

A pluripotential theoretic framework for polynomial interpolation of vector-valued functions and differential forms

Original

A pluripotential theoretic framework for polynomial interpolation of vector-valued functions and differential forms / Bruni Bruno, L.; Piazzon, F.. - In: DOLOMITES RESEARCH NOTES ON APPROXIMATION. - ISSN 2035-6803. - 17:3(2024), pp. 97-113. [10.14658/PUPJ-DRNA-2024-3-12]

Availability:

This version is available at: 11583/3004890 since: 2025-11-06T10:15:13Z

Publisher:

Padova University Press

Published

DOI:10.14658/PUPJ-DRNA-2024-3-12

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

Improving Network-on-Chip-based Turbo Decoder Architectures

Maurizio Martina & Guido Masera

Journal of Signal Processing Systems
for Signal, Image, and Video Technology
(formerly the Journal of VLSI Signal
Processing Systems for Signal, Image,
and Video Technology)

ISSN 1939-8018
Volume 73
Number 1

J Sign Process Syst (2013) 73:83-100
DOI 10.1007/s11265-013-0733-7

Journal of
SIGNAL PROCESSING SYSTEMS
for Signal, Image, and Video Technology

Volume 73, No. 1, October 2013
Editor-in-Chief
S. Y. Kung
Co-Editor-in-Chief
Shuvra S. Bhattacharyya
Co-Editor-in-Chief
Jarmo Takala



CONTENTS

**Robust Recursive Beamforming in the Presence
of Impulsive Noise and Steering Vector Mismatch**
B. Liao · S.C. Chan 1

**Using Planar Embedded DRAM in Memory Intensive
Signal Processing Circuits: Case Studies on LDPC
Decoding and Motion Estimation**
K.S. Venkataraman · Y. Li · Q. Wu · N. Xie · H. Sun ·
N. Zheng · T. Zhang 11

PSO-based Fusion Method for Video Super-Resolution
M.-H. Cheng · K.-S. Hwang · J.-H. Jeng · N.-W. Lin 25

**Empirical Mode Decomposition:
Real-Time Implementation and Applications**
A. Eftekhar · C. Tournazou · E.M. Drakakis 43

**A Direction-Based Unsymmetrical-Cross Multi-
Hexagon-Grid Search Algorithm for H.264/AVC
Motion Estimation**
Z. Pan · S. Kwong 59

**Evolutionary Extreme Learning Machine
and Its Application to Image Analysis**
N. Liu · H. Wang 73

**Improving Network-on-Chip-based Turbo Decoder
Architectures**
M. Martina · G. Masera 83

**Palm-Dorsal Vein Recognition Method Based
on Histogram of Local Gabor Phase XOR
Pattern with Second Identification**
Z. Meng · X. Gu 101

 Springer

ISSN 1939-8018

Available
online
www.springerlink.com

Your article is protected by copyright and all rights are held exclusively by Springer Science +Business Media New York. This e-offprint is for personal use only and shall not be self-archived in electronic repositories. If you wish to self-archive your article, please use the accepted manuscript version for posting on your own website. You may further deposit the accepted manuscript version in any repository, provided it is only made publicly available 12 months after official publication or later and provided acknowledgement is given to the original source of publication and a link is inserted to the published article on Springer's website. The link must be accompanied by the following text: "The final publication is available at link.springer.com".

Improving Network-on-Chip-based Turbo Decoder Architectures

Maurizio Martina · Guido Masera

Received: 9 February 2012 / Accepted: 1 February 2013 / Published online: 19 March 2013
© Springer Science+Business Media New York 2013

Abstract In this work novel results concerning Network-on-Chip-based turbo decoder architectures are presented. Stemming from previous publications, this work concentrates first on improving the throughput by exploiting adaptive-bandwidth-reduction techniques. This technique shows in the best case an improvement of more than 60 Mb/s. Moreover, it is known that double-binary turbo decoders require higher area than binary ones. This characteristic has the negative effect of increasing the data width of the network nodes. Thus, the second contribution of this work is to reduce the network complexity to support double-binary codes, by exploiting bit-level and pseudo-floating-point representation of the extrinsic information. These two techniques allow for an area reduction of up to more than the 40 % with a performance degradation of about 0.2 dB.

Keywords NoC · Turbo decoder · VLSI

1 Introduction

Today, modern telecommunications are a pervasive experience of data exchange among users and devices. One critical aspect of this scenario is the continuous demand for higher data rates, a problem that is exacerbated by the need for reliable transmission of data. To that purpose, the push on the so-called beyond-3G technologies, such as WiMAX [1]

and 3GPP-LTE [2], is a possible answer, where the reliability is obtained exploiting effective error correcting codes, such as turbo [3] and Low-Density-Parity-Check (LDPC) [4] codes. Unfortunately, the decoding algorithms for these codes are iterative making high throughput implementations a challenging task [5, 6].

As shown in Table 1 in [7], several modern standards for communications use turbo codes as a reliable channel coding scheme. However, since these codes have limited similarities, flexible architectures able to support different standards are interesting solutions to achieve interoperability [8]. This direction has been investigated in several works, such as [9–19] where not only flexibility but also high throughput, achieved by the means of parallel architectures, is addressed. As an example [11, 13, 16, 20–22] deal with optimized Application-Specific-Integrated-Circuit (ASIC) architectures where the flexibility is limited to two standards: UMTS/WiMAX [11], 3GPP-LTE/WiMAX [13, 21, 22], W-CDMA/CDMA2000 [20] 3GPP-LTE/HSDPA [16]. On the other hand, [9, 10, 14, 15, 23] are based on the Application-Specific-Instruction-set-Processor (ASIP) approach, where optimized processor-like architectures are designed with Coware Processor Designer based on the LISA description language [24]. It is worth observing that ASIP-based solutions allow for greater flexibility than ASIC-based architectures, as they can support several different codes and standards. In this direction mixed Turbo/LDPC decoder architectures have been proposed either as ASIC [17, 18] or ASIP [19] architectures. However, mixed code decoder architectures are not discussed in this work. Moreover, as suggested in [14], ASIP solutions are well suited to implement high throughput multiprocessor turbo decoder architectures [7].

Several works, including [7, 25–29] address the problem of communication flexibility in multistandard turbo decoder

M. Martina (✉) · G. Masera
Dipartimento di Elettronica e Telecomunicazioni, Politecnico di Torino, C.so Duca degli Abruzzi 24, 10129 Torino, Italy
e-mail: maurizio.martina@polito.it

G. Masera
e-mail: guido.masera@polito.it

architectures. Recently, in [30] we introduced the concept of intra-IP Network-on-Chip (NoC), where the well known NoC paradigm [31–34] is applied to the communication structure of processing elements that belong to the same IP [35]. Intra-IP NoC [7, 25–28] is a flexible solution to enable multi-ASIP turbo decoder architectures. However, as shown in detail in [7, 28], flexibility comes at the expense of increasing the complexity of the decoder architecture. In this work we improve the complexity/performance trade-off of NoC-based turbo decoder architectures by reducing the traffic load on the network as suggested in [36]. In [36] techniques to reduce the traffic load are proposed and their effect in terms of bandwidth reduction is analyzed. However, the resulting improvement of throughput is not investigated. In this work the actual effect of these techniques on a complete NoC-based turbo decoder architecture is studied for the first time. The adopted technique of traffic reduction offers in the best case a throughput improvement of more than 60 Mb/s and 40 Mb/s for binary and double-binary codes respectively. To the best of our knowledge this is the first work showing the benefit of traffic reduction techniques in an NoC-based turbo decoder architecture. Furthermore, we exploit two known techniques [37, 38], originally proposed to limit the amount of memory in turbo decoder architectures, as possible solutions to reduce the complexity of the NoC when double-binary turbo codes [39] are employed, as in the WiMAX standard. To the best of our knowledge this is the first work where the techniques proposed in [36–38] are jointly used to design an NoC-based turbo decoder architecture.

The paper is structured as follows: in Section 2 we recall the equations required to implement the decoding algorithm, whereas in Section 3 we describe the peculiar characteristics of an NoC-based turbo decoder architecture, including the architecture of routing elements, low-complexity routing algorithms and topologies. Section 4 describes the experimental setup we defined to increase the throughput and reduce the area of NoC-based turbo decoder architectures both in the case of binary and double-binary codes. To this purpose we considered the HSDPA and the 3GPP-LTE standards for the case of binary codes, and the WiMAX standard for the case of double-binary codes. Finally, in Section 5 conclusions are drawn.

2 Decoding Algorithms

Turbo code decoding algorithms are briefly reviewed in the following. For the sake of brevity, this description does not provide details that are not strictly required for the understanding of the NoC based decoding architectures considered in the following sections of the paper. For a deeper

knowledge of decoding algorithms used for turbo codes the reader can refer to [5].

Turbo codes are based on the concatenation (usually parallel) of two constituent Convolutional Codes (CC) (Fig. 1a), where CC1 process the the uncoded sequence u in natural order ($u1$), whereas CC2 encodes the uncoded sequence in scrambled order ($u2$) according to the interleaver Π . On the other hand, the decoder is made of two constituent decoders that exchange their data by means of an interleaver (Π) and a deinterleaver (Π^{-1}), see Fig. 1b. For the sake of brevity in the next paragraph we define the symbols used in Fig. 1a and b without specifying if they are related to CC1 or CC2.

The decoding algorithm of turbo codes is an iterative process made of two half iterations, one for each constituent decoder, where each half iteration is based on Maximum-A-Posteriori (MAP) estimation achieved by means of the BCJR algorithm [40], where Log-Likelihood-Ratio (LLR) representation is usually adopted [41]. During each half iteration one constituent decoder reads from a memory the intrinsic information $\lambda_k[c]$ received from the channel and the a-priori information $\lambda_k^{apr}[u]$ to produce extrinsic information $\lambda_k^{ext}[u]$. Extrinsic information computed in one half iteration becomes the a-priori information for the next half iteration. Based on the trellis notation shown in Fig. 1c and said \mathcal{U} the set of uncoded symbols, each constituent MAP decoder, often referred to as Soft-In-Soft-Out (SISO) module, computes

$$\lambda_k^{ext}[u] = \max_{e:u(e)=u}^* \{b^{ext}(e)\} - \max_{e:u(e)=\tilde{u}}^* \{b^{ext}(e)\} - \lambda_k^{apr}[u] \quad (1)$$

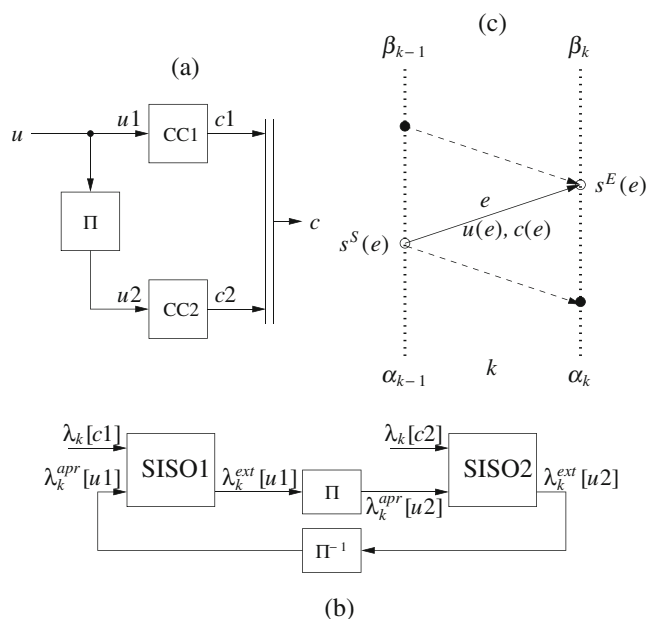


Figure 1 Parallel concatenation of two convolutional codes: encoder (a), decoder (b), notation for a trellis section (c).

where $\tilde{u} \in \mathcal{U}$ is an uncoded symbol taken as a reference (usually $\tilde{u} = \mathbf{0}$), $u \in \mathcal{U} \setminus \{\tilde{u}\}$, k is a trellis step, e is a transition in a trellis step and $u(e)$ is the corresponding uncoded symbol. Thus, $\lambda_k^{ext}[u]$ and $\lambda_k^{apr}[u]$ are extrinsic and a-priori information respectively for symbol u at trellis step k expressed as LLRs. The $\max^*\{x_i\}$ is usually implemented as a tree structure where each node of the tree is a 2-input \max^* function:

$$\max^*\{x_1, x_2\} = \max\{x_1, x_2\} + f_c(|x_1 - x_2|). \quad (2)$$

As it can be observed, the $f_c(\cdot)$ term in Eq. 2 is a correction term, that is often precalculated and stored in a small Look-Up-Table (LUT) [42, 43]. The correction term, usually adopted when decoding binary codes (Log-MAP), can be omitted for double-binary turbo codes with minor error rate performance degradation (Max-Log-MAP). In other words, each $\max^*\{\cdot\}$ function in Eq. 1 selects at each trellis step the maximum¹ metric $b^{ext}(e)$ among the ones whose uncoded symbol is u or \tilde{u} respectively.

The term $b^{ext}(e)$ in Eq. 1 is defined as:

$$b^{ext}(e) = \alpha_{k-1}[s^S(e)] + \gamma_k^{ext}[e] + \beta_k[s^E(e)] \quad (3)$$

$$\alpha_k[s] = \max_{e:s^E(e)=s} \left\{ \alpha_{k-1}[s^S(e)] + \gamma_k[e] \right\} \quad (4)$$

$$\beta_k[s] = \max_{e:s^S(e)=s} \left\{ \beta_{k+1}[s^E(e)] + \gamma_k[e] \right\} \quad (5)$$

$$\gamma_k[e] = \lambda_k[\mathbf{u}(e)] + \lambda_k[\mathbf{c}^u(e)] + \lambda_k[\mathbf{c}^p(e)] \quad (6)$$

$$\gamma_k^{ext}[e] = \lambda_k[\mathbf{c}^p(e)] \quad (7)$$

where $s^S(e)$ and $s^E(e)$ are the starting and the ending states of e , $\alpha_k[s^S(e)]$ and $\beta_k[s^E(e)]$ are the forward and backward metrics associated to $s^S(e)$ and $s^E(e)$ respectively. In other words, each α (β) metric is the \max^* value among the path metrics whose transition ends to (starts from) state s . The terms $\lambda_k[\mathbf{u}(e)]$, $\lambda_k[\mathbf{c}^u(e)]$ and $\gamma_k^{ext}[\mathbf{c}^p(e)]$ are obtained adding the corresponding a-priori, intrinsic systematic and intrinsic parity LLRs respectively.

In a parallel decoder P SISOs operate concurrently on disjoint portions of the trellis. Said N the number of trellis steps processed by each constituent decoder, we have that each SISO operates on a trellis slice made of N/P steps. As a consequence, we can extend the notation introduced in the previous paragraph to a parallel decoder, where $\lambda_{i,j}^{ext}[u]$ is the extrinsic information produced by SISO i at the j -th trellis step. Moreover, we observe that N is also the number of elements exchanged by means of Π and Π^{-1} . For further details on the decoding algorithm the reader can refer to [5].

3 NoC-based Turbo Decoder Architectures

An NoC-based turbo decoder architecture can be represented as a graph of P nodes where each node is made of a Routing Element (RE) and a Processing Element (PE) (see Fig. 2). Each PE, devoted to perform the processing required by the BCJR algorithm, i.e. Eqs. 1–7, contains a SISO processor and two memories where intrinsic and a-priori information are stored respectively. On the other hand, each RE has a simple structure made of M input buffers (FIFOs), an $M \times M$ crossbar switch and M output registers. REs are devoted to route the data produced by PEs to the correct destination node according to Π and Π^{-1} . To this purpose we introduce $d(i, j)$ as the destination node, or destination identifier (ID) of $\lambda_{i,j}^{ext}[u]$. In order to complete a half iteration, $\lambda_{i,j}^{ext}[u]$ is stored at the location $t(i, j)$ in the a-priori information memory of node $d(i, j)$. Since the network does not guarantee the delivery-order of the extrinsic information, in Fig. 2 we refer to extrinsic information coming from network as $\hat{\lambda}_{i,j}^{ext}[u]$. Similarly, we refer to the memory location where $\hat{\lambda}_{i,j}^{ext}[u]$ is stored as $\hat{t}(i, j)$. It is worth observing that $d(i, j)$ and $t(i, j)$ are univocally determined by the interleaver (deinterleaver) sequences, namely given Π , Π^{-1} and P we have

$$d(i, j) = \left\lfloor \frac{P}{N} \cdot \Theta \left(i \cdot \frac{N}{P} + j \right) \right\rfloor \quad (8)$$

$$t(i, j) = \Theta \left(i \cdot \frac{N}{P} + j \right) \bmod \frac{N}{P} \quad (9)$$

where $\lfloor \cdot \rfloor$ is the next lowest integer value and $\Theta(\cdot)$ can be either $\Pi(\cdot)$ or $\Pi^{-1}(\cdot)$ depending on the current half iteration.

In general PEs and REs can operate at different rates, thus, to decouple the design of PEs and REs we define R as the number of packets injected in the network in a clock cycle. As a consequence, $R = 1$ means that each PE injects in the network one new packet per clock cycle, whereas $R = 0.5$ means that a new packet is injected in the network every two clock cycles. It is worth noting that the case $R = 1$ corresponds to REs and PEs working at the same clock frequency (isochronous), with PEs able to output new packet of extrinsic information at each clock cycle. On the contrary, $R < 1$ models either an isochronous system where PEs output less than one packet per clock cycle or a mesochronous system where REs work at a higher clock frequency than PEs.

3.1 RE Architectures

In [28] three possible architectures for REs (see Fig. 2), referred to as Fully-Adaptive (FA), All Precalculated (AP)

¹If Log-MAP algorithms are used the result is not actually the maximum among the input values due to $f_c(x)$.

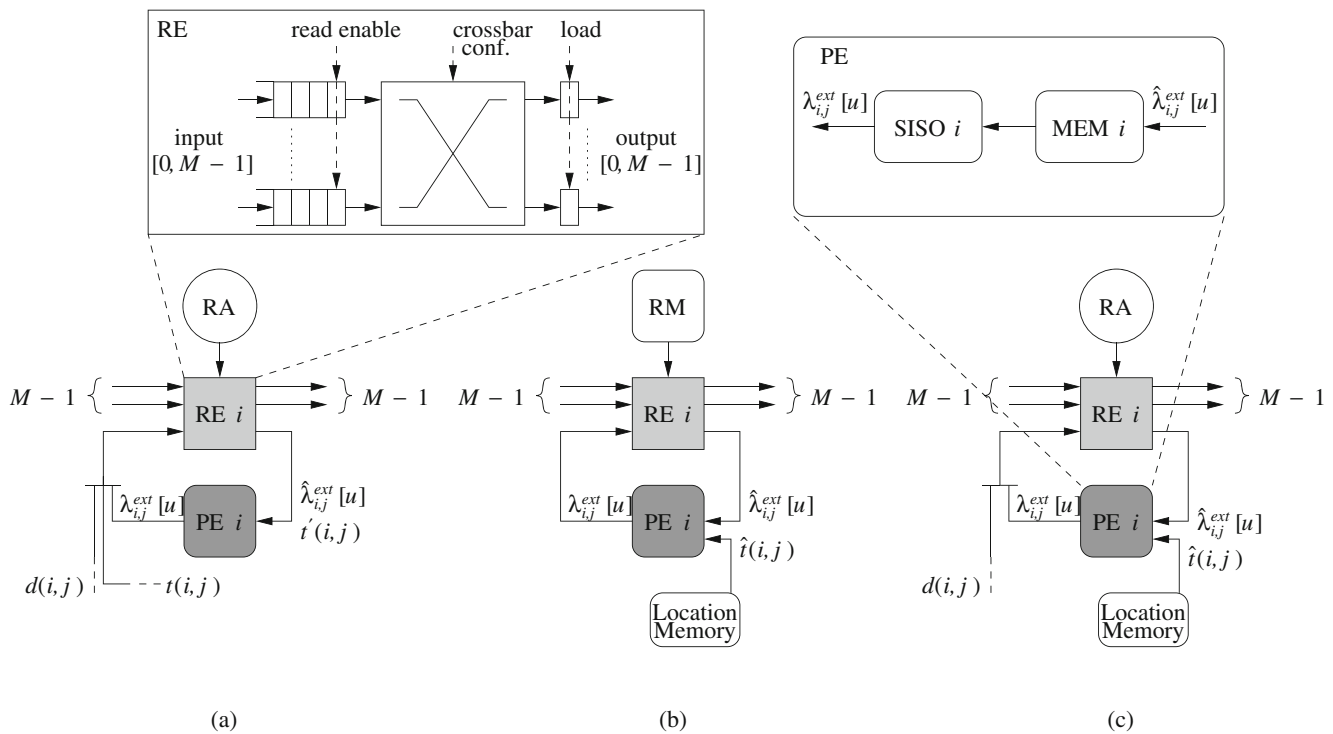


Figure 2 Node block scheme: **a** FA architecture, **b** AP architecture, **c** PP architecture.

and Partially Precalculated (PP) architectures were presented. All these architectures are intended to support distributed routing, namely each RE computes only the next link to send the packet towards the destination ID. Several works in the literature proposed efficient algorithmic implementations of distributed routing. As an example, in [44, 45] Logic-Based-Distributed-Routing (LBDR) and some improved versions are studied, whereas in [46] a general methodology for generating deadlock-free routing algorithms in irregular topologies is presented. On the other hand, since this work aims to improve the application specific system proposed in [28], the routing algorithm is based on forwarding tables for fair comparison.

The FA architecture (Fig. 2a) sends on the network packets of data made of a header, containing $d(i, j)$ and a payload containing $\lambda_{i,j}^{ext}[u]$ and $t(i, j)$. The data are routed by the means of a Routing Algorithm (RA) based on forwarding tables. The AP architecture (Fig. 2b) stems from Eqs. 8 and 9 observing that for each node we can precalculate and store in a Routing Memory (RM) and in a Location Memory the routing information (i.e read enable signals for FIFOs, crossbar configuration and load signals for registers) and $\hat{t}(i, j)$, the location where the received value $\hat{\lambda}_{i,j}^{ext}[u]$ will be stored, respectively. Thus, with the AP architecture we reduce the width of the data bus at the expense of some extra memory. In the PP architecture

(Fig. 2c) only $\hat{t}(i, j)$ sequences are precalculated. Thus, PP requires a narrower data width than the FA architecture, but less memory than the AP one.

To improve the throughput/area figures of NoC-based turbo decoder architecture we infer from [28] two main results:

- The AP architecture can be conveniently used with complex routing algorithms to concurrently maximize the throughput and minimize the area. Indeed, all read enable, crossbar configuration and load signals are computed off-line and stored in the RM. Unfortunately, as pointed out in [7] this comes at the expense of a significant amount of external memory to store the routing information; as an example to support all the interleavers specified by the HSDPA standard [47] about 64 MB of memory are required.
- As long as the network is faster than the PEs ($R < 1$), throughput and area figures tend to be independent of the routing algorithm.

Thus, both FA and PP architectures with simple RAs, based on forwarding tables, should be further investigated. In particular, the performance of the FA architecture can be improved by using Adaptive Bandwidth Reduction (ABR) techniques as the one proposed in [36], namely avoiding

the exchange of unnecessary extrinsic information values. This distinguishing feature of the FA architecture, that is not available with AP and PP architectures, is detailed in Section 4.1. On the contrary, the PP architecture features a narrower data bus than the FA one, however, it requires some external memory to store the configurations of all the Location Memories. Moreover, in several standards, such as HSDPA, 3GPP-LTE and WiMAX, the generation of $d(i, j)$ and $t(i, j)$ sequences can be obtained algorithmically with simple architectures [13, 48, 49]. As a consequence, the FA architecture can also take advantage of this feature to reduce the complexity of the whole decoder.

3.2 Low Complexity RAs

In order to increase the throughput and reduce the area of the decoder, RAs should be based on simple, deadlock-free and livelock-free routing policies than can be implemented with few logic and completed in one clock cycle. Simple routing architectures as LBDR and its improved versions [44, 45] are interesting solutions to reduce the complexity of routing elements. However, they are designed to work with topologies derived from the 2D-mesh. Unfortunately, this is not the case of the topologies studied in this work and in previous works we compare with, e.g. [7, 28]. In particular, in [28] it is shown that logarithmic topologies, e.g. De-Bruijn and Kautz topologies, are the most suited ones for NoC-based turbo decoder architectures as they reduce the latency overhead added by the interconnection structure with lower complexity than mesh-based topologies. As a consequence, in the following we will focus on distributed shortest-path routing based on forwarding tables for Kautz topologies. It is worth pointing out that, since the traffic pattern in turbo decoder architectures is imposed by the interleaver, buffers can be sized off-line through simulations [50]. As a consequence, a buffer or a link is always available avoiding deadlock. Moreover, the use of shortest-path routing prevents livelocks.

3.2.1 RAs Description

In a P nodes network, each node contains one or more tables with P entries. The value of the i -th entry is the output port connected to a node that is on one shortest path from current node to node i . As a consequence, the value of each entry depends on both the ID of current node and the destination ID of current packet. The value for each entry is obtained off-line by means of the Floyd-Warshall-Algorithm (FWA) [51]. Running the FWA on pruned versions of the network graph until no more local paths exist between one node and the adjacent ones we obtain the entries of all the tables. As an example in Fig. 3 it is shown that, in the generalized Kautz topology with $P = 8$ and $D = 4$, two equivalent

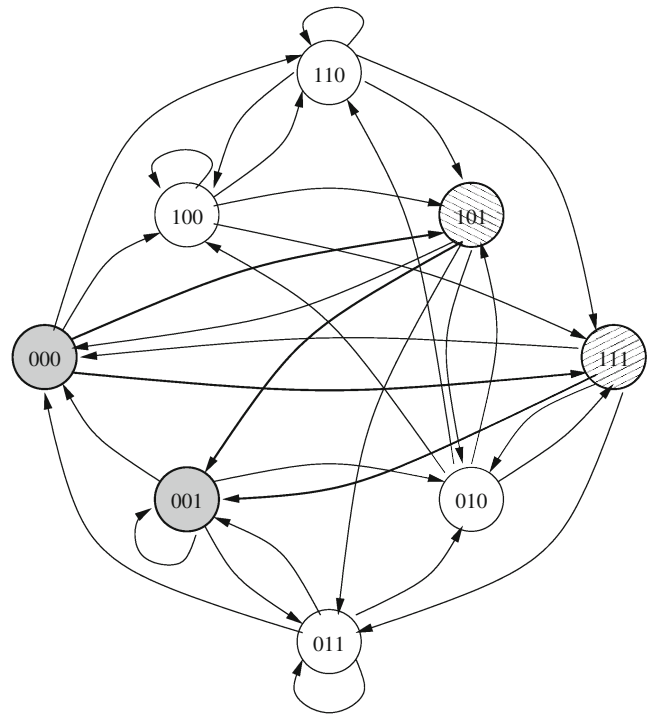


Figure 3 Shortest path routing algorithm example.

paths exist to go from node 0 to node 1 (through node 5 or node 7). Let us assume output ports numbered clockwise, that is for node 0 the link to node 6 is on port 0 and the link to node 7 is on port 3 (the link to the local SISO is always on port D). We build up to two forwarding tables for each node. As an example, from Fig. 3 we infer that the second entry (destination node 1) in the forwarding tables of node 0 are 2 and 3 respectively, the output ports connected to nodes 5 and 7. It is worth noting that when a message reaches the destination node it is routed to output port $M - 1$, that is directly connected to the extrinsic information memory (see Fig. 2).

To limit the complexity of the node in [28] is suggested to use only one forwarding table with FA and PP architectures. This solution is referred to as Single-Shortest-Path (SSP). On the other hand, with AP node architectures the complexity of multiple forwarding tables is hidden by the off-line computation of the routing memory content. Thus, in [28] local shortest paths are exploited to spread the traffic over the network. This solution referred to as All-local-Shortest-Path (ASP) aims at reducing the network congestion. SSP and ASP solutions are coupled with different policies for serving input buffers. In the SSP case simple serving policies, suitable for NoC-based turbo decoder architectures, are used: Round-Robin (RR) and FIFO-length (FL). RR is based on a circular serving policy, whereas with FL policies each input is served considering the number of elements stored in its input buffer, namely FL sorts the input buffers

according to the number of stored elements, then it serves them in decreasing order. In the ASP case an enhanced version of FL, referred to as FL-with-Traffic-spreading (FT), is used instead. The basic idea is to use a counter for each entry of the forwarding tables to obtain accurate information on the use of each path. Then, when two paths are equivalent (i.e. they are both shortest paths) the less used one is selected. Since the ASP-FT technique is rather complex it is implemented on AP nodes only and off-line simulations are run to obtain the content of the routing memory.

3.2.2 RAs Architecture

Input buffer management strategies and single-shortest-path routing algorithm described in previous paragraphs have been implemented as in [28]. The RR is implemented as a simple circular shift-register that is enabled when there is at least one element in the input FIFOs. On the other hand, FL relies on a small sorting network that, based on the number of elements into the input FIFOs, computes the serving order. At each node the forwarding table is implemented as an M -input- M -output LUT that converts M destinations (dst_i) to the corresponding output ports (dport_i). Then, given that S_0 and S_{M-1} are the first and the last served input FIFOs, M reservation blocks update an M -position binary mask to avoid collisions on output ports. Finally, a priority decoder implements the selected priority and FIFO management policies by properly generating the read enable (ren_i) signals for the FIFOs and the configuration command (adx_i) for the crossbar (see the bottom-right part of Fig. 4). Since the load enable for the registers (le_i) and adx_i signals must be asserted the clock cycle after ren_i , an early signal for the crossbar configuration (ladx_i) is generated and then delayed by means of registers. In particular, le_i and adx_i are obtained by delaying the ren_i and ladx_i by one clock cycle.

Reservation Block According to the order defined by the S_0, \dots, S_{M-1} sequence, each reservation block (bottom-left part of Fig. 4) receives the dport_i signals, 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 input rmask : if input rmask contains a '0' in position port_i , then reserve_i goes to '1'.

Priority Decoder The priority decoder is made of two blocks: the read-enable generator (upper-left part of

Fig. 4) and the destination-port generator (upper-right part of Fig. 4).

The read-enable generator is based on few logic gates that implement $\text{ren}_i = \text{reserve}_{S_i}$ when FIFO i is not empty ($\text{FIFOempty}_i = '0'$). This is obtained by combining S_i one-hot representation with the corresponding reserve signal.

The destination-port generator is an array of multiplexers, where each multiplexer in position i, i implements $\text{ladx}_i = \text{port}_i$ when $\text{ren}_i = '1'$. On the other hand, ladx_i must take the value of an un-reserved output port when $\text{ren}_i = '0'$. 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

$$\text{mux}_{j,0} = \begin{cases} 0 & \text{if } \text{port}_0 = j \\ j & \text{otherwise} \end{cases} \quad (10)$$

and for $i > 0$

$$\text{mux}_{j,i} = \begin{cases} \text{mux}_{i,i-1} & \text{if } \text{port}_i = j \\ \text{mux}_{j,k} & \text{otherwise} \end{cases} \quad (11)$$

where

$$k = \begin{cases} i - 2 & \text{if } j = i - 1 \\ i - 1 & \text{otherwise} \end{cases} \quad (12)$$

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

3.3 NoC Topologies

In [28] several fixed degree topologies for NoC-based turbo decoder architectures are considered. However, since Π and Π^{-1} tend to spread almost uniformly $\lambda_{i,j}^{ext}[u]$, the traffic pattern on the network is almost uniform too. Experimental results in [28] show that topologies with logarithmic diameter as generalized De-Bruijn [52] and generalized Kautz [53] achieve higher throughput and require lower area than other well known fixed degree topologies such as ring, honeycomb and toroidal-mesh ones. It is worth noting that Kautz topologies (shown in Fig. 5 for $P = 16$) have also been recently suggested as a viable solution to improve performance and latency of NoCs exploiting 3D VLSI design [54, 55].

4 Experimental Setup

Since in this work we aim at increasing the throughput and reducing the area of NoC-based turbo decoder architectures, we focus on the most significant cases discussed in Section 3, namely FA node architecture with SSP-RR

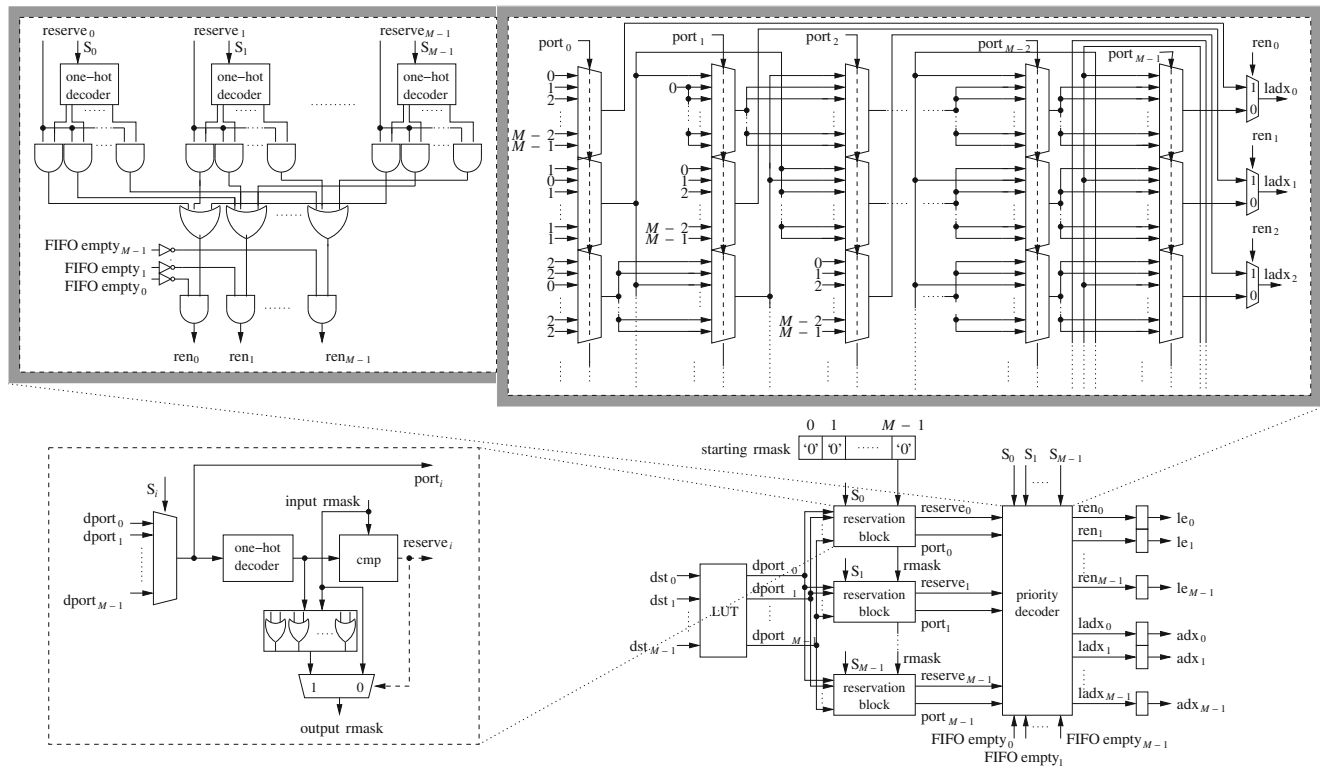


Figure 4 Single-shortest-path routing architecture.

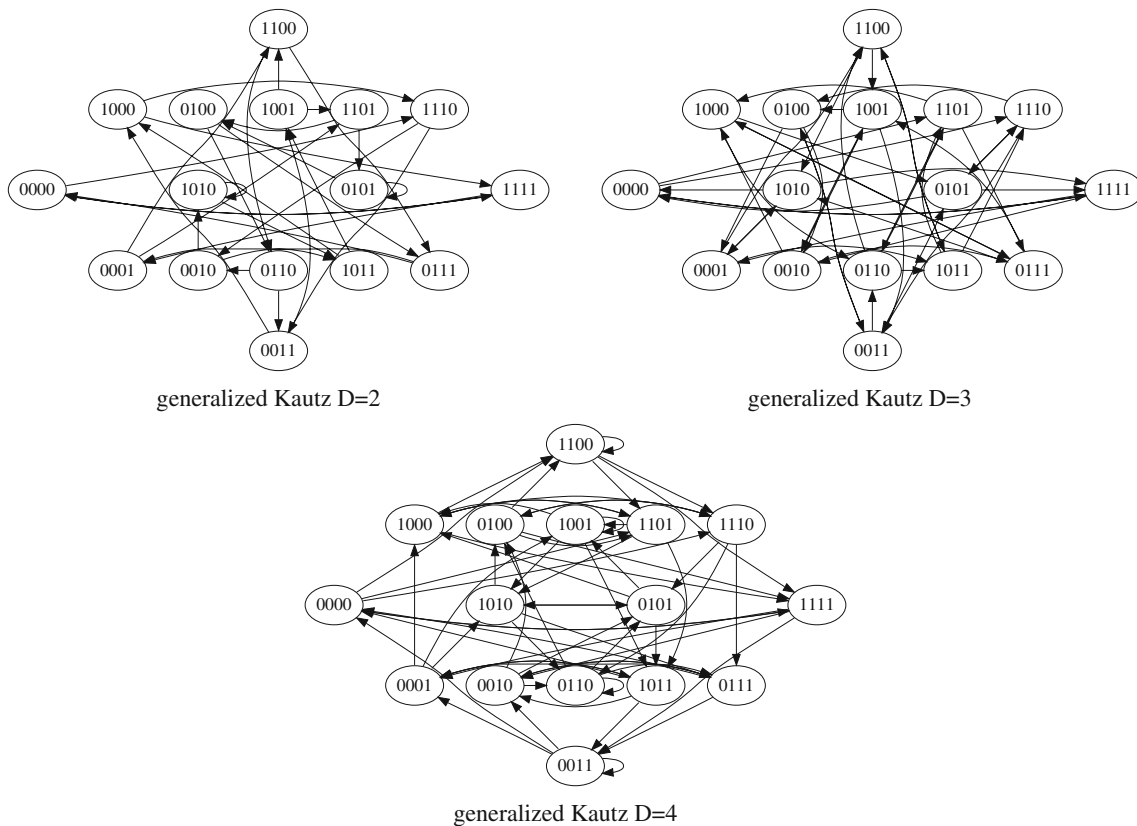
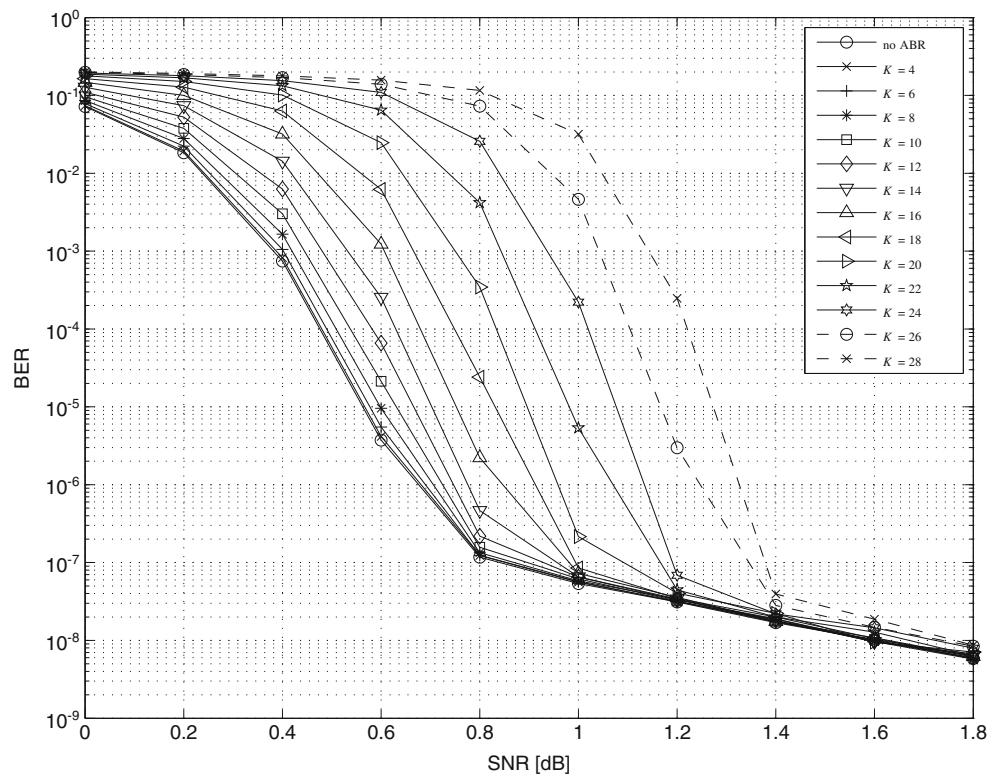


Figure 5 Example of considered Kautz topologies for $P = 16$.

Figure 6 BER performance of the HSDPA $N = 5114$ turbo decoder with ABR technique for different values of K .



and SSP-FL routing. Moreover, we consider only generalized Kautz topologies, as they have logarithmic diameter and less self-loops² than generalized De-Bruijn ones [28, 52, 53]. The degree of the network $D = M - 1$ ranges in $\{2, 3, 4\}$ and the parameter R varies in $\{0.33, 0.5, 1\}$. Then we simulated both HSDPA and 3GPP-LTE interleavers for the case of binary turbo codes. Furthermore, we simulated the double-binary turbo code used in the WiMAX standard as well.

In the following the throughput is computed as

$$T = \frac{N_b \cdot f_{clk}}{I \cdot (N_0^{cyc} + N_1^{cyc})} \quad (13)$$

where N_b is the number of decoded bits, f_{clk} is the clock frequency, I is the number of iterations, N_0^{cyc} and N_1^{cyc} are the number of clock cycles required to complete the interleaved and deinterleaved half iterations respectively. It is worth pointing out that $N_b = N$ for binary codes and $N_b = 2N$ for double-binary codes. Results shown in the following sections have been obtained for $f_{clk} = 200$ MHz and $I = 8$ with the Turbo-NoC simulator [50] and Synopsys Design Compiler for a 130 nm standard cell technology. The complexity characterization shown in this work does

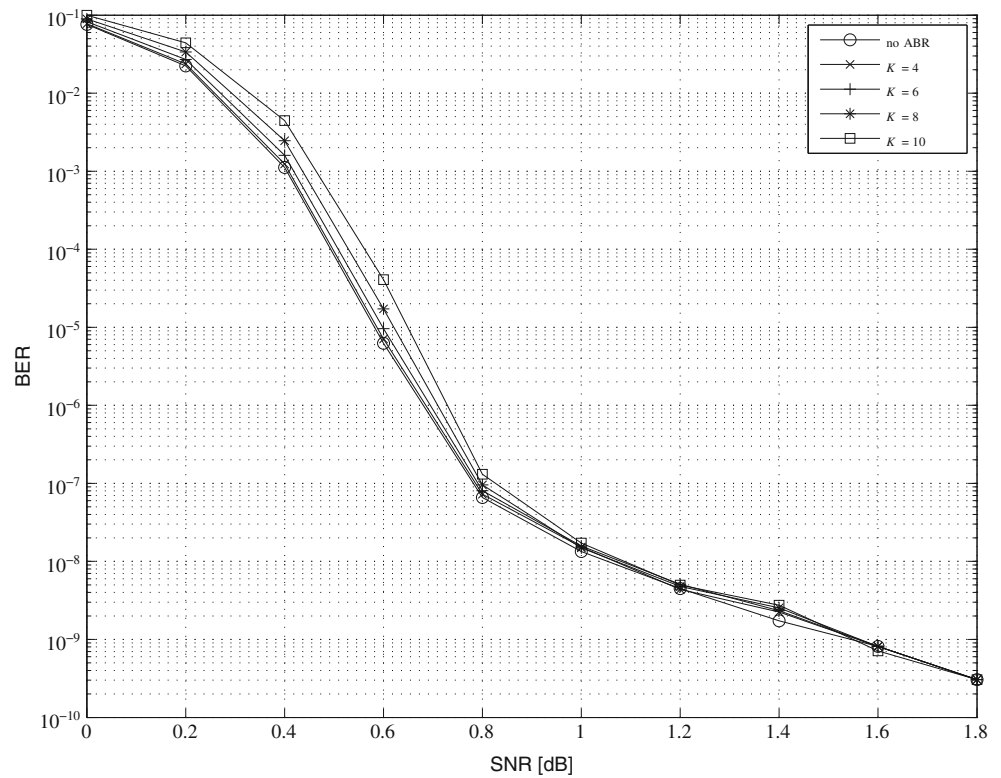
not consider post place and route area overhead [56]. However, as pointed out in [57] and [58] for regular topologies, at least at the 130 nm technology node, the area occupation of logic cells in the design gives a useful indication about the actual complexity of the NOC. Moreover, results are available in the literature showing that a low area layout can be generated for logarithmic diameter networks. For example in [59] and [60] 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 $\Omega(|B(x)|_{min}^2)$ [61], where $|B(x)|_{min}$ is the minimum bisection width of the network, this model can be used to roughly compare different topologies. For example, $|B(T)|_{min} = 2\sqrt{P}$ and $|B(B)|_{min} = \Omega(P/\log P)$ for toroidal mesh and for de-Bruijn topologies respectively: this shows that the routing overhead of logarithmic diameter networks is similar to that of toroidal meshes, at least up to 64 nodes.

4.1 ABR in NoC-based Turbo Decoder Architectures

Since ABR techniques reduce the traffic in the network, they reduce the latency of the decoder, i.e. N_0^{cyc} and N_1^{cyc} in Eq. 13. As a consequence, they are suited to increase the throughput of an NOC-based turbo decoder. This approach is similar to well known early stopping criteria that are routinely used to both increase the throughput and reduce the

²If we model a topology as a graph, a self-loop is an edge whose source and destination nodes coincide.

Figure 7 BER performance of the 3GPP-LTE $N = 6144$ turbo decoder with ABR technique, $K = 4, 6, 8, 10$.



power consumption in turbo decoder architectures [62–64]. However, most of related works focus on frame-level early stopping criteria. On the contrary, bit-level/symbol-level

early stopping criteria [65] take into account that the reliability of each bit/symbol in a frame converges at different speed. As a consequence, when the extrinsic information

Figure 8 Average throughput improvement at different SNR values of the HSDPA $N = 5114$ turbo decoder with $K = 4, 6, 8, 10$ on generalized Kautz networks $D = 2$ and $P = 64$.

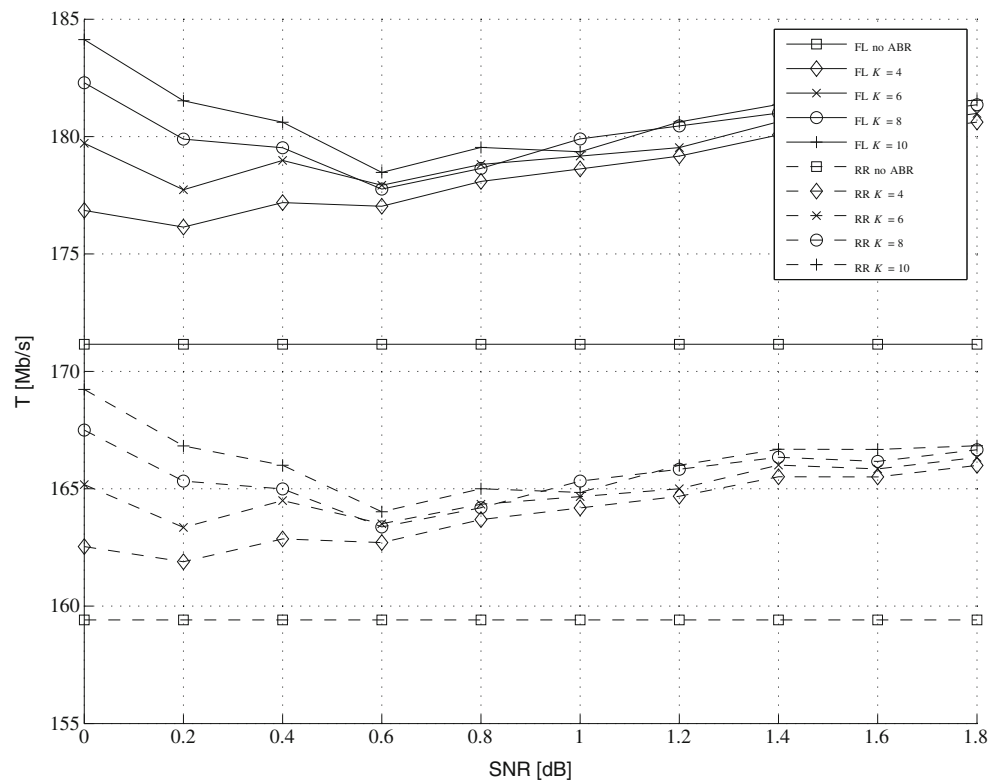
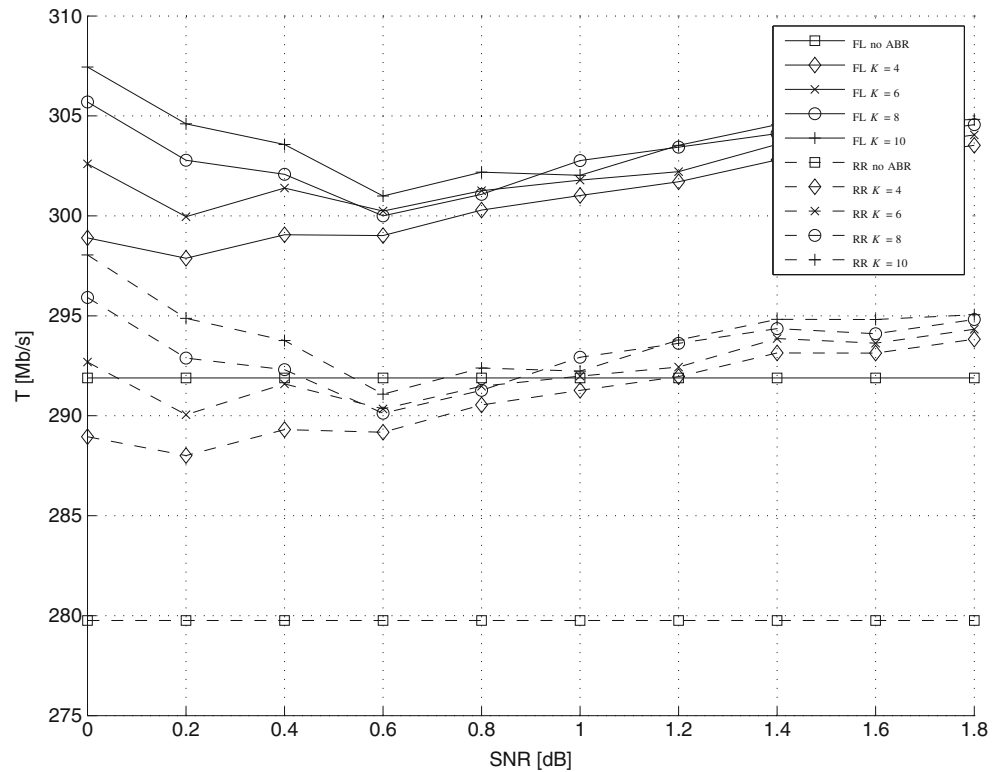


Figure 9 Average throughput improvement at different SNR values of the HSDPA $N = 5114$ turbo decoder with $K = 4, 6, 8, 10$ on generalized Kautz networks $D = 3$ and $P = 64$.



of a certain bit/symbol meets a proper reliability criterion, it is not necessary to further refine it. From an NoC-based turbo decoder perspective, this means that reliable $\lambda_{i,j}^{ext}[u]$ are no longer sent over the network.

4.2 HSDPA and 3GPP-LTE Case of Study

For binary turbo codes, as the ones employed in HSDPA and 3GPP-LTE standards, a simple ABR technique is obtained

Figure 10 Average throughput improvement at different SNR values of the HSDPA $N = 5114$ turbo decoder with $K = 4, 6, 8, 10$ on generalized Kautz networks $D = 4$ and $P = 64$.

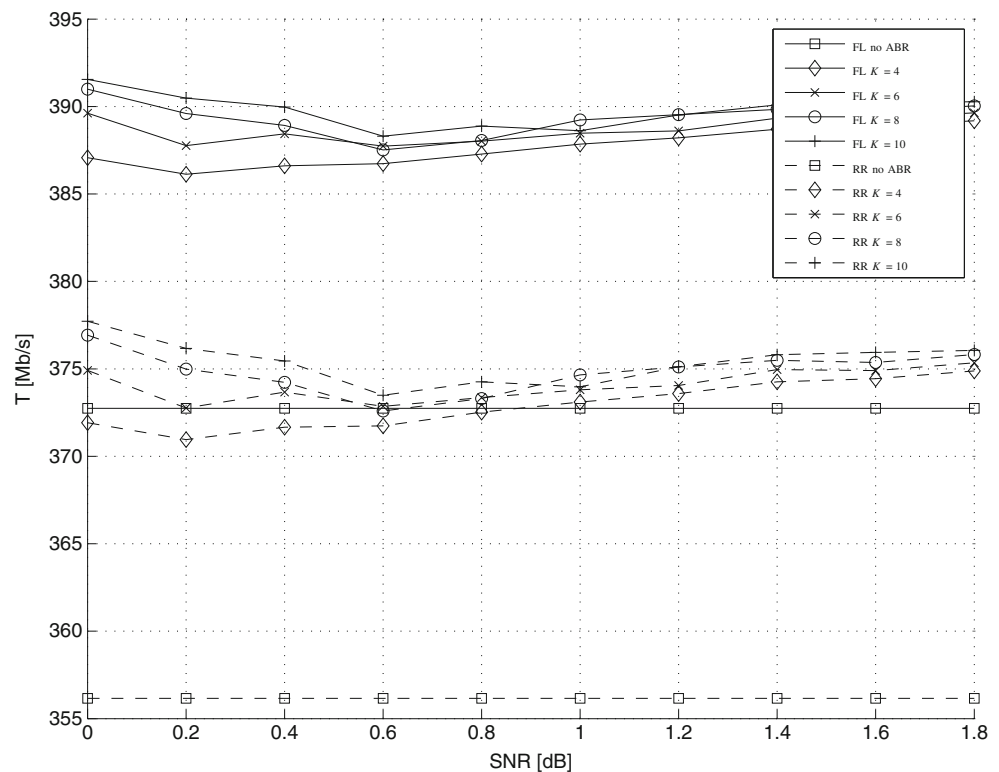


Table 1 Throughput [Mb/s] - area [mm²] achieved with the HSDPA $N = 5114$ and LTE $N = 6144$ interleavers, with generalized Kautz topologies for $P \in \{8, 16, 32, 64\}$, $R \in \{0.33, 0.5, 1\}$, SSP-RR, SSP-FL and ASP-FT routing algorithms, no ABR.

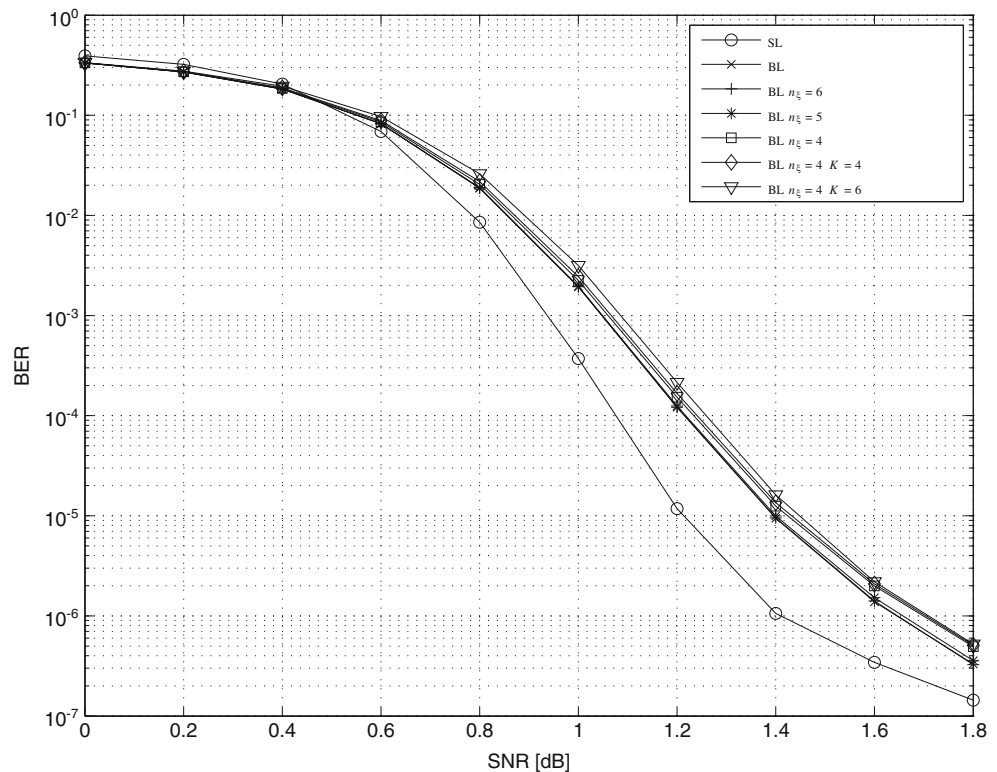
		$D = 2$, HSDPA				$D = 2$, LTE			
		$P = 8$	$P = 16$	$P = 32$	$P = 64$	$P = 8$	$P = 16$	$P = 32$	$P = 64$
$R = 1.00$	SSP-RR (FA)	54 - 3.13	74 - 5.29	105 - 7.36	159 - 9.71	53 - 3.69	71 - 6.25	101 - 8.76	140 - 11.14
	SSP-FL (FA)	58 - 3.16	81 - 5.04	117 - 6.65	171 - 8.48	54 - 3.88	75 - 5.93	109 - 7.79	151 - 9.74
	ASP-FT (AP)	58 - 1.53	81 - 2.51	117 - 3.40	171 - 4.51	54 - 2.01	75 - 3.01	109 - 4.00	151 - 5.14
$R = 0.50$	SSP-RR (FA)	46 - 0.59	72 - 1.71	101 - 4.07	142 - 6.64	44 - 0.64	66 - 1.95	90 - 4.63	120 - 7.44
	SSP-FL (FA)	46 - 0.56	78 - 1.26	112 - 3.39	156 - 5.76	44 - 0.61	72 - 1.37	100 - 3.79	131 - 6.31
	ASP-FT (AP) [7]	46 - 0.62	78 - 1.01	112 - 2.06	156 - 3.40	44 - 0.71	72 - 1.13	100 - 2.34	131 - 3.72
$R = 0.33$	SSP-RR (FA)	31 - 0.54	58 - 0.86	92 - 2.01	129 - 4.57	29 - 0.59	52 - 0.90	79 - 2.25	102 - 4.84
	SSP-FL (FA)	31 - 0.53	58 - 0.81	101 - 1.56	140 - 3.68	29 - 0.58	52 - 0.85	86 - 1.53	112 - 3.79
	ASP-FT (AP)	31 - 0.62	58 - 0.88	101 - 1.36	140 - 2.54	29 - 0.72	52 - 0.98	86 - 1.44	112 - 2.69
		$D = 3$, HSDPA				$D = 3$, LTE			
$R = 1.00$	SSP-RR (FA)	84 - 1.74	111 - 2.81	188 - 4.51	279 - 7.06	75 - 1.89	107 - 3.29	161 - 5.07	232 - 8.08
	SSP-FL (FA)	90 - 1.00	126 - 2.54	194 - 4.44	291 - 6.82	86 - 0.90	116 - 2.95	171 - 5.05	240 - 7.82
	ASP-FT (AP)	90 - 0.75	142 - 1.34	207 - 2.37	298 - 3.71	86 - 0.76	132 - 1.50	185 - 2.69	254 - 4.18
$R = 0.50$	SSP-RR (FA)	46 - 0.62	87 - 0.99	151 - 1.92	230 - 3.83	44 - 0.66	78 - 1.00	128 - 1.81	178 - 3.96
	SSP-FL (FA)	46 - 0.61	86 - 0.96	152 - 1.77	237 - 3.41	44 - 0.65	78 - 0.95	129 - 1.64	183 - 3.40
	ASP-FT (AP)	46 - 0.70	87 - 0.96	152 - 1.45	238 - 2.42	44 - 0.79	78 - 1.04	129 - 1.48	186 - 2.45
$R = 0.33$	SSP-RR (FA)	31 - 0.59	58 - 0.91	103 - 1.65	167 - 3.15	29 - 0.64	52 - 0.94	87 - 1.59	129 - 3.06
	SSP-FL (FA)	31 - 0.59	58 - 0.90	103 - 1.62	167 - 3.02	29 - 0.61	52 - 0.92	87 - 1.53	128 - 2.85
	ASP-FT (AP)	31 - 0.66	58 - 0.93	103 - 1.43	167 - 2.33	29 - 0.74	52 - 1.00	87 - 1.46	129 - 2.35
		$D = 4$, HSDPA				$D = 4$, LTE			
$R = 1.00$	SSP-RR (FA)	75 - 1.72	156 - 2.17	191 - 4.01	356 - 5.84	66 - 1.98	133 - 2.40	167 - 4.40	301 - 6.03
	SSP-FL (FA)	83 - 1.31	163 - 1.63	199 - 3.89	372 - 5.51	76 - 1.45	151 - 1.44	183 - 4.16	312 - 5.68
	ASP-FT (AP)	90 - 0.74	163 - 1.12	246 - 1.99	372 - 3.31	86 - 0.78	151 - 1.12	217 - 2.10	312 - 3.45
$R = 0.50$	SSP-RR (FA)	46 - 0.64	87 - 1.08	152 - 2.03	246 - 3.87	44 - 0.67	79 - 1.04	130 - 1.87	192 - 3.39
	SSP-FL (FA)	46 - 0.62	87 - 1.05	152 - 1.94	245 - 3.80	44 - 0.66	79 - 1.00	129 - 1.77	192 - 3.20
	ASP-FT (AP)	46 - 0.73	87 - 1.04	152 - 1.63	245 - 2.77	44 - 0.84	79 - 1.13	130 - 1.65	192 - 2.66
$R = 0.33$	SSP-RR (FA)	31 - 0.61	58 - 1.02	103 - 1.86	170 - 3.53	29 - 0.65	53 - 1.01	87 - 1.75	130 - 3.16
	SSP-FL (FA)	31 - 0.61	58 - 0.99	104 - 1.82	170 - 3.51	29 - 0.62	53 - 0.97	87 - 1.69	130 - 3.04
	ASP-FT (AP)	31 - 0.69	58 - 1.01	104 - 1.59	170 - 2.69	29 - 0.77	53 - 1.05	87 - 1.60	130 - 2.61

by fixing a threshold K that is compared with $\delta = |\lambda_{i,j}^{ext}[u] - \lambda_{i,j}^{apr}[u]|$, namely if $\delta < K$, then $\lambda_{i,j}^{ext}[u]$ is not sent. The choice of K depends not only on the specific code considered but also on the quantization parameters used to represent $\lambda_{i,j}^{ext}[u]$ and on the performance loss in terms of Bit-Error-Rate (BER) that can be accepted. In the following we consider $N = 5114$ for HSDPA and $N = 6144$ for 3GPP-LTE respectively. In both cases the extrinsic information is represented on eight bits whereas the intrinsic information is represented on six bits with three fractional bits. Both decoders perform eight iterations ($I = 8$) with $P = 64$ using the Log-MAP algorithm [42] with a LUT-stored correction term. In Figs. 6 and 7 we show the BER performance for the HSDPA and 3GPP-LTE codes respectively obtained by applying the ABR technique described in

the previous paragraph with several values³ for K . In particular, in Fig. 6 we show for the HSDPA code that when $K > 10$ the performance worsens significantly. As an example, with $K = 10$ there is a performance loss of less than 0.1 dB in the waterfall region and nearly ideal performance when the code floors. On the other hand, with $K = 16$ the performance loss is of about 0.2 dB in the waterfall region and the code floor is shifted to higher SNR values of about 0.2 dB as well. Similar results were observed for the 3GPP-LTE code, so, for the sake of clarity, in Fig. 7 only results obtained with $K = 4, 6, 8, 10$ are shown. For

³Since we use three fractional bits for data representation the integer values of K we considered correspond to 0.25, 0.75, 1 and so on.

Figure 11 BER performance of the WiMAX $N = 1920$ turbo decoder with SL, BL, PFP representation and ABR technique, $K = 4, 6$.



both cases we obtained the corresponding best and average bandwidth reduction at different SNR values through Monte Carlo simulations⁴. Experimental results show that the throughput increase is significant when there is a high load on the network ($R = 1$) either using FL or RR input FIFO management. In particular, in Figs. 8, 9 and 10 we show the average throughput increase for the HSDPA turbo decoder for $D = 2, 3, 4$ respectively with different values of K . As it can be observed when $R = 1$ there is an average throughput increase, with respect to a decoder where ABR is not applied, that ranges from about 5 to 20 Mb/s for the HSDPA turbo decoder. Furthermore, we observed that in the best case there is a throughput increase of at least 60 Mb/s. On the other hand, when $R < 1$ the average throughput improvement is at most of 5 Mb/s. Similar results have been obtained for the LTE turbo decoder.

To complete the comparison, we show in Table 1 the throughput/area results for the HSDPA and LTE cases respectively, where the results for the HSDPA case with ASP-FT routing algorithm and AP node architecture are taken from [7]. For the sake of clarity we present only area results concerning the interconnection network, namely all the REs, and we suggest the reader to refer to specific papers as ASIC and ASIP implementations show significant differences in terms of complexity [9–11, 13–16, 20–23]. As it can be observed the significant throughput increase obtained

with the ABR technique on the FA node architecture when $R = 1$ is paid as an area overhead with respect to the AP node architecture. However, as pointed out in [7], the AP node architecture requires a large external memory to store the routing information. Moreover, the difference in terms of area between FA and AP node architectures reduces when $R < 1$. In particular, as shown in Table 1, when $R = 0.33$ with $P = 8$ and $P = 16$ the FA node architecture with the SSP-FL routing algorithm requires less area than the AP one.

4.3 WiMAX Case of Study

Simulation results shown in this section have been obtained with $N = 1920$, as in [38]. Each component of the extrinsic information is represented on eight bits whereas the intrinsic information is represented on six bits with two fractional bits. The decoder performs eight iterations ($I = 8$) with $P = 64$ using the Max-Log-MAP algorithm [42].

Since in binary turbo codes $\mathcal{U} = \{0, 1\}$, the LLR of the extrinsic information is a scalar value. On the other hand, for double-binary turbo codes $\mathcal{U} = \{00, 01, 10, 11\}$, as a consequence $\lambda_{i,j}^{ext}[u]$ is an array containing three elements. In [38], a bit level double-binary turbo decoder architecture is proposed to reduce the amount of memory to store the extrinsic information. The same idea is exploited in this work to reduce the area overhead of the NoC. Basically, a double-binary uncoded symbol u can be represented

⁴The worst case corresponds to simulations where ABR is not applied

as a couple of binary random variables AB . Then, with a slight abuse of notation, said X a binary random variable, we denote $X = 0$ with \bar{X} and $X = 1$ with X . Resorting to the Max-Log-MAP approximation we can convert Symbol-Level (SL) LLRs to Bit-Level (BL) LLRs as

$$\lambda_{i,j}^{ext}[A] \simeq \mu_A - \mu_{\bar{A}} \quad (14)$$

$$\lambda_{i,j}^{ext}[B] \simeq \mu_B - \mu_{\bar{B}} \quad (15)$$

where

$$\mu_A = \max \left\{ \lambda_{i,j}^{ext}[\bar{A}\bar{B}], \lambda_{i,j}^{ext}[AB] \right\} \quad (16)$$

$$\mu_{\bar{A}} = \max \left\{ 0, \lambda_{i,j}^{ext}[\bar{A}B] \right\} \quad (17)$$

$$\mu_B = \max \left\{ \lambda_{i,j}^{ext}[\bar{A}B], \lambda_{i,j}^{ext}[AB] \right\} \quad (18)$$

$$\mu_{\bar{B}} = \max \left\{ 0, \lambda_{i,j}^{ext}[A\bar{B}] \right\}. \quad (19)$$

Similarly, we can convert BL LLRs to SL LLRs with the following approximations.

$$1) \quad \lambda_{i,j}^{ext}[A] \geq 0 \text{ and } \lambda_{i,j}^{ext}[B] \geq 0$$

$$\lambda_{i,j}^{ext}[\bar{A}\bar{B}] \simeq \mu_{AB} - \lambda_{i,j}^{ext}[B] \quad (20)$$

$$\lambda_{i,j}^{ext}[\bar{A}B] \simeq \mu_{AB} - \lambda_{i,j}^{ext}[A] \quad (21)$$

$$\lambda_{i,j}^{ext}[AB] \simeq \mu_{AB} \quad (22)$$

$$2) \quad \lambda_{i,j}^{ext}[A] \geq 0 \text{ and } \lambda_{i,j}^{ext}[B] < 0$$

$$\lambda_{i,j}^{ext}[\bar{A}\bar{B}] \simeq \lambda_{i,j}^{ext}[A] \quad (23)$$

$$\lambda_{i,j}^{ext}[\bar{A}B] \simeq 0 \quad (24)$$

$$\lambda_{i,j}^{ext}[AB] \simeq \lambda_{i,j}^{ext}[A] + \lambda_{i,j}^{ext}[B] \quad (25)$$

Table 2 Throughput [Mb/s] - area SL [mm²] - area BL [mm²], PFP achieved with the WiMAX $N = 1920$ interleaver, with generalized Kautz topologies for $P \in \{8, 16, 32, 64\}$, $R \in \{0.33, 0.5, 1\}$, SSP-RR, SSP-FL and ASP-FT routing algorithms, no ABR.

		$D = 2$			
		$P = 8$	$P = 16$	$P = 32$	$P = 64$
$R = 1.00$	SSP-RR (FA)	104 - 2.15 - 1.46	138 - 3.61 - 2.43	195 - 5.16 - 3.51	264 - 6.97 - 4.85
	SSP-FL (FA)	105 - 2.17 - 1.47	144 - 3.40 - 2.30	208 - 4.57 - 3.13	285 - 6.11 - 4.29
	ASP-FT (AP)	105 - 1.62 - 0.92	144 - 2.54 - 1.43	208 - 3.42 - 1.99	285 - 4.61 - 2.79
$R = 0.50$	SSP-RR (FA)	86 - 0.47 - 0.38	127 - 1.48 - 1.07	176 - 3.20 - 2.26	231 - 5.33 - 3.80
	SSP-FL (FA)	86 - 0.42 - 0.35	134 - 1.19 - 0.88	187 - 2.74 - 1.96	246 - 4.60 - 3.33
	ASP-FT (AP)	86 - 0.42 - 0.35	134 - 1.00 - 0.69	187 - 2.15 - 1.37	246 - 3.56 - 2.29
$R = 0.33$	SSP-RR (FA)	58 - 0.39 - 0.33	102 - 0.78 - 0.62	153 - 1.94 - 1.45	199 - 4.05 - 2.98
	SSP-FL (FA)	58 - 0.36 - 0.31	103 - 0.69 - 0.57	161 - 1.55 - 1.20	209 - 3.41 - 2.56
	ASP-FT (AP)	58 - 0.38 - 0.33	103 - 0.67 - 0.54	161 - 1.33 - 0.99	209 - 2.73 - 1.89
		$D = 3$			
$R = 1.00$	SSP-RR (FA)	148 - 1.07 - 0.77	204 - 2.19 - 1.53	306 - 3.59 - 2.53	432 - 5.70 - 4.08
	SSP-FL (FA)	165 - 0.69 - 0.53	218 - 2.10 - 1.48	328 - 3.39 - 2.41	448 - 5.39 - 3.89
	ASP-FT (AP)	165 - 0.59 - 0.43	249 - 1.34 - 0.86	344 - 2.57 - 1.60	452 - 4.07 - 2.59
$R = 0.50$	SSP-RR (FA)	87 - 0.47 - 0.38	152 - 0.90 - 0.71	243 - 1.80 - 1.38	333 - 3.87 - 2.91
	SSP-FL (FA)	87 - 0.45 - 0.37	152 - 0.85 - 0.68	242 - 1.68 - 1.31	334 - 3.45 - 2.64
	ASP-FT (AP)	87 - 0.47 - 0.39	152 - 0.80 - 0.63	244 - 1.43 - 1.07	338 - 2.75 - 1.96
$R = 0.33$	SSP-RR (FA)	58 - 0.44 - 0.37	103 - 0.84 - 0.67	168 - 1.64 - 1.28	243 - 3.27 - 2.53
	SSP-FL (FA)	58 - 0.42 - 0.35	103 - 0.80 - 0.64	167 - 1.57 - 1.23	243 - 3.10 - 2.41
	ASP-FT (AP)	58 - 0.44 - 0.37	103 - 0.76 - 0.60	168 - 1.38 - 1.05	243 - 2.57 - 1.89
		$D = 4$			
$R = 1.00$	SSP-RR (FA)	135 - 1.24 - 0.88	253 - 1.79 - 1.29	323 - 3.41 - 2.45	513 - 5.38 - 3.93
	SSP-FL (FA)	153 - 0.94 - 0.69	279 - 1.41 - 1.05	344 - 3.21 - 2.33	533 - 5.09 - 3.76
	ASP-FT (AP)	166 - 0.63 - 0.46	279 - 1.15 - 0.80	393 - 2.28 - 1.51	533 - 3.99 - 2.66
$R = 0.50$	SSP-RR (FA)	87 - 0.50 - 0.41	155 - 0.92 - 0.73	247 - 1.98 - 1.53	354 - 3.83 - 2.94
	SSP-FL (FA)	87 - 0.48 - 0.39	155 - 0.90 - 0.72	248 - 1.89 - 1.47	356 - 3.70 - 2.84
	ASP-FT (AP)	87 - 0.52 - 0.43	155 - 0.87 - 0.69	249 - 1.66 - 1.24	356 - 3.12 - 2.27
$R = 0.33$	SSP-RR (FA)	58 - 0.47 - 0.39	104 - 0.92 - 0.73	169 - 1.85 - 1.45	248 - 3.61 - 2.80
	SSP-FL (FA)	58 - 0.44 - 0.36	104 - 0.88 - 0.70	169 - 1.75 - 1.37	248 - 3.45 - 2.67
	ASP-FT (AP)	58 - 0.48 - 0.40	104 - 0.84 - 0.66	169 - 1.57 - 1.19	248 - 2.97 - 2.19

$$3) \quad \lambda_{i,j}^{ext}[A] < 0 \text{ and } \lambda_{i,j}^{ext}[B] \geq 0$$

$$\lambda_{i,j}^{ext}[A\bar{B}] \simeq 0 \quad (26)$$

$$\lambda_{i,j}^{ext}[\bar{A}B] \simeq \lambda_{i,j}^{ext}[B] \quad (27)$$

$$\lambda_{i,j}^{ext}[AB] \simeq \lambda_{i,j}^{ext}[A] + \lambda_{i,j}^{ext}[B] \quad (28)$$

$$4) \quad \lambda_{i,j}^{ext}[A] < 0 \text{ and } \lambda_{i,j}^{ext}[B] < 0$$

$$\lambda_{i,j}^{ext}[A\bar{B}] \simeq \lambda_{i,j}^{ext}[A] \quad (29)$$

$$\lambda_{i,j}^{ext}[\bar{A}B] \simeq \lambda_{i,j}^{ext}[B] \quad (30)$$

$$\lambda_{i,j}^{ext}[AB] \simeq \lambda_{i,j}^{ext}[A] + \lambda_{i,j}^{ext}[B] - \mu_{AB} \quad (31)$$

where

$$\mu_{AB} = \max\{\lambda_{i,j}^{ext}[A], \lambda_{i,j}^{ext}[B]\} \quad (32)$$

For further details on bit to symbol and symbol to bit conversion the reader can refer to [38].

The use of BL LLRs introduces a BER performance loss of about 0.2 dB (see Fig. 11), but it reduces the data width of one third with respect to SL LLRs, as the payload of each packet contains $\lambda_{i,j}^{ext}[A]$ and $\lambda_{i,j}^{ext}[B]$ instead

of $\lambda_{i,j}^{ext}[u]$. To further reduce the data width we applied to BL LLRs the Pseudo-Floating-Point (PFP) representation suggested in [37]. As highlighted also in [66, 67] the most significant bits of the extrinsic information play an important role in the decoding procedure. Indeed, the basic idea is to analyze the binary representation of $\lambda_{i,j}^{ext}[A]$ and $\lambda_{i,j}^{ext}[B]$ (as 2's complement values) from the most significant bit to the least significant bit and to detect the first zero-one or one-zero transition, which represents the starting bit of the extrinsic information significand. We denote the significand as ξ and the number of bits that prefix ξ are coded as a shift index σ . Thus, for each couple $\lambda_{i,j}^{ext}[A]$, $\lambda_{i,j}^{ext}[B]$ we obtain $\xi_{i,j}[A]$, $\xi_{i,j}[B]$, $\sigma_{i,j}[A]$ and $\sigma_{i,j}[B]$. Then, according with [37], we impose $\sigma_{i,j} = \min\{\sigma_{i,j}[A], \sigma_{i,j}[B]\}$. Said n_λ , n_ξ and n_σ the number of bits to represent λ , ξ and σ respectively we obtain

$$\tilde{\xi}_{i,j}[A] = \lambda_{i,j}^{ext}[A] \gg (n_\lambda - n_\xi - \sigma_{i,j}) \quad (33)$$

$$\tilde{\xi}_{i,j}[B] = \lambda_{i,j}^{ext}[B] \gg (n_\lambda - n_\xi - \sigma_{i,j}) \quad (34)$$

where \gg stands for arithmetic right shift. As a consequence, the payload of each packet sent on the network now contains $\tilde{\xi}_{i,j}[A]$, $\tilde{\xi}_{i,j}[B]$ and $\sigma_{i,j}$ instead of $\lambda_{i,j}^{ext}[u]$.

Figure 12 Average throughput improvement at different SNR values of the WiMAX $N = 1920$ turbo decoder with $K = 4, 6$ on generalized Kautz networks $D = 2$ and $P = 64$.

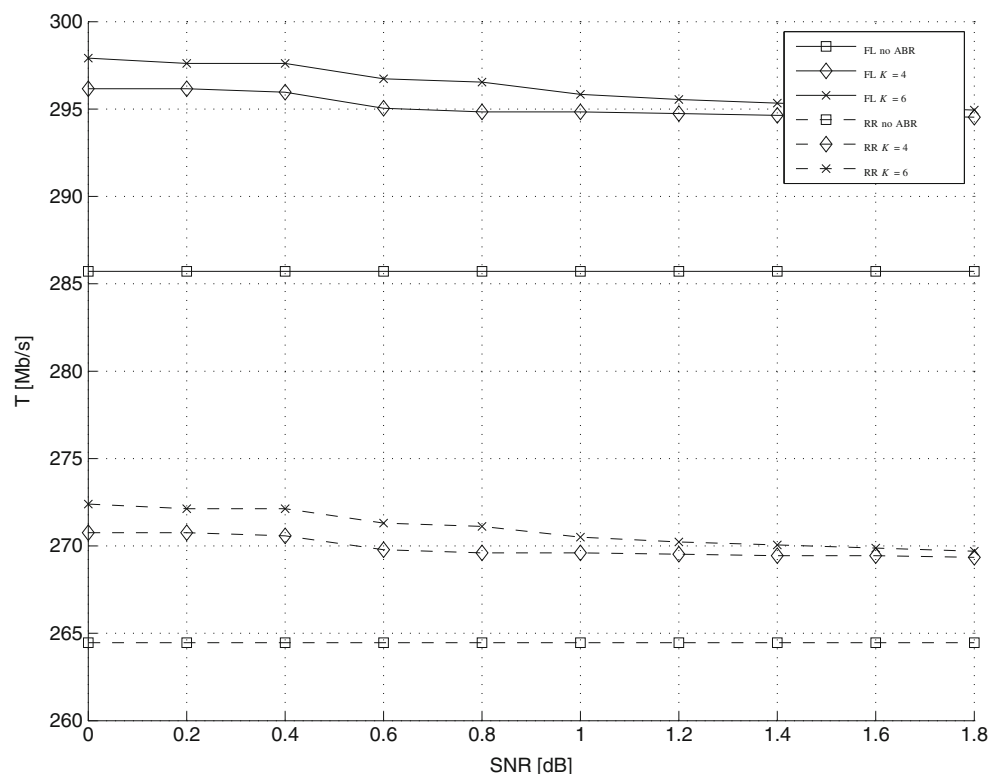
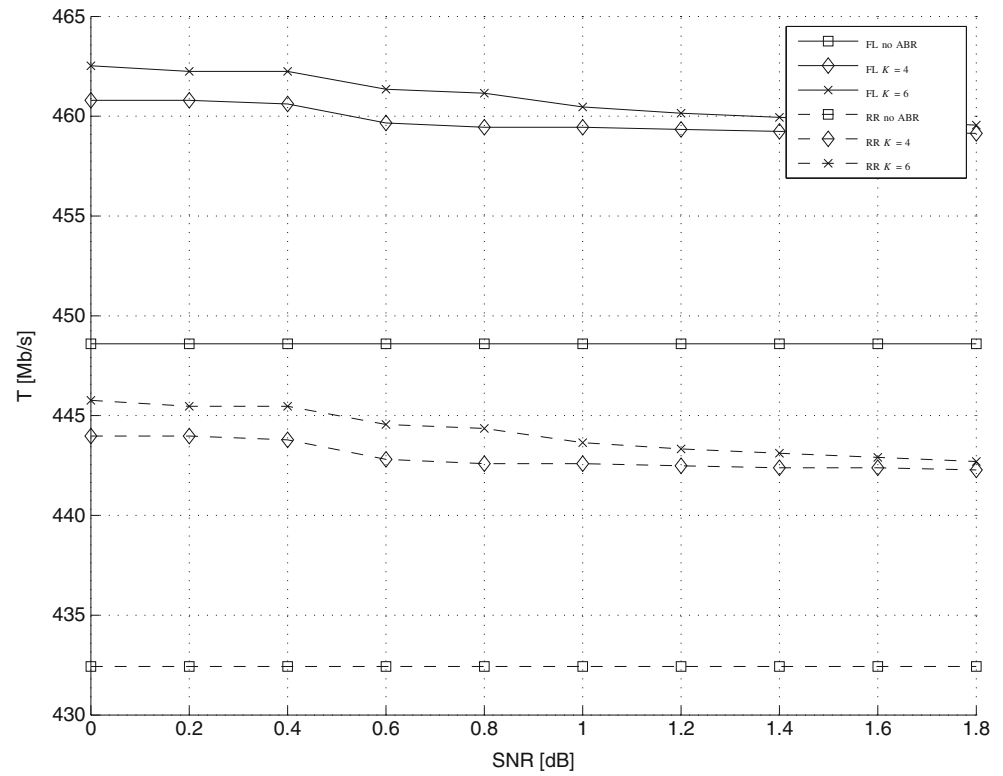


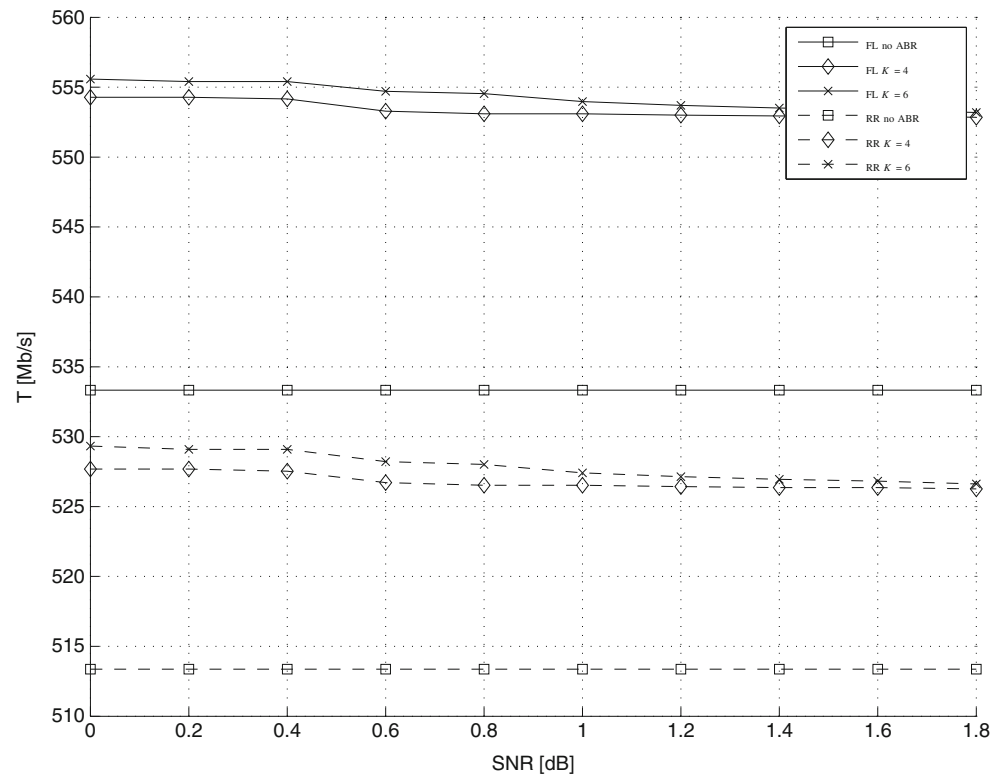
Figure 13 Average throughput improvement at different SNR values of the WiMAX $N = 1920$ turbo decoder with $K = 4, 6$ on generalized Kautz networks $D = 3$ and $P = 64$.



As stated in the first paragraph of this section $n_\lambda = 8$. Thus, said n_d the number of bits devoted to represent the extrinsic information in the payload we have: i) $n_d = 3n_\lambda =$

24 for $\lambda_{i,j}^{ext}[u]$ and ii) $n_d = 2n_\lambda = 16$ for $\lambda_{i,j}^{ext}[A]$ and $\lambda_{i,j}^{ext}[B]$. If we impose $n_\xi = 4$, we obtain $\sigma_{i,j} \leq 4$ and so $n_\sigma = 3$, leading to $n_d = 2n_\xi + n_\sigma = 11$ that is less than half

Figure 14 Average throughput improvement at different SNR values of the WiMAX $N = 1920$ turbo decoder with $K = 4, 6$ on generalized Kautz networks $D = 4$ and $P = 64$.



the value of n_d for $\lambda_{i,j}^{ext}[u]$. As shown in Fig. 11 the BER performance loss of BL, PFP LLR representation, is nearly the same as the fixed point BL one. In Table 2 the throughput and area results obtained by using SL and BL, PFP LLR representation are shown for generalized Kautz topologies. As it can be observed, the area decrease as a function of n_d is not linear, however, it becomes particularly interesting when $R = 1$. As an example, with $R = 1$, $D = 4$ and $P = 64$ there is an area saving of up to the 40 %.

The techniques described in the previous paragraphs are all aimed at reducing the area of the NoC-based turbo decoder. Furthermore, the ABR technique described in Section 4.1 can be used to improve the throughput as well. In order to limit the BER performance loss introduced by the ABR technique, we employ the SL reliability criterion proposed in [36] but we send BL, PFP extrinsic information when the criterion is not met. The ABR technique we used is summarized in Algorithm 1 and can be summarized as follows: said $\vartheta_{i,j}^{apr}$, $\varrho_{i,j}^{apr}$ and $\vartheta_{i,j}^{ext}$, $\varrho_{i,j}^{ext}$ the first and the

Algorithm 1 SL reliability criterion proposed in [36]

```

1:  $\vartheta_{i,j}^{apr} \leftarrow \max \{ \lambda_{i,j}^{apr}[u] \}$ 
2:  $\varrho_{i,j}^{apr} \leftarrow \max \{ \lambda_{i,j}^{apr}[u] \setminus \vartheta_{i,j}^{apr} \}$ 
3:  $\vartheta_{i,j}^{ext} \leftarrow \max \{ \lambda_{i,j}^{ext}[u] \}$ 
4:  $\varrho_{i,j}^{ext} \leftarrow \max \{ \lambda_{i,j}^{ext}[u] \setminus \vartheta_{i,j}^{ext} \}$ 
5:  $\Delta_{i,j}^{apr} \leftarrow |\vartheta_{i,j}^{apr} - \varrho_{i,j}^{apr}|$ 
6:  $\Delta_{i,j}^{ext} \leftarrow |\vartheta_{i,j}^{ext} - \varrho_{i,j}^{ext}|$ 
7:  $\Phi_{i,j} \leftarrow |\Delta_{i,j}^{ext} - \Delta_{i,j}^{apr}|$ 
8: if  $\Phi_{i,j} < K$  then
9:   do not send any packet
10: else
11:   send  $\tilde{\xi}_{i,j}[A]$ ,  $\tilde{\xi}_{i,j}[B]$ ,  $\sigma_{i,j}$ 
12: endif

```

second maximum values in $\lambda_{i,j}^{apr}[u]$ and $\lambda_{i,j}^{ext}[u]$ respectively, we compute $\Delta_{i,j}^{apr} = |\vartheta_{i,j}^{apr} - \varrho_{i,j}^{apr}|$ and $\Delta_{i,j}^{ext} = |\vartheta_{i,j}^{ext} - \varrho_{i,j}^{ext}|$; finally, we compare $\Phi_{i,j} = |\Delta_{i,j}^{ext} - \Delta_{i,j}^{apr}|$ with the threshold K .

As shown in Fig. 11 the BER performance loss introduced by the ABR technique is negligible. Moreover, as shown in Figs. 12, 13 and 14 when $R = 1$ the ABR technique induces an average throughput increase of about 5 to 20 Mb/s. Similarly to the binary codes in the best case the throughput improvement is at least of more than 40 Mb/s, whereas when $R < 1$ the average throughput improvement is at most of 5 Mb/s.

5 Conclusions

In this work ABR techniques have been exploited to improve the throughput of NoC-based turbo decoder architectures. When the load of the network is high the average throughput is improved of about 5 to 20 Mb/s and in the best case the throughput is increased of more than 60 Mb/s and 40 Mb/s for binary and double-binary codes respectively. Moreover, the area required to support double-binary codes has been significantly reduced (up to more than the 40 %) by applying BL, PFP representation of the extrinsic information with a BER performance loss of about 0.2 dB.

References

1. IEEE Std 802.16, part 16: air interface for fixed broadband wireless access systems, Oct. 2004.
2. TS 36.212 v8.0.0: Multiplexing and Channel Coding (FDD) (Release 8), 2007-09.
3. Berrou, C., Glavieux, A., Thitimajshima, P. (1993). Near Shannon limit error correcting coding and decoding: turbo codes. In *IEEE international conference on communications* (pp. 1064–1070).
4. Gallager, R.G. (1962). Low density parity check codes. *IRE Transactions on Information Theory*, IT-8(1), 21–28.
5. Boutillon, E., Douillard, C., Montorsi, G. (2007). Iterative decoding of concatenated convolutional codes: implementation issues. *Proceedings of the IEEE*, 95(6), 1201–1227.
6. Guilloud, F., Boutillon, E., Tusch, J., Danger, J.L. (2007). Generic description and synthesis of LDPC decoders. *IEEE Transactions on Communications*, 55(11), 2084–2091.
7. Martina, M., Masera, G., Moussa, H., Baghdadi, A. (2011). On chip interconnects for multiprocessor turbo decoding architectures. *Elsevier Microprocessors and Microsystems*, 35(2), 167–181.
8. Polydoros, A. (2008). Algorithmic aspects of radio flexibility. In *IEEE international symposium on personal, indoor and mobile communications* (pp. 1–5).
9. Vogt, T., & Wehn, N. (2008). Reconfigurable ASIP for convolutional and turbo decoding in an SDR environment. *IEEE Transactions on VLSI*, 16(10), 1309–1320.
10. Bougard, B., Priewasser, R., der Perre, L.V., Huemer, M. (2008). Algorithm-architecture co-design of a multi-standard FEC decoder ASIP. In *ICT mobile summit conference*.
11. Martina, M., Nicola, M., Masera, G. (2008). A flexible UMTS-WiMax turbo decoder architecture. *IEEE Transactions on Circuits and Systems II*, 55(4), 369–373.
12. Rovini, M., Gentile, G., Fanucci, L. (2009). A flexible state-metric recursion unit for a multi-standard BCJR decoder. In *IEEE international conference on signals circuits and systems* (pp. 1–6).
13. Kim, J.H., & Park, I.C. (2009). A unified parallel radix-4 turbo decoder for mobile WiMAX and 3GPP-LTE. In *IEEE custom integrated circuits conference* (pp. 487–490).
14. Muller, O., Baghdadi, A., Jezequel, M. (2009). From parallelism levels to a multi-ASIP architecture for turbo decoding. *IEEE Transactions on VLSI*, 17(1), 92–102.

15. Reddy, P., Alkhayat, R., Clermidy, F., Baghdadi, A., Jezequel, M. (2010). Power consumption analysis and energy efficient optimization for turbo decoder implementation. In *International symposium on system-on-chip* (pp. 12–17).
16. Inseher, T., May, M., Wehn, N. (2010). A multi-mode 3GPP-LTE/HSDPA turbo decoder. In *IEEE international conference on communication systems* (pp. 336–340).
17. Gentile, G., Rovini, M., Fanucci, L. (2010). A multi-standard flexible turbo/LDPC decoder via ASIC design. In *International symposium on turbo codes & iterative information processing* (pp. 294–298).
18. Sun, Y., & Cavallaro, J.R. (2010). A flexible LDPC/turbo decoder architecture. *Journal of Signal Processing Systems*, 64(1), 1–16.
19. Murugappa, P., Al-Khayat, R., Baghdadi, A., Jezequel, M. (2011). A flexible high throughput multi-ASIP architecture for LDPC and turbo decoding. In *Design, automation and test in Europe conference and exhibition* (pp. 1–6).
20. Shin, M.C., & Park, I.C. (2007). SIMD processor-based turbo decoder supporting multiple third-generation wireless standards. *IEEE Transactions on VLSI*, 15(7), 801–810.
21. Sun, Y., Zhu, Y., Goel, M., Cavallaro, J.R. (2008). Configurable and scalable high throughput turbo decoder architecture for multiple 4G wireless standards. In *IEEE international conference on application-specific systems, architectures and processors* (pp. 209–214).
22. Lin, C.H., Chen, C.Y., Wu, A.Y. (2011). Area efficient scalable MAP processor design for high-throughput multistandard convolutional turbo decoding. *IEEE Transactions on VLSI*, 19(2), 305–318.
23. Al-Khayat, R., Murugappa, P., Baghdadi, A., Jezequel, M. (2011). Area and throughput optimized ASIP for multi-standard turbo decoding. In *IEEE international symposium on rapid system prototyping* (pp. 79–84).
24. Hoffmann, A., Schliebusch, O., Nohl, A., Braun, G., Meyr, H. (2001). A methodology for the design of application specific instruction set processors (ASIP) using the machine description language LISA. In *International conference on computer-aided design*.
25. Neeb, C., Thul, M.J., Wehn, N. (2005). Network-on-chip-centric approach to interleaving in high throughput channel decoders. In *IEEE international symposium on circuits and systems* (pp. 1766–1769).
26. Moussa, H., Muller, O., Baghdadi, A., Jezequel, M. (2007). Butterfly and Benes-based on-chip communication networks for multiprocessor turbo decoding. In *Design, automation and test in Europe conference and exhibition* (pp. 654–659).
27. Moussa, H., Baghdadi, A., Jezequel, M. (2008). Binary de Bruijn interconnection network for a flexible LDPC/turbo decoder. In *IEEE international symposium on circuits and systems* (pp. 97–100).
28. Martina, M., & Masera, G. (2010). Turbo NOC: a framework for the design of network on chip based turbo decoder architectures. *IEEE Transactions on Circuits and Systems I*, 57(10), 2776–2789.
29. Wang, G., Sun, Y., Cavallaro, J.R., Guo, Y. (2011). High-throughput contention-free concurrent interleaver architecture for multi-standard turbo decoder. In *IEEE international conference on application-specific systems, architectures and processors* (pp. 113–121).
30. Vacca, F., Moussa, H., Baghdadi, A., Masera, G. (2009). Flexible architectures for LDPC decoders based on network on chip paradigm. In *Euromicro conference on digital system design* (pp. 582–589).
31. Guerrier, P., & Greiner, A. (2000). A generic architecture for on-chip packet-switched interconnections. In *Design, automation and test in Europe conference and exhibition* (pp. 250–256).
32. Dally, W.J., & Towels, B. (2001). Route packets, not wires: On-chip interconnection networks. In *Design automation conference* (pp. 684–689).
33. Benini, L., & Micheli, G.D. (2002). Networks on chips: a new soc paradigm. *IEEE Computer*, 35(1), 70–78.
34. Kumar, S., Jantsch, A., Soininen, J.P., Forsell, M., Millberg, M., Oberg, J., Tiensyrja, K., Hemani, A. (2002). A network on chip architecture and design methodology. In *IEEE computer society annual symposium on VLSI* (pp. 105–112).
35. Awais, M., & Condo, C. (2012). Flexible LDPC decoder architectures. *VLSI Design, 2012*, 1–16.
36. Muller, O., Baghdadi, A., Jezequel, M. (2006). Bandwidth reduction of extrinsic information exchange in turbo decoding. *IET Electronics Letters*, 42(19), 1104–1105.
37. Park, S.M., Kwak, J., Lee, K. (2008). Extrinsic information memory reduced architecture for non-binary turbo decoder implementation. In *IEEE vehicular technology conference* (pp. 539–543).
38. Kim, J.H., & Park, I.C. (2009). Bit-level extrinsic information exchange method for double-binary turbo codes. *IEEE Transactions on Circuits and Systems II*, 56(1), 81–85.
39. Berrou, C., Jezequel, M., Douillard, C., Kerouedan, S. (2001). The advantages of non-binary turbo codes. In *IEEE information theory workshop* (pp. 61–63).
40. Bahl, L.R., Cocke, J., Jelinek, F., Raviv, J. (1974). Optimal decoding of linear codes for minimizing symbol error rate. *IEEE Transactions on Information Theory*, 20(3), 284–287.
41. Robertson, P., Villebrun, E., Hoeher, P. (1995). A comparison of optimal and sub-optimal MAP decoding algorithms operating in the Log domain. In *IEEE ICC* (pp. 1009–1013).
42. Robertson, P., Hoeher, P., Villebrun, E. (1997). Optimal and sub-optimal maximum a posteriori algorithms suitable for turbo decoding. *European Transactions on Telecommunications*, 8(2), 119–125.
43. Papaharalabos, S., Mathiopoulou, P.T., Masera, G., Martina, M. (2009). On optimal and near-optimal turbo decoding using generalized max* operator. *IEEE Communications Letters*, 13(7), 522–524.
44. Flich, J., & Duato, J. (2008). Logic-based distributed routing for NoCs. *IEEE Computer Architecture Letters*, 7(1), 13–16.
45. Shi, Z., Yang, Y., Zeng, X., Yu, Z. (2011). A reconfigurable and deadlock-free routing algorithm for 2D Mesh Network-on-Chip. In *IEEE international symposium of circuits and systems* (pp. 2934–2937).
46. Moraveji, R., Sarbazi-Azad, H., Zomaya, A.Y. (2010). A general methodology for direction-based irregular routing algorithms. *Journal of Parallel and Distributed Computing*, 70(4), 363–370.
47. <http://www.3gpp2.org>. 2004.
48. Wang, Z., & Li, Q. (2007). Very low-complexity hardware interleaver for turbo decoding. *IEEE Transactions on Circuits and Systems II*, 54(7), 636–640.
49. Martina, M., Nicola, M., Masera, G. (2008). Hardware design of a low complexity, parallel interleaver for wimax duo-binary turbo decoding. *IEEE Communications Letters*, 12(11), 846–848.
50. Martina, M. (2012). Turbo NOC: Network On Chip based turbo decoder architectures, downloadable at <http://personal.delen.polito.it/maurizio.martina/turbo.html>.
51. Bang-Jensen, J., & Gutin, G. (2008). *Digraphs, theory, algorithms and applications* (2nd ed.) London: Springer-Verlag.

52. Imase, M., & Itoh, M. (1981). Design to minimize diameter on building-block network. *IEEE Transactions on Computers*, 30(6), 439–442.
53. Imase, M., & Itoh, M. (1983). A design for directed graphs with minimum diameter. *IEEE Transactions on Computers*, 32(8), 782–784.
54. Sabbaghi-Nadooshan, R., & Sarbazi-Azad, H. (2008). The kautz mesh: a new topology for socs. In *IEEE international SoC design conference* (pp. 300–303).
55. Sabbaghi-Nadooshan, R. (2011). Kautz mesh topology for on-chip networks. *Journal of Computing*, 3(2), 33–40.
56. Pulimeno, A., Graziano, M., Piccinini, G. (2012). UDSM trends, comparison: From technology roadmap to UltraSparc Niagara2. *IEEE Transactions on VLSI Systems*, 20(7), 1341–1346.
57. Angiolini, F., Meloni, P., Carta, S.M., Raffo, L., Benini, L. (2007). A layout-aware analysis of networks-on-chip and traditional interconnects for MPSoCs. *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems*, 26(3), 421–434.
58. Meloni, P., Loi, I., Angiolini, F., Carta, S., Barbaro, M., Raffo, L., Benini, L. (2007). Area and power modeling for networks-on-chip with layout awareness. *VLSI Design*. special issue on Networks on Chip, Article ID 50285, 12 pages.
59. Hosseinabady, M., Kakoei, M.R., Mathew, J., Pradhan, D.K. (2007). Reliable network-on-chip based on generalized de Bruijn graph. In *IEEE international high level design validation and test workshop* (pp. 3–10).
60. Hosseinabady, M., Kakoei, M.R., Mathew, J., Pradhan, D.K. (2008). De Bruijn graph as a low latency scalable architecture for energy efficient massive NoCs. In *Design, automation and test in Europe conference and exhibition* (pp. 1370–1373).
61. Samatham, M.R., & Pradhan, D.K. (1989). The De Bruijn multiprocessor network: a versatile parallel processing and sorting network for VLSI. *IEEE Transactions on Computers*, 38(4), 567–581.
62. Cheng, C.C., Tsai, Y.M., Chen, L.G., Chandrakasan, A.P. (2010). A 0.077 to 0.168 nJ/bit/iteration scalable 3GPP LTE turbo decoder with an adaptive sub-block parallel scheme and an embedded DVFS engine. In *IEEE custom integrated circuits conference* (pp. 1–4).
63. Gentile, G., Rovini, M., Fanucci, L. (2010). Low-power techniques for flexible channel decoders. In *International conference on applied electronics* (pp. 1–4).
64. Reddy, P., Clermidy, F., Baghdadi, A., Jezequel, M. (2011). A low complexity stopping criterion for reducing power consumption in turbo decoders. In *Design, automation and test in Europe conference and exhibition* (pp. 1–6).
65. Kim, D.H., & Kim, S.W. (2006). Bit-level stopping of turbo decoding. *IEEE Communications Letters*, 10(3), 183–185.
66. Vogt, J., Ertel, J., Finger, A. (2000). Reducing bit width of extrinsic memory in turbo decoder realisations. *IEE Electronics Letters*, 36(20), 1714–1716.
67. Singh, A., Boutillon, E., Masera, G. (2008). Bit-width optimization of extrinsic information in turbo decoder. In *International symposium on turbo codes & related topics* (pp. 134–138).



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 assistant professor at the VLSI Lab, Politecnico di Torino. His research activities include VLSI design and implementation of architectures for digital signal processing and communications.



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.