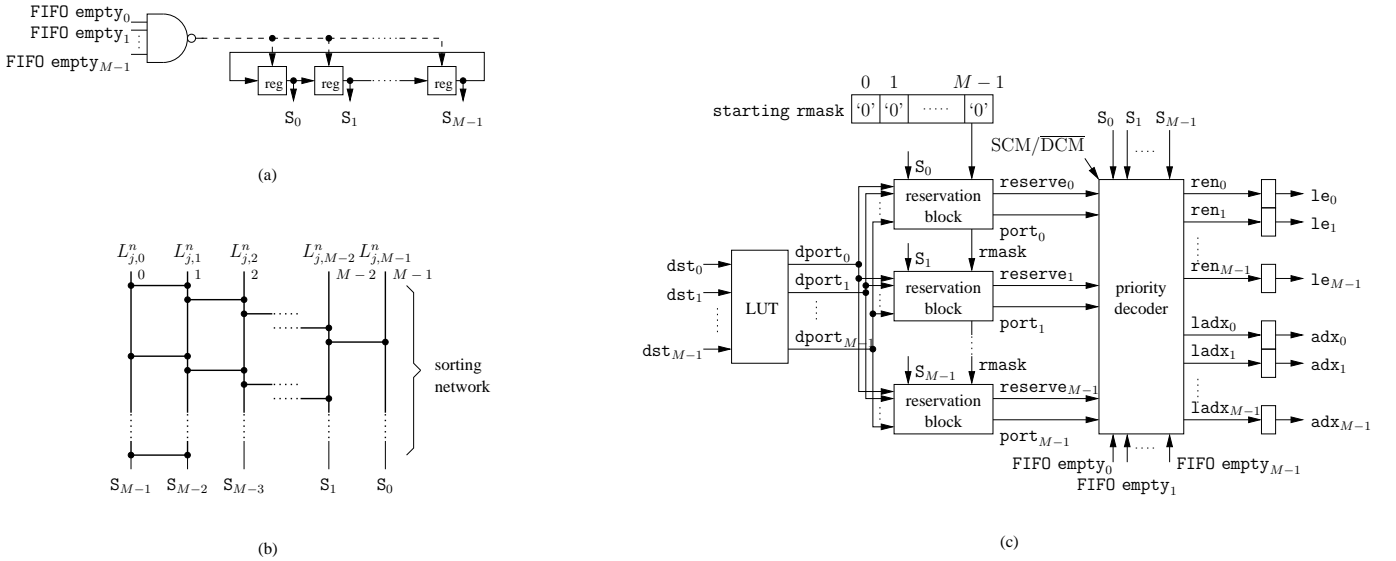17: $Q_{j,\tilde{p}}^{i,\tilde{l}} = Q_{j,\tilde{p}}^{i,\tilde{l}} + 1$

Figure 4. Routing algorithm architecture: (a) RR block scheme, (b) FL block scheme, (c) SSP block scheme

routing algorithms, DCM does not use a self loop as it is not on a shortest path. On the other hand, SCM can use a self loop, as a possible destination for a colliding message. However, by sending a message to a self loop we reduce the number of elements in one of the input FIFOs but we increase the number of elements in the input FIFO of the self loop.

VI. TURBO NOC SIMULATOR

The Turbo NOC simulator [37] is a cycle accurate, SystemC [38] based NOC simulator, specifically tailored for turbo decoder architectures. It estimates the throughput and complexity of a parallel NOC based turbo decoder architecture. It requires as inputs the following elements: the topology description in the form of an adjacency matrix, the permutation law Π used at the encoder and represented as a sequence of integer values, the routing algorithm (SSP-RR, SSP-FL, ASP-FT), the selected approach to handle contention (DCM/SCM) and the description of the key SISO characteristics. An example of the SISO characteristics is given in Fig. 3, where the upper part represents a frame of extrinsic informations ($\lambda_{i,j}$) that is sliced in two windows, each of which contains W values. The bottom part of Fig. 3 describes the timing of the SISO output: when a start signal is asserted, the SISO outputs the first extrinsic information after a certain latency Δ, that depends on the forward and backward recursion scheduling [39], on the trellis initialization strategy [40] and on the parallelism level of the SISO architecture [41]. Then, depending on SISO internal scheduling, the $\lambda_{i,j}$ values are output in a certain order. In Fig. 3 we assume that $\lambda_{i,j}$ values are output in a last-in-first-out order (from light gray to black) according to the backward recursion order. The number of clock cycles between two consecutive $\lambda_{i,j}$ is τ. Similarly, the number of clock cycles between the last output λ of a window (line-filled cell) and the first λ of the successive window (light gray cell) is θ. Stemming from Fig. 3, the required parameters to describe the SISO architecture are:

1) the window size (W) [42],

2) the SISO latency (Δ) expressed in clock cycles;
3) the Recursion Order (RO) used to send extrinsic informations on the network, namely Forward Recursion Order (FRO) or Backward Recursion Order (BRO),
4) the number of clock cycles between two consecutive outputs λ within a window (τ),
5) the number of clock cycles between the last output λ of a window and the first λ of the successive window (θ).

The simulator acts in two phases, a static phase (instantiation and binding) and a dynamic phase (cycle accurate simulation). During the static phase, the topology description defines P, D and all possible paths from one node to the other. The simulator represents the topology as a graph and calculates all the local shortest paths repeating the Floyd-Warshall algorithm [43] on pruned versions of the graph until no more local paths exist between a source node i and its adjacent nodes; it then stores each result of the Floyd-Warshall algorithm as an array. If a SSP routing algorithm is selected, only the first shortest path array is employed, otherwise all the shortest paths are considered. Moreover, P nodes are instantiated and binded according to the assigned topology and each SISO memory is loaded with N/P messages, based on the assigned permutation. The actual decoding process executed by SISO elements is not included in the tool, which only simulates the exchange of extrinsic informations. However, the SISO architecture parameters are employed to initialize a set of counters that are used to send the extrinsic information over the network with the same timing as in the real SISO architecture. The node is described by means of a hardware-description-language-like (hdl-like) model. When the static phase is completed, the simulation starts resorting to the SystemC kernel simulator and performs a cycle accurate hdl-like simulation. The results provided by the Turbo NOC simulator can be divided in two categories: cycle by cycle results and global results. The cycle by cycle results are: i) for each node, the status of each FIFO, ii) for each node the FIFO read enable and the crossbar configuration signals, iii) for each SISO the
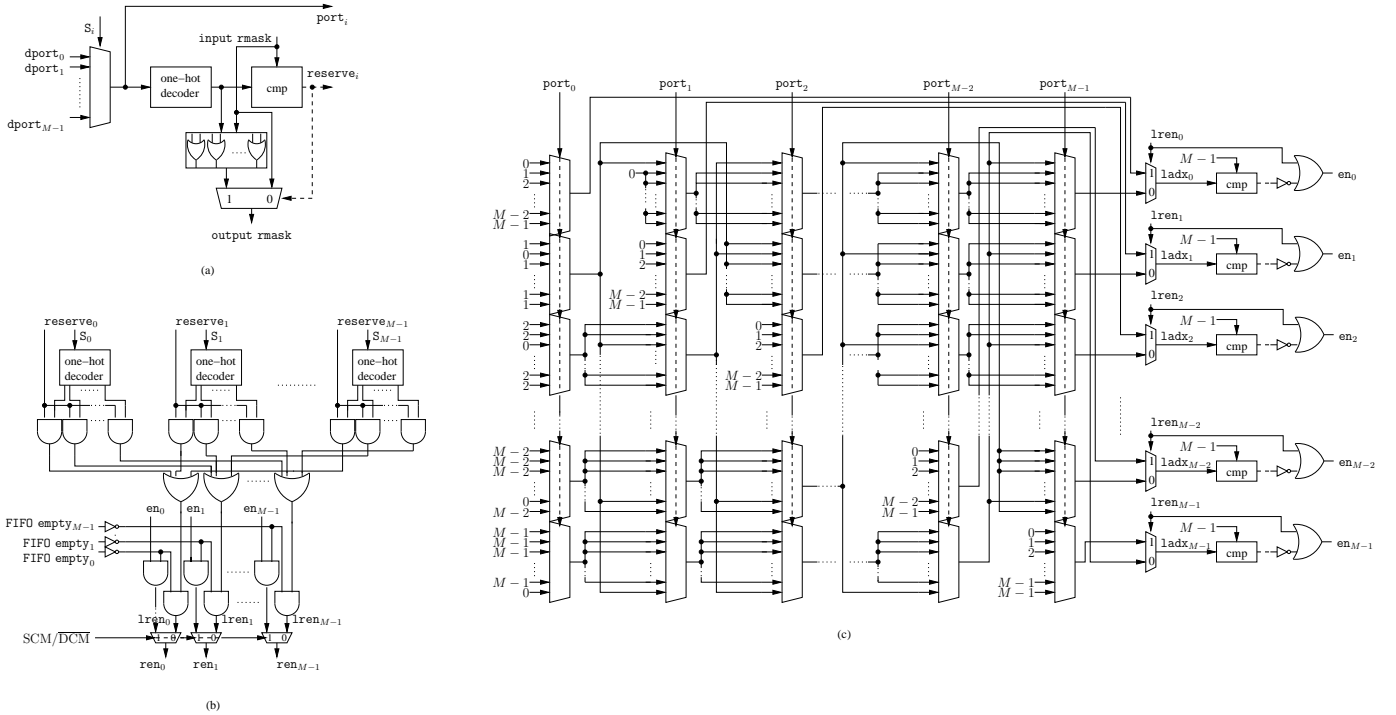
Figure 5.   Routing algorithm architecture details: (a) reservation block, (b) read-enable generation block, (c) destination-port generation block

$t'(i,j)$ sequence. The global results are: i) for each FIFO in each node the maximum FIFO size, ii) for each node the minimum, maximum and average latency (in clock cycles) of each received message and the total number of clock cycles to deliver all the messages. It is worth pointing out that, since self loops are not used with DCM, the maximum size of the corresponding input FIFOs is zero. In this case the tool automatically reduces the degree of those nodes with the effect of saving complexity.

VII. ROUTING ALGORITHM ARCHITECTURE

In order to keep the NOC complexity as small as possible, SSP-RR and SSP-FL routing algorithms have been implemented with FA and PP node architectures, whereas the ASP-FT algorithm has been implemented as a routing memory with the AP node architecture.

A. SSP-RR and SSP-FL architectures

The SSP-RR and SSP-FL architectures, thoroughly shown in Fig. 4 and 5, are made of two main parts. The first part sorts the input FIFOs based on the selected priority method (round-robin or FIFO length) and generates M signals, S_0, ..., S_{M-1}, where S_0 is the label of the input that is served first and S_{M-1} is the label of the input that is served last. The second part serves the input FIFOs according to the order specified by the S_0, ..., S_{M-1} sequence and generates the read-enable (ren_i) signals for the FIFOs, the load enable (le_i) signals for the output registers and the configuration commands for the crossbar (adx_i), where adx_i represents the label of the destination node specified by the first message in FIFO i. As shown in Fig. 4 (a), in the SSP-RR architecture a rotate register generates the S_i signals. On the other hand, (see Fig.

4 (b)), in the SSP-FL architecture the S_i signals are obtained with a sorting network. In node n the sorting network takes as an input the number of elements contained at time j in each input FIFO ($L_{j,p}^n$ with $p \in \{0, \ldots, M-1\}$) and outputs $S_0 = \arg\{\max_p\{L_{j,p}^n\}\}$, ..., $S_{M-1} = \arg\{\min_p\{L_{j,p}^n\}\}$. Both SSP-RR and SSP-FL architectures have been designed as parametric units to support different M values.

The generation of the ren_i, adx_i and le_i signals is enabled by the FIFO empty signals of the input FIFOs and requires the following units: a look-up-table (LUT), M reservation blocks and a priority decoder (Fig. 4 (c)). The LUT contains the shortest-path information.

In the SSP approach for each node i the $\mathcal{N}^{i,k}$ set contains only one node, and $\mathcal{I}_j^{i,l}$ contains only one port. There is only an output port on node i that connects node i with node k on a shortest path. Thus, every LUT contains P locations and the LUT in node i at location k contains the label of the output port to connect node i to node k. As a consequence, each LUT is a $P \times \lceil \log_2(M) \rceil$ table that converts M destinations (dst_i) to the corresponding ports (dport_i).

The reservation blocks update an M-position binary mask to avoid collisions on output ports, whereas the priority decoder implements the selected priority and FIFO management policies by properly generating the the ren_i and ladx_i signals. Since the le_i and adx_i signals must be asserted the clock cycle after the ren_i and ladx_i, they are delayed by means of registers. In particular, the le_i and adx_i are obtained by delaying the ren_i and ladx_i of one clock cycle.

1) Reservation block: Each reservation block (Fig. 5 (a)) receives the dport_i signals, according to the $S_0 \ldots S_{M-1}$ sequence, generates a reservation signal (reserve_i) and specifies the output port to be reserved (port_i). The reservation is obtained by updating the rmask, which contains a '1'

in the position of a reserved output port and a '0' in the position of a free output port. Each reservation block generates $\text{port}_i = \text{dport}_{S_i}$, that is converted by a one-hot decoder into a mask with a '1' in position port_i. The reservation mask is updated (`output rmask`) by comparing this mask with the `input rmask`: if the `input rmask` contains a '0' in position port_i the reserve_i goes to '1'.

2) Priority decoder: The priority decoder is made of two blocks: the read-enable generation block (Fig. 5 (b)) and the destination-port generation block (Fig. 5 (c)).

a) read-enable generation block: The read-enable generation block is based on few logic gates that act differently depeding on the approach selected to manage the input FIFOs (SCM/$\overline{\text{DCM}}$): i) in the DCM approach, $\text{ren}_i = \text{reserve}_{S_i}$ when FIFO i is not empty (FIFO empty_i='0') is obtained by combining S_i one-hot representation with the corresponding `reserve` signal. ii) in the SCM approach, $\text{ren}_i = \text{en}_i$ when FIFO i is not empty (FIFO empty_i='0') is based on en_i that is a set of M signals produced by the destination-port generation block, where en_i ='0' when lren_i ='0' and $\text{ladx}_i = M-1$, namely the output port with label $M-1$ is used only for messages whose destination is the node itself (Fig. 5 (c)).

b) destination-port generation block: This is an array of multiplexers, where each multiplexer in position i, i implements $\text{ladx}_i = \text{port}_i$ when lren_i ='1'. On the other hand, ladx_i must take the value of an un-reserved output port when lren_i ='0'. This is obtained by means of the permutation network implemented by the multiplexers in position j, i with $j \neq i$ whose outputs ($\text{mux}_{j,i}$) are

$$\text{mux}_{j,0} = \begin{cases} 0 & \text{if } \text{port}_0 = j \\ j & \text{otherwise} \end{cases} \tag{8}$$

and for $i > 0$

$$\text{mux}_{j,i} = \begin{cases} \text{mux}_{i,i-1} & \text{if } \text{port}_i = j \\ \text{mux}_{j,k} & \text{otherwise} \end{cases} \tag{9}$$

where

$$k = \begin{cases} i-2 & \text{if } j = i-1 \\ i-1 & \text{otherwise} \end{cases} \tag{10}$$

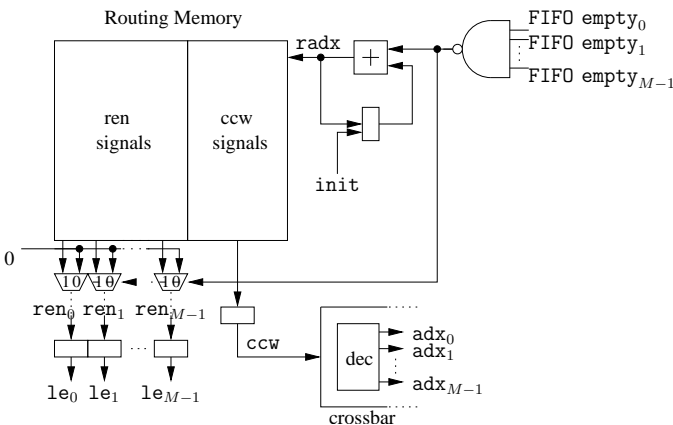and if $k < 0$, then $\text{mux}_{j,k} = 0$.

Figure 6. Routing memory architecture

B. ASP-FT architecture

The ASP-FT algorithm is simply implemented by means of a routing memory. As a consequence, DCM and SCM approaches are integrated by filling the routing memory with the appropriate configuration words. Each word is the concatenation of the ren_0, ...ren_{M-1} signals with the adx_0, ..., adx_{M-1} signals. In order to reduce the word width, the adx_0, ..., adx_{M-1} signals, which can be represented on $M \times \lceil \log_2(M) \rceil$ bits, are coded into a crossbar configuration word (`ccw`). Since for an M-port crossbar the possible configurations are $M!$, `ccw` is represented on $\lceil \log_2(M!) \rceil$ bits. The corresponding decoder is hardwired into the crossbar. Thus, as shown in Fig. 6 the main component in the routing memory architecture is a RAM. The RAM address (`radx`) is generated by an adder and a register. The adder is incremented when at least one input FIFO is not empty (FIFO empty_i ='0') and it is initialized to zero when the half iteration starts (`init`). Moreover, the ren_i are forced to '0' when FIFOs are empty.

C. Architecture implementation

To achieve high throughput the routing algorithm should be able to serve the input FIFOs in one clock cycle. This requirement, which is an intrinsic feature of the routing memory architecture used for the ASP-FT, implies that the architectures for the SSP-RR and SSP-FL routing algorithms are combinational circuits. As it can be inferred from Fig. 4 and 5 the speed of SSP-RR and SSP-FL architectures depends mainly on M, in fact, M impacts on the size of several parts of the routing algorithm architectures, namely the sorting network, the shortest-path information LUT, the reservation mask, the priority decoder and on the number of reservation blocks. Given the topologies presented in section IV, we described the SSP-RR and SSP-FL architectures as parametric blocks and we performed the logical synthesis on a 130 nm standard cell technology for $M \in \{3, 4, 5\}$. Post synthesis results confirm that a clock frequency of more than 200 MHz is achieved with a complexity that ranges from about 1000 μm^2 to about 6000 μm^2.

VIII. SIMULATIONS AND RESULTS

The Turbo NOC simulator has been used to simulate both interleaving and de-interleaving with four significant permutation laws, namely:

1) WiMax interleaver with N=2400 and W=38
2) UMTS interleaver with N=5114 and W=40
3) A prunable S-random interleaver [44] with N=16384 and W=37
4) A circular shifting interleaver [45] with N=24576 and W=39

We tested the following topologies:

1) ring (R)
2) toroidal mesh (T)
3) honeycomb (H)
4) generalized de-Bruijn (B)
5) generalized Kautz (K)

Table II
THROUGHPUT [MB/S]/AREA [MM^2] ACHIEVED FOR THE WIMAX INTERLEAVER (N=2400) WITH DIFFERENT TOPOLOGIES, P, R AND ROUTING ALGORITHMS (DCM APPROACH)

		D=2, ring				D=2, generalized Kautz			
		P=8	P=16	P=32	P=64	P=8	P=16	P=32	P=64
R=1.00	SSP-RR (PP)	115.50/1.64	132.30/3.85	147.06/5.71	152.28/6.91	104.35/2.05	140.85/3.48	195.44/5.17	270.27/7.17
	SSP-FL (PP)	112.89/1.56	134.38/3.18	147.78/4.41	139.86/5.43	108.99/1.85	149.07/3.21	209.06/4.65	287.77/6.35
	ASP-FT (AP)	130.15/1.40	144.75/2.87	152.87/3.97	142.35/5.02	108.99/1.73	149.07/2.90	209.06/4.07	287.77/5.41
R=0.50	SSP-RR (PP)	86.15/0.43	122.32/1.61	133.48/3.96	137.77/5.45	86.21/0.44	131.15/1.33	172.91/3.19	229.89/5.29
	SSP-FL (PP)	86.15/0.42	123.71/1.37	132.89/3.67	130.15/5.17	86.15/0.41	138.25/1.06	188.68/2.62	241.94/4.59
	ASP-FT (AP)	86.02/0.49	134.53/1.29	137.30/3.35	130.72/4.80	86.15/0.46	138.25/1.05	188.68/2.38	241.94/3.99
R=0.33	SSP-RR (PP)	57.80/0.41	101.10/0.74	122.08/2.96	125.92/5.01	57.86/0.39	102.48/0.75	155.44/1.82	195.76/3.97
	SSP-FL (PP)	57.78/0.39	100.84/0.72	121.58/2.67	120.97/4.87	57.80/0.38	102.21/0.67	161.51/1.43	207.25/3.28
	ASP-FT (AP)	57.83/0.46	100.84/0.81	122.70/2.52	121.70/4.56	57.80/0.44	102.21/0.74	161.51/1.40	207.25/2.94
		D=3, honeycomb				D=3, generalized Kautz			
		P=8	P=16	P=32	P=64	P=8	P=16	P=32	P=64
R=1.00	SSP-RR (PP)	113.21/1.69	184.05/2.29	181.27/5.16	323.45/6.47	156.45/0.83	203.74/2.06	314.96/3.60	428.57/5.84
	SSP-FL (PP)	114.83/1.67	187.79/2.22	179.64/4.37	314.96/5.99	166.67/0.67	229.89/1.93	332.41/3.41	451.13/5.49
	ASP-FT (AP)	127.93/1.51	247.42/1.64	242.91/3.62	385.85/4.93	166.67/0.67	250.52/1.58	339.94/2.98	456.27/4.64
R=0.50	SSP-RR (PP)	85.96/0.45	151.32/0.91	160.64/3.52	267.26/4.69	86.52/0.45	152.67/0.86	242.91/1.81	331.49/3.76
	SSP-FL (PP)	85.90/0.43	151.52/0.84	163.71/3.10	261.44/4.20	86.52/0.44	152.48/0.82	241.94/1.66	338.03/3.37
	ASP-FT (AP)	85.90/0.49	151.32/0.90	213.52/2.08	305.34/3.48	86.52/0.53	152.28/0.88	244.40/1.58	337.08/2.98
R=0.33	SSP-RR (PP)	57.72/0.40	102.48/0.80	144.75/2.27	223.88/3.49	58.00/0.44	103.18/0.78	167.13/1.56	243.90/3.15
	SSP-FL (PP)	57.72/0.40	102.48/0.79	148.88/2.01	226.42/3.21	58.00/0.44	103.09/0.78	168.07/1.50	240.48/2.96
	ASP-FT (AP)	57.75/0.46	102.65/0.86	162.38/1.58	233.92/2.87	58.00/0.50	103.09/0.83	168.07/1.48	242.91/2.70
		D=4, toroidal mesh				D=4, generalized Kautz			
		P=8	P=16	P=32	P=64	P=8	P=16	P=32	P=64
R=1.00	SSP-RR (PP)	123.84/1.17	171.67/2.10	187.21/4.24	310.08/6.24	140.19/0.94	268.46/1.56	334.26/3.31	517.24/5.29
	SSP-FL (PP)	129.87/1.14	167.83/2.00	200.33/3.67	323.45/5.77	155.24/0.82	281.69/1.30	347.83/3.13	550.46/4.92
	ASP-FT (AP)	165.29/0.69	282.35/1.30	357.14/2.79	497.93/4.52	167.13/0.67	281.69/1.24	397.35/2.49	550.46/4.28
R=0.50	SSP-RR (PP)	86.15/0.46	147.78/1.03	165.29/2.75	260.30/4.58	86.52/0.47	153.65/0.89	248.45/1.91	359.28/3.64
	SSP-FL (PP)	86.08/0.45	151.52/0.95	178.31/2.51	270.27/4.25	86.52/0.46	153.85/0.88	247.93/1.81	360.36/3.50
	ASP-FT (AP)	86.21/0.54	152.28/1.05	242.42/1.85	334.26/3.46	86.52/0.57	153.85/0.96	248.96/1.77	360.36/3.27
R=0.33	SSP-RR (PP)	57.80/0.44	102.92/0.93	154.64/1.97	223.46/3.81	58.00/0.46	103.54/0.88	169.01/1.75	248.96/3.46
	SSP-FL (PP)	57.80/0.44	102.92/0.91	163.49/1.79	226.42/3.57	58.00/0.45	103.54/0.86	169.25/1.69	248.45/3.34
	ASP-FT (AP)	57.83/0.50	102.92/0.98	166.90/1.79	238.57/3.25	58.00/0.53	103.54/0.92	169.25/1.68	248.45/3.15

Table III
THROUGHPUT [MB/S]/AREA [MM^2] ACHIEVED FOR THE CIRCULAR SHIFTING INTERLEAVER (N=24576) WITH DIFFERENT TOPOLOGIES, P, R AND ROUTING ALGORITHMS WITH DCM APPROACH. LIGHT-GRAY, MID-GRAY AND DARK-GRAY CELLS INDICATE THE HIGHEST THROUGHPUT, THE HIGHEST AREA AND THE LOWEST AREA POINTS FOR EACH D VALUE RESPECTIVELY

		D=2, ring				D=2, generalized Kautz			
		P=8	P=16	P=32	P=64	P=8	P=16	P=32	P=64
R=1.00	SSP-RR (PP)	62.56/6.58	72.23/15.73	81.22/24.45	87.04/30.37	56.62/8.43	77.26/14.10	116.01/20.56	169.96/26.72
	SSP-FL (PP)	62.57/6.84	73.89/13.90	82.98/18.67	88.25/22.11	59.52/8.09	83.52/13.50	125.31/18.55	183.79/23.53
	ASP-FT (AP)	71.48/5.93	81.57/11.29	88.17/15.12	91.14/19.36	59.52/6.94	83.52/10.74	125.31/13.90	183.79/16.74
R=0.50	SSP-RR (PP)	49.12/1.80	72.36/6.00	80.26/15.71	86.11/21.02	49.13/1.79	77.37/4.10	114.99/10.17	165.12/16.37
	SSP-FL (PP)	49.12/1.78	73.42/5.10	82.28/14.68	87.29/20.48	49.13/1.78	86.74/2.98	129.59/8.00	186.75/13.78
	ASP-FT (AP)	49.12/2.50	82.40/4.93	87.48/12.55	90.23/18.42	49.13/2.39	86.74/3.43	129.59/7.03	186.75/10.80
R=0.33	SSP-RR (PP)	32.78/1.76	63.67/2.09	79.52/11.40	85.06/19.10	32.78/1.76	63.83/2.06	111.61/4.13	162.20/9.53
	SSP-FL (PP)	32.78/1.76	63.68/2.04	81.51/9.65	86.27/18.43	32.77/1.75	63.81/2.01	123.57/2.66	186.58/6.51
	ASP-FT (AP)	32.78/2.51	63.67/3.37	86.71/9.32	88.90/17.17	32.77/2.44	63.81/2.99	123.57/3.71	186.58/6.45
		D=3, honeycomb				D=3, generalized Kautz			
		P=8	P=16	P=32	P=64	P=8	P=16	P=32	P=64
R=1.00	SSP-RR (PP)	63.28/6.51	107.23/8.07	103.96/20.19	219.19/21.14	87.61/2.86	118.27/6.64	210.05/10.39	332.65/16.23
	SSP-FL (PP)	64.03/6.67	109.73/8.31	106.61/16.64	214.53/20.29	97.37/2.06	135.54/6.08	220.22/10.86	350.29/16.37
	ASP-FT (AP)	72.48/5.74	152.42/5.28	160.67/11.92	313.79/13.45	97.37/2.20	153.26/4.55	239.53/8.07	375.55/11.66
R=0.50	SSP-RR (PP)	49.12/1.80	95.52/2.19	102.95/12.05	213.78/10.75	49.16/1.81	95.63/2.15	185.62/2.91	322.18/5.24
	SSP-FL (PP)	49.12/1.79	95.48/2.13	106.06/10.62	208.55/9.56	49.16/1.81	95.69/2.10	185.68/2.76	346.34/4.32
	ASP-FT (AP)	49.12/2.50	95.60/3.05	163.71/4.91	312.99/6.14	49.16/2.63	95.69/3.02	185.62/3.54	348.10/4.60
R=0.33	SSP-RR (PP)	32.78/1.77	63.84/2.08	102.14/5.83	205.69/5.13	32.79/1.79	63.89/2.08	124.30/2.68	235.31/3.96
	SSP-FL (PP)	32.78/1.76	63.84/2.06	108.17/4.43	216.03/4.54	32.79/1.79	63.89/2.07	124.27/2.62	235.58/3.78
	ASP-FT (AP)	32.78/2.50	63.85/2.99	123.62/4.18	233.35/5.14	32.79/2.47	63.89/2.95	124.30/3.62	235.58/4.68
		D=4, toroidal mesh				D=4, generalized Kautz			
		P=8	P=16	P=32	P=64	P=8	P=16	P=32	P=64
R=1.00	SSP-RR (PP)	70.18/4.23	97.20/6.89	110.05/13.77	202.77/16.59	77.99/3.34	174.15/3.71	215.50/8.33	493.89/10.22
	SSP-FL (PP)	73.67/4.04	96.02/6.58	117.84/11.43	214.68/15.00	86.09/3.21	184.17/2.97	232.38/8.38	516.74/9.82
	ASP-FT (AP)	96.57/2.36	184.12/3.13	275.76/6.43	471.89/9.19	97.11/2.35	184.17/3.06	298.83/5.09	516.74/7.61
R=0.50	SSP-RR (PP)	49.14/1.82	95.26/2.32	109.34/7.13	198.32/7.77	49.16/1.82	95.75/2.19	185.62/2.95	350.48/4.47
	SSP-FL (PP)	49.14/1.80	95.48/2.22	119.74/6.10	213.85/7.03	49.16/1.82	95.75/2.17	185.96/2.89	350.89/4.28
	ASP-FT (AP)	49.14/2.66	95.61/3.34	185.17/4.03	347.12/5.17	49.16/2.72	95.75/3.15	185.90/3.83	350.89/4.96
R=0.33	SSP-RR (PP)	32.78/1.79	63.85/2.17	110.01/3.59	196.55/5.02	32.79/1.81	63.91/2.15	124.37/2.82	236.04/4.17
	SSP-FL (PP)	32.78/1.78	63.85/2.15	123.10/2.97	216.80/4.56	32.79/1.80	63.92/2.14	124.40/2.78	235.94/4.06
	ASP-FT (AP)	32.78/2.42	63.85/3.03	124.00/3.95	234.15/5.26	32.79/2.51	63.92/3.02	124.40/3.72	235.94/4.96

for $P \in \{8, 16, 32, 64\}$, with SSP-RR, SSP-FL and ASP-FT routing algorithms and including DCM and SCM approaches for FIFO management. The SISO architecture parameters were set as follows: $\Delta = W/R$, $\theta = \tau = R^{-1}$ and backward

recursion order (RO=BRO). The choice of the window size has been obtained as follows. Usually, in optimized turbo decoder architectures W is chosen such that $N/(P \cdot W) \in \mathbb{N}$ to maximize the decoder throughput and simplify SISOs synchronization [24]. In some cases this condition can be easily satisfied (e.g. N=2400, P=8 and W=30), whereas in other cases it is not possible (e.g. N=5114, P=8 leads to $N/P \notin \mathbb{N}$). Given this situation we decided to fix a value of W for each N so that $N/(P \cdot W) \notin \mathbb{N}$. This choice avoids "lucky cases" with $N/(P \cdot W) \in \mathbb{N}$ and makes the comparison fair. Moreover, we tried to choose the W values as close as possible one to the other to avoid results that are too influenced by the W value itself. For each case, the Turbo NOC simulator provided the total number of cycles required to perform a complete iteration (interleaving and de-interleaving), the depth of each FIFO in the network, the content of each routing memory (see Fig. 1 (b)) and the $t'(i, j)$ sequence to be stored into the location memory (see Fig. 1 (b) and (c)). As a consequence, for each case we can estimate the achieved throughput for a certain clock frequency with a given number of iterations. Moreover, to characterize the complexity of each solution we give the synthesis results of all simulated networks for a 130 nm standard cell technology. Memories have been generated by means of a 130 nm memory generator. The area results concern all the nodes in the network where each node includes the blocks depicted in Fig. 1 except the SISO and the memory used to store the extrinsic information (shaded gray blocks in Fig. 1). The complexity characterization shown in this work does not consider post place and route area overhead. The actual routing overhead is expected to be quite limited for regular topologies, such as meshes and toroids, which can be implemented by means of a modular layout. For example in [46] and [47] it is shown that for regular topologies, at least at the 130 nm technology node, the area occupation of logic cells in the design gives a useful indication about the actual complexity of the NOC. Routing overhead in less regular topologies is somewhat more difficult to estimate before place and route. However, results are available in the literature showing that a low area layout can be generated for generalized de-Bruijn networks. For example in [48] and [49] an optimal layout algorithm is run on generalized de-Bruijn topologies leading to a layout comparable and in some cases smaller than the one required by toroidal meshes. More in general, since the layout area of a network x can be expressed as $\Omega(|B(x)|_{min}^2)$ [50], where $|B(x)|_{min}$ is the minimum bisection width of the network, this model can be used to roughly compare different topologies. For example, $|B(\text{T})|_{min} = 2\sqrt{P}$ and $|B(\text{B})|_{min} = \Omega(P/\log P)$ for toroidal mesh and for de-Bruijn topologies respectively: this shows that the routing overhead of de-Bruijn network is similar to that of toroidal meshes, at least up to 64 nodes.

As a significant case of study we consider each extrinsic information value represented on 8 bits. Thus, we represented λ on 8 bits for all the simulations, except the ones related to the WiMax permutation law. In fact, since the WiMax turbo code is double binary, its extrinsic information is an array made of three log-likelihood ratios, as a consequence a message is represented on 24 bits. Moreover, we consider $f_{clk} = 200$

MHz and $\hat{I} = 8$; thus, from (3) we can infer that to sustain a target throughput of $\hat{T} = 200$ Mb/s, we need at least $d \cdot P \cdot R = 16$, namely at least $P \cdot R = 16$ for binary codes and at least $P \cdot R = 8$ for double binary codes. However, due to the IL term in (2), higher values of the $P \cdot R$ product are also of interest.

The analysis of the experimental results obtained with the Turbo NOC simulator shows some interesting general properties.

1) SSP solutions adopting the FA node architecture are the most demanding implementations in terms of area. Since the PP node architecture achieves the same throughput as the FA solution with a lower area, in the following only the PP node architecture will be addressed.

2) The DCM FIFO management method performs better than the SCM one both in terms of throughput and complexity. As a consequence, in the following only results that are referred to the DCM approach will be presented.

3) Generalized de-Bruijn and generalized Kautz topologies achieve nearly the same results both in terms of throughput and complexity. In the following only results obtained with generalized Kautz topologies will be presented.

4) Results tend to be clustered into two families, namely short interleavers (WiMax interleaver with N=2400 and UMTS interleaver with N=5114) and long interleavers (prunable S-random interleaver with N=16384 and circular shifting interleaver with N=24576). For the sake of clarity, in the following, only results obtained for the WiMax interleaver (N=2400) and circular shifting interleaver (N=24576) will be presented.

The most significant experimental results are summarized in Table II and III that refer to the WiMax interleaver with $N = 2400$ and to the circular shifting interleaver with $N = 24576$ respectively. Each cell of the two tables gives the throughput in Mb/s and the area in mm^2 obtained for different P and R values, routing algorithms and architectures for the DCM approach. In Table III light-gray, mid-gray and dark-gray cells indicate the highest throughput, the highest area and the lowest area points for each D value respectively.

The most important conclusions that can be derived from results in Table II and III are:

1) In most cases, topologies with D=4 achieve higher throughput with lower complexity overhead than topologies with D=2 when $R \to 1$. On the other hand, when $R < 1$ increasing D from 2 to 3 always leads to a higher throughput and a smaller area, whereas increasing D from 3 to 4 leads to higher throughput but also to higher area.

2) The ASP-FT routing algorithm is the best performing solution both in terms of throughput and area when $R = 1$.

3) The routing memory overhead of the ASP-FT algorithm (see Fig. 1 (b)) becomes relevant as R decreases and SSP solutions become the best solutions mainly for $P = 8$ and $P = 16$.
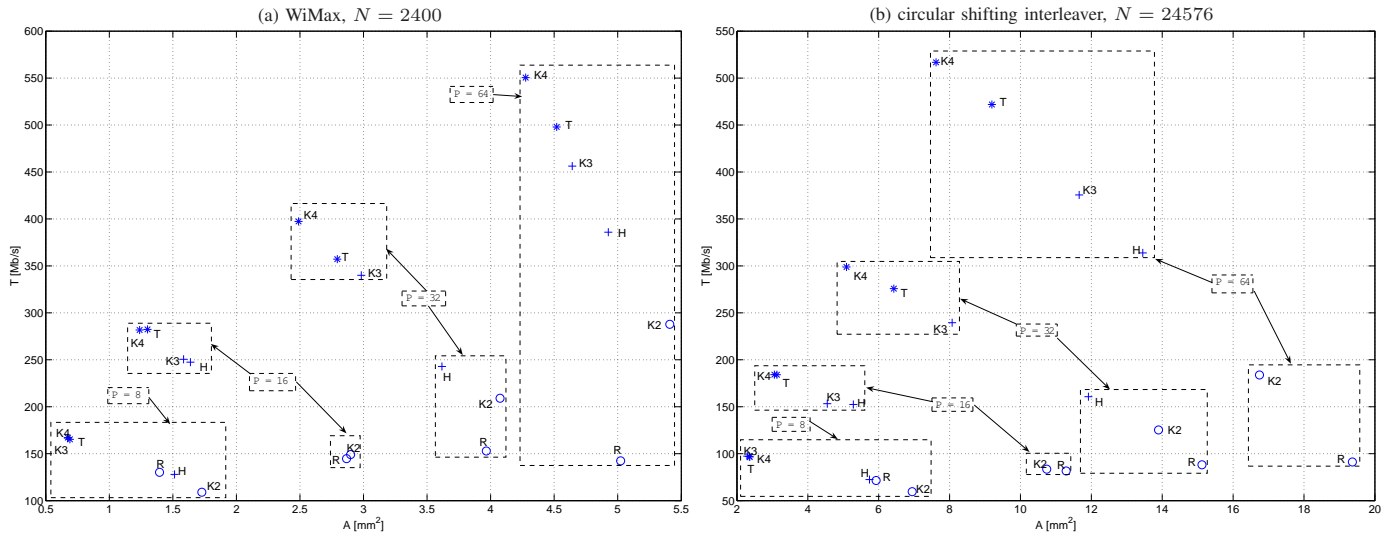
Figure 7. Throughput/area comparison of different topologies for the case $R = 1$, ASP-FT routing algorithm, DCM approach
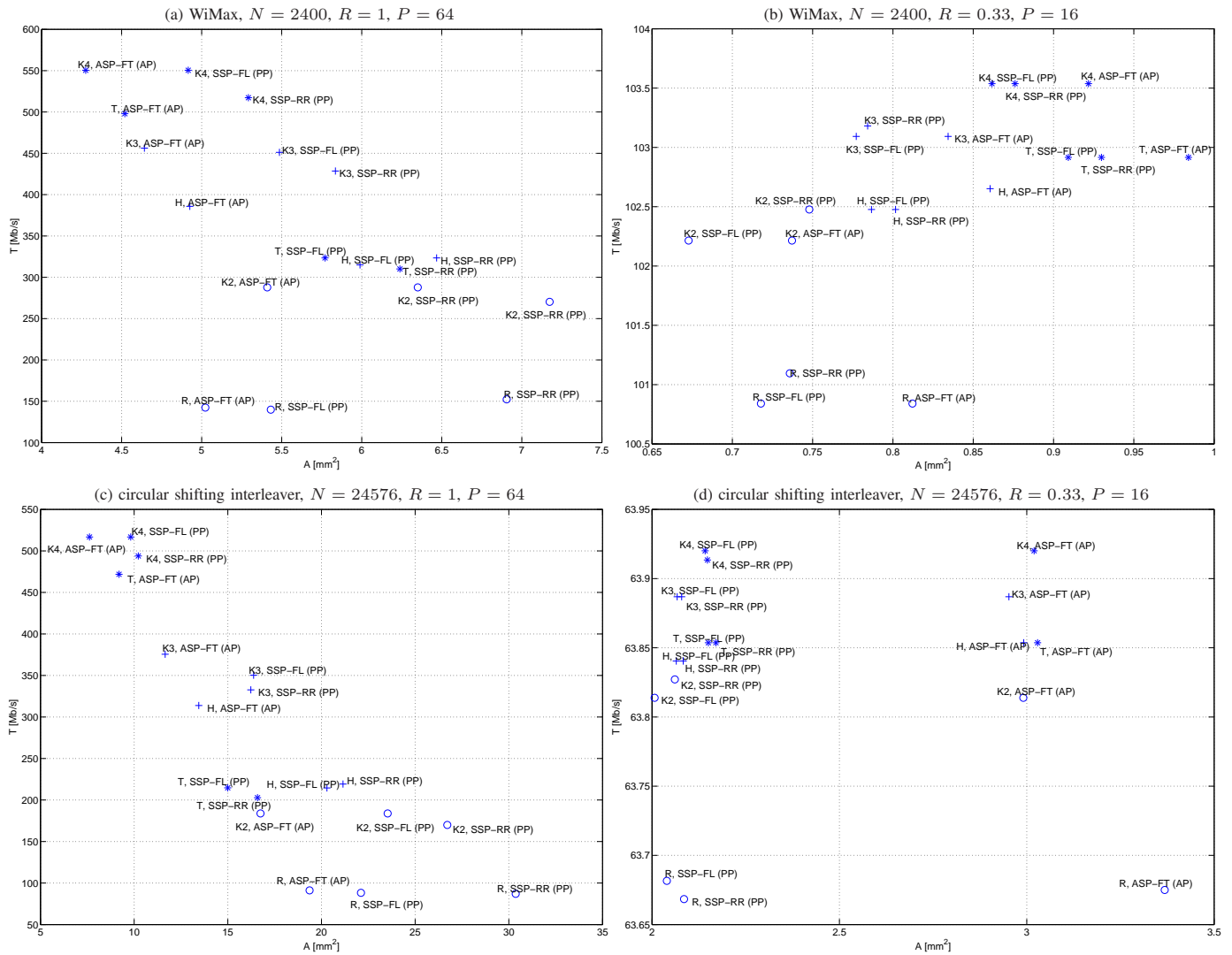

Figure 8. Throughput/area comparison of different topologies and routing algorithm with DCM approach: WiMax interleaver, $N = 2400$ for $R = 1$, $P = 64$ (a) and $R = 0.33$, $P = 16$ (b); circular shifting interleaver, $N = 24576$ for $R = 1$, $P = 64$ (c) and $R = 0.33$, $P = 16$ (d)

Table IV
HARDWARE RESOURCES BREAKDOWN FOR THE CIRCULAR SHIFTING INTERLEAVER WITH N=24576, DCM APPROACH: SOME SIGNIFICANT POINTS

D	top.	P	R	routing alg.	Tot. FIFOs area [mm^2]	Tot. CB area [mm^2]	Tot. reg. area [mm^2]	RA/M area [mm^2]	IM+LM area [mm^2]	Tot. area [mm^2]
2	R	64	1	ASP-FT (AP)	11.45 (59.15%)	0.03 (0.15%)	0.08 (0.41%)	6.35 (32.80%)	1.45 (7.49%)	19.36 (100%)
2	R	64	1	SSP-RR (PP)	27.46 (90.43%)	0.05 (0.16%)	0.14 (0.46%)	0.11 (0.36%)	2.61 (8.59%)	30.37 (100%)
2	R	8	0.33	SSP-FL/RR (PP)	0.05 (2.84%)	0.01 (0.57%)	0.01 (0.57%)	0.01 (0.57%)	1.68 (95.45%)	1.76 (100%)
2	K	64	0.5	ASP-FT (AP)	6.53 (60.46%)	0.03 (0.28%)	0.08 (0.74%)	2.71 (25.09%)	1.45 (13.43%)	10.80 (100%)
2	K	64	1	SSP-RR (PP)	23.81 (89.11%)	0.05 (0.19%)	0.14 (0.52%)	0.11 (0.41%)	2.61 (9.77%)	26.72 (100%)
2	K	8	0.33	SSP-FL (PP)	0.05 (2.86%)	0 (0%)$^{(1)}$	0.01 (0.57%)	0.01 (0.57%)	1.68 (96.00%)	1.75 (100%)
3	H	64	1	ASP-FT (AP)	9.25 (68.77%)	0.09 (0.67%)	0.11 (0.82%)	2.55 (18.96%)	1.45 (10.78%)	13.45 (100%)
3	H	64	1	SSP-RR (PP)	18.02 (85.24%)	0.10 (0.47%)	0.18 (0.85%)	0.23 (1.09%)	2.61 (12.35%)	21.14 (100%)
3	H	8	0.33	SSP-FL (PP)	0.05 (2.84%)	0.01 (0.57%)	0.01 (0.57%)	0.01 (0.57%)	1.68 (95.45%)	1.76 (100%)
3	K	64	1	ASP-FT (AP)	7.86 (67.41%)	0.09 (0.77%)	0.11 (0.94%)	2.15 (18.44%)	1.45 (12.44%)	11.66 (100%)
3	K	64	1	SSP-FL (PP)	13.15 (80.33%)	0.10 (0.61%)	0.18 (1.10%)	0.33 (2.02%)	2.61 (15.94%)	16.37 (100%)
3	K	8	0.33	SSP-FL (PP)	0.06 (3.35%)	0.01 (0.56%)	0.02 (1.12%)	0.02 (1.12%)	1.68 (93.85%)	1.79 (100%)
4	T	64	1	ASP-FT (AP)	5.14 (55.94%)	0.23 (2.5%)	0.13 (1.41%)	2.24 (24.37%)	1.45 (15.78%)	9.19 (100%)
4	T	64	1	SSP-RR (PP)	13.21 (79.63%)	0.17 (1.02%)	0.23 (1.39%)	0.37 (2.23%)	2.61 (15.73%)	16.59 (100%)
4	T	8	0.33	SSP-FL (PP)	0.06 (3.35%)	0.01 (0.56%)	0.02 (1.12%)	0.02 (1.12%)	1.67 (93.85%)	1.78 (100%)
4	K	64	1	ASP-FT (AP)	3.78 (49.67%)	0.22 (2.89%)	0.13 (1.71%)	2.03 (26.68%)	1.45 (19.05%)	7.61 (100%)
4	K	64	1	SSP-RR (PP)	6.86 (67.12%)	0.16 (1.57%)	0.23 (2.25%)	0.36 (3.52%)	2.61 (25.54%)	10.22 (100%)
4	K	8	0.33	SSP-FL (PP)	0.06 (3.33%)	0.10 (0.56%)	0.02 (1.11%)	0.03 (1.67%)	1.68 (93.33%)	1.80 (100%)

$^{(1)}$ The area and the percentage are not really zero, but they are negligible compared with the IM and LM contribution to the total area.

4) In most cases, generalized Kautz topologies are the best performing topologies.

As a significant example, in Fig. 7, we show the experimental results obtained with $R = 1$ and ASP-FT routing algorithm for the WiMax interleaver with $N = 2400$ (a) and the circular shifting interleaver with $N = 24576$ (b). Each point represents the throughput and the area obtained for a certain topology with a certain parallelism degree P. Results referred to the same P value are bounded into the same box and a label is assigned to each point to highlight the corresponding topology, namely topologies are identified as R-ring, H-honeycomb, T-toroidal mesh, K-generalized Kautz with the corresponding D value (K2, K3, K4).

As it can be observed, generalized Kautz topologies with $D = 4$ (K4) are always the best solutions to achieve high throughput with minimum area overhead. Moreover, we observe that increasing D has the effect of reducing network congestion, which has two positive consequences: higher throughput and smaller size of input FIFOs. However, the saved area benefit is partially mitigated by the increased complexity of the routing algorithm, as highlighted in the first part of Table V for PP node architectures: for example, K topology requires 23.81 mm^2 for FIFOs and 0.11 mm^2 for routing logic/memory when $D = 2$, whereas areas of 6.86 mm^2 and 0.36 mm^2 are occupied when $D = 4$. On the contrary, if an AP architecture is used for switching nodes, the routing memory size progressively becomes smaller with increasing D, due to the higher number of packets that can be switched at each clock cycle. For example, in the second part of Table V (AP node architecture), the area occupied by the routing memory (RA/M) reduces with increasing D.

In Fig. 8 significant results extracted from Table II and III are shown in graphical form. In particular, for $R = 1$ the ASP-FT routing algorithm is the best solution, whereas for $R < 1$ SSP routing algorithms, implemented as PP node architectures, tend to achieve the same performance as the ASP-FT routing algorithm with lower complexity overhead (see Fig. 8 (a) and (b) for the WiMax interleaver, $N = 2400$ and Fig. 8 (c) and

Table V
AREA OCCUPIED BY INPUT FIFOs AND ROUTING ALGORITHM/MEMORY (RA/M) FOR DIFFERENT TOPOLOGIES AND ROUTING ALGORITHMS WITH $P = 64$, CIRCULAR SHIFTING INTERLEAVER ($N = 24576$), $R = 1$ AND DCM APPROACH

Node arch.	routing alg.	D	top.	Tot. FIFOs area [mm^2]	RA/M area [mm^2]
PP	SSP-RR	2	R	27.46	0.11
		2	K	23.81	0.11
		3	H	18.02	0.23
		3	K	13.01	0.23
		4	T	13.21	0.37
		4	K	6.86	0.36
AP	ASP-FT	2	R	11.45	6.35
		2	K	12.38	2.79
		3	H	9.25	2.55
		3	K	7.86	2.15
		4	T	5.14	2.24
		4	K	3.78	2.03

(d) for the circular shifting interleaver, $N = 24576$).

An interesting phenomenon that arises increasing the interleaver size is the performance saturation that can be observed in the Table III for $D = 2$ topologies, namely the throughput tends to saturate and increasing R has the effect of augmenting the area with a negligible increase or even with a decrease of throughput. As an example, the generalized Kautz topology with $P = 64$ and ASP-FT routing algorithm achieves more than 180 Mb/s with $R = 1$, $R = 0.5$, $R = 0.33$. However, the solution with the smallest area is the one obtained with $R = 0.33$.

The throughput flattening of low D topologies can be explained by observing that high values of R tend to saturate the network. Furthermore, high values of R lengthen the input FIFOs as highlighted in Table IV, where the total area of the network is given as the breakdown of the building blocks, namely the input FIFOs, the crossbars (CB), the output registers, the routing algorithm/memory (RA/M), the identifier memory (IM) and the location memory (LM) is given for some significant cases: the highest throughput (light-gray), the highest area (mid-gray), and lowest area (dark-gray) points for

each D value in Table III.

IX. Conclusions

In this work a general framework to design network on chip based turbo decoder architectures has been presented. The proposed framework can be adapted to explore different topologies, degrees of parallelism, message injection rates and routing algorithms. Experimental results show that generalized de-Bruijn and generalized Kautz topologies achieve nearly the same results both in terms of throughput and complexity. Moreover, as highlighted in Table II and III generalized Kautz topologies achieve high throughput with a limited complexity overhead. In particular, in most cases, topologies with D=4 achieve higher throughput with lower complexity overhead than topologies with D=2 when $R \to 1$. On the other hand, when $R < 1$ increasing D from 2 to 3 always leads to a higher throughput and a smaller area, whereas increasing D from 3 to 4 leads to higher throughput but also to higher area. Finally, depending on the target throughput requirements different parallelism degrees, message injection rates and routing algorithms can be used to minimize the network area overhead.

Acknowledgment

References

[1] T. Vogt and N. Wehn, "Reconfigurable ASIP for convolutional and turbo decoding in an SDR environment," *IEEE Transactions on VLSI*, vol. 16, no. 10, pp. 1309–1320, Oct 2008.

[2] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error correcting coding and decoding: Turbo codes," in *IEEE International Conference on Communications*, 1993, pp. 1064–1070.

[3] C. Berrou, M. Jezequel, C. Douillard, and S. Kerouedan, "The advantages of non-binary turbo codes," in *IEEE Information Theory Workshop*, 2001, pp. 61–63.

[4] R. Dobkin, M. Peleg, and R. Ginosar, "Parallel interleaver design and VLSI architecture for low-latency MAP turbo decoders," *IEEE Transactions on VLSI*, vol. 13, no. 4, pp. 427–438, Apr 2005.

[5] J. H. Kim and I. C. Park, "Double-binary circular turbo decoding based on border metric encoding," *IEEE Transactions on Circuits and Systems II*, vol. 55, no. 1, pp. 79–83, Jan 2008.

[6] M. Martina, M. Nicola, and G. Masera, "A flexible UMTS-WiMax turbo decoder architecture," *IEEE Transactions on Circuits and Systems II*, vol. 55, no. 4, pp. 369–373, Apr 2008.

[7] J. H. Kim and I. C. Park, "Bit-level extrinsic information exchange method for double-binary turbo codes," *IEEE Transactions on Circuits and Systems II*, vol. 56, no. 1, pp. 81–85, Jan 2009.

[8] C. H. Lin, C. Y. Chen, A. Y. Wu, and T. H. Tsai, "Low-power memory-reduced traceback MAP decoding for double-binary convolutional turbo decoder," *IEEE Transactions on Circuits and Systems I*, vol. 56, no. 5, pp. 1005–1016, May 2009.

[9] G. Masera, *Turbo Code applications: a journey from a paper to realization*. Springer, 2005, ch. 14 - "VLSI for turbo codes".

[10] E. Boutillon, C. Douillard, and G. Montorsi, "Iterative decoding of concatenated convolutional codes: Implementation issues," *Proceedings of the IEEE*, vol. 95, no. 6, pp. 1201–1227, Jun 2007.

[11] A. Giulietti, L. V. der Perre, and M. Strum, "Parallel turbo coding interleavers: avoiding collisions in accesses to storage elements," *IET Electronics Letters*, vol. 38, no. 5, pp. 232–234, Feb 2002.

[12] J. Kwak and K. Lee, "Design of dividable interleaver for parallel decoding in turbo codes," *IET Electronics Letters*, vol. 38, no. 22, pp. 1362–1364, Oct 2002.

[13] M. J. Thul, N. Wehn, and L. P. Rao, "Enabling high-speed turbo-decoding through concurrent interleaving," in *IEEE International Symposium on Circuits and Systems*, 2002, pp. 897–900.

[14] A. Tarable and S. Benedetto, "Mapping interleaving laws to parallel turbo decoder architectures," *IEEE Communications Letters*, vol. 8, no. 3, pp. 162–164, Mar 2004.

[15] A. Polydoros, "Algorithmic aspects of radio flexibility," in *IEEE International Symposium on Personal, Indoor and Mobile Communications*, 2008, pp. 1–5.

[16] B. Bougard, R. Priewasser, L. V. der Perre, and M. Huemer, "Algorithm-architecture co-design of a multi-standard FEC decoder ASIP," in *ICT Mobile Summit Conference*, 2008.

[17] O. Muller, A. Baghdadi, and M. Jezequel, "From parallelism levels to a multi-ASIP architecture for turbo decoding," *IEEE Transactions on VLSI*, vol. 17, no. 1, pp. 92–102, Jan 2009.

[18] M. J. Thul, F. Gilbert, and N. Wehn, "Optimized concurrent interleaving architecture for high-throughput turbodecoding," in *IEEE International Conference on Electronics, Circuits and Systems*, 2002, pp. 1099–1102.

[19] ——, "Concurrent interleaving architectures for high-throughput channel coding," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2003, pp. 613–616.

[20] F. Speziali and J. Zory, "Scalable and area efficient concurrent interleaver for high throughput turbo-decoders," in *IEEE Euromicro Symposium on Digital System Design*, 2004, pp. 334–341.

[21] C. Neeb, M. J. Thul, and N. Wehn, "Network-on-chip-centric approach to interleaving in high throughput channel decoders," in *IEEE International Symposium on Circuits and Systems*, 2005, pp. 1766–1769.

[22] H. Moussa, O. Muller, A. Baghdadi, and M. Jezequel, "Butterfly and Benes-based on-chip communication networks for multiprocessor turbo decoding," in *Design, Automation and Test in Europe Conference and Exhibition*, 2007, pp. 654–659.

[23] H. Moussa, A. Baghdadi, and M. Jezequel, "Binary de Bruijn interconnection network for a flexible LDPC/turbo decoder," in *IEEE International Symposium on Circuits and Systems*, 2008, pp. 97–100.

[24] M. Martina, M. Nicola, and G. Masera, "Hardware design of a low complexity, parallel interleaver for wimax duo-binary turbo decoding," *IEEE Communications Letters*, vol. 12, no. 11, pp. 846–848, Nov 2008.

[25] O. Muller, A. Baghdadi, and M. Jezequel, "Bandwidth reduction of extrinsic information exchange in turbo decoding," *IET Electronics Letters*, vol. 42, no. 19, pp. 1104–1105, Sep 2006.

[26] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Transactions on Information Theory*, vol. 20, no. 3, pp. 284–287, Mar 1974.

[27] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Soft-input soft-output modules for the construction and distributed iterative decoding of code networks," *European Transactions on Telecommunications*, vol. 9, no. 2, pp. 155–172, Mar/Apr 1998.

[28] O. Muller, A. Baghdadi, and M. Jezequel, "ASIP-based multiprocessor SoC design for simple and double binary turbo decoding," in *Design, Automation and Test in Europe Conference and Exhibition*, 2006, pp. 1330–1335.

[29] L. Benini and G. D. Micheli, "Networks on chips: a new soc paradigm," *IEEE Computer*, vol. 35, no. 1, pp. 70–78, Jan 2002.

[30] L. Benini, "Application specific NoC design," in *Design, Automation and Test in Europe Conference and Exhibition*, 2006, pp. 1330–1335.

[31] F. Vacca, H. Moussa, A. Baghdadi, and G. Masera, "Flexible architectures for LDPC decoders based on network on chip paradigm," in *Euromicro Conference on Digital System Design*, 2009, pp. 582–589.

[32] C. A. Zeferino, M. E. Kreutz, L. Carro, and A. A. Susin, "A study on communication issues for systems-on-chip," in *IEEE Symposium on Integrated Circuits and Systems Design*, 2002, pp. 121–126.

[33] M. Imase and M. Itoh, "Design to minimize diameter on building-block network," *IEEE Transactions on Computers*, vol. 30, no. 6, pp. 439–442, Jun 1981.

[34] ——, "A design for directed graphs with minimum diameter," *IEEE Transactions on Computers*, vol. 32, no. 8, pp. 782–784, Aug 1983.

[35] B. Parhami and D. M. Kwai, "A unified formulation of honeycomb and diamond networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 12, no. 1, pp. 74–80, Jan 2001.

[36] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003.

[37] M. Martina, "Turbo NOC: Network On Chip based turbo decoder architectures," downloadable at www.vlsilab.polito.it/~martina.

[38] "http://www.systemc.org."

[39] Y. Zhang and K. K. Parhi, "Parallel turbo decoding," in *IEEE International Symposium on Circuits and Systems*, 2004, pp. 509–512.

[40] A. Abbasfar and K. Yao, "An efficient and practical architecture for high speed turbo decoders," in *IEEE Vehicular Technology Conference*, 2003, pp. 337–341.

[41] O. Muller, A. Baghdadi, and M. Jezequel, "Exploring parallel processing levels for convolutional turbo decoding," in *IEEE International Conference on Information and Communication Technologies: from Theory to Applications*, 2006, pp. 2353–2358.

[42] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Algorithm for continuous decoding of turbo codes," *IET Electronics Letters*, vol. 32, no. 4, pp. 314–315, Feb 1996.

[43] J. Bang-Jensen and G. Gutin, *Digraphs: Theory, Algorithms and Applications, 2nd edition*. London, UK: Springer-Verlag, 2008.

[44] L. Dinoi and S. Benedetto, "Variable-size interleaver design for parallel turbo decoder architectures," *IEEE Transactions on Communications*, vol. 53, no. 11, pp. 1833–1840, Nov 2005.

[45] S. Dolinar and D. Divsalar, "Weight distributions for turbo codes using random and nonrandom permutations," *TDA Progress Report*, vol. 42-122, pp. 56–65, Aug 1995.

[46] F. Angiolini, P. Meloni, S. M. Carta, L. Raffo, and L. Benini, "A layout-aware analysis of networks-on-chip and traditional interconnects for MPSoCs," *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems*, vol. 26, no. 3, pp. 421–434, Mar 2007.

[47] P. Meloni, I. Loi, F. Angiolini, S. Carta, M. Barbaro, L. Raffo, and L. Benini, "Area and power modeling for networks-on-chip with layout awareness," *VLSI Design*, 2007, special issue on Networks on Chip, Article ID 50285, 12 pages.

[48] M. Hosseinabady, M. R. Kakoee, J. Mathew, and D. K. Pradhan, "Reliable network-on-chip based on generalized de Bruijn graph," in *IEEE International High Level Design Validation and Test Workshop*, 2007, pp. 3–10.

[49] ——, "De Bruijn graph as a low latency scalable architecture for energy efficient massive NoCs," in *Design, Automation and Test in Europe Conference and Exhibition*, 2008, pp. 1370–1373.

[50] M. R. Samatham and D. K. Pradhan, "The De Bruijn multiprocessor network: A versatile parallel processing and sorting network for VLSI," *IEEE Transactions on Computers*, vol. 38, no. 4, pp. 567–581, Apr 1989.

PLACE PHOTO HERE

Maurizio Martina was born in Pinerolo, Italy, in 1975. He received the M.Sc. and Ph.D. in electrical engineering from Politecnico di Torino, Italy, in 2000 and 2004 respectively. He is currently a Postdoctoral Researcher at the VLSI Lab, Politecnico di Torino. His research activities include VLSI design and implementation of architectures for digital signal processing and comunications.

PLACE PHOTO HERE

Guido Masera received the Dr.Eng. degree (summa cum laude) in 1986, and the Ph.D. degree in electrical engineering from Politecnico di Torino, Italy, in 1992. Since 1986 to 1988 he was with CSELT (Centro Studi e Laboratori in Telecomunicazioni, Torino, Italy) as a researcher involved in the standardization activities for the GSM system. Since 1992 he has been Assistant Professor and then Associate Professor at the Electronic Department, where he is a member of the VLSI-Lab group. His research interests include several aspects in the design of digital integrated circuits and systems, with special emphasis on high-performance architecture development (especially for wireless communications and multimedia applications) and on-chip interconnect modeling and optimization. He has coauthored more than 160 journal and conference papers in the areas of ASIC-SoC development, architectural synthesis, VLSI circuit modeling and optimization. In the frame of National and European research projects, he has been co-designer of several ASIC and FPGA implementations in the fields of Artificial Intelligence, Computer Networks, Digital Signal Processing, Transmission and Coding.