Reducing the dissipated energy in multi-standard turbo and LDPC decoders

Circuits, Systems & Signal Processing manuscript No.
(will be inserted by the editor)

Reducing the dissipated energy in multi-standard turbo and LDPC decoders

Carlo Condo · Amer Baghdadi · Guido Masera

the date of receipt and acceptance should be inserted later

Abstract Parallel Low-Density Parity-Check and turbo code decoding consists of iterative processes that rely on the exchange of messages among multiple processing elements (PEs). They are characterized by complex communication patterns that require area expensive interconnect and memory management. Channel decoders based on Networks-on-Chip (NoCs) have been proposed in the literature, showing unmatched degrees of flexibility, but yielding high area occupation and power consumption. While general and application-specific power reduction techniques are available to save energy, the gap with respect to dedicated decoders is still large. This paper proposes techniques that reduce and optimize the traffic on the network for NoC-based channel decoders, and can be applied to any NoC architecture. The proposed techniques exploit the probabilistic nature and the processing order of the exchanged messages in

Carlo Condo, Guido Masera
Politecnico di Torino
Corso Duca degli Abruzzi 24
10129 Torino, Italy
E-mail: ¡name.surname¿@polito.it

Amer Baghdadi
Telecom Bretagne
Technopole Brest-Iroise
29238 Brest Cedex 3, France
E-mail: ¡name.surname¿@telecom-bretagne.eu

the iterative decoding and define novel importance and urgency metrics. Given a target throughput, these techniques allow to consistently reduce and optimize the NoC traffic with minor or no bit error rate (BER) degradation with respect to a decoder with no traffic optimization. An already available NoC based decoder enhanced with the proposed traffic shaping techniques leads to 13.1% area overhead and 15.0% power and energy reduction, while 40.2% of power is saved on the NoC alone.

Keywords LDPC · turbo · decoder · low power

1 Introduction

Low-Density Parity-Check (LDPC) codes and turbo codes are used in a wide range of applications, like mobile, wireless and wired communication, deep space and satellite links, TV broadcasting, magnetic storage, flash memories. Quite a large number of hardware implementations are available in the open literature for turbo and LDPC decoders and most of them are based on parallel architectures, made of several interconnected Processing Elements (PEs). The capabilities of these decoders in terms of supported codes strongly depend on the structure of the interconnect network among the PEs. Simple and efficient structures are used to support families of homogeneous codes, like the ones included in a single or limited number of communication standards. However, the design of a decoder with larger versatility, able to support both turbo and LDPC codes in a wide set of standards requires the allocation of more complex and flexible networks. A possible solution towards the implementation of higly flexible channel decoders is given by Networks-on-Chip (NoCs), which were originally devised for general purpose System-On-Chip (SoC), with multiple applications are mapped on a number of PEs. They potentially guarantee very high flexibility and better scalability with respect to traditional bus based

solutions. Lately, the original idea of general purpose NoCs has evolved and new kinds of NoCs are today proposed with features fully optimized for a single or reduced number of applications (Application Specific NoCs, or ASNoCs, [11]). LDPC and turbo decoders based on very special ASNoCs have been proposed, providing a high degree of flexibility [14, 21, 22, 27, 29, 30, 32]. These implementations support multiple standards and code types, with dynamic switching capabilities between codes.

In general, ASNoCs proposed for channel decoding are *intra-IP* NoCs that connect homogeneous PEs; in order to reduce occupied area and dissipated power, they are also characterized by lower complexity and limited capabilities with respect to NoCs usually reported for other applications. For example, it is shown in [27] that the best choices in terms of NoC topology and routing algorithm for a NoC-based turbo decoder are given by Kautz topology, which is less regular than usual 2D-mesh but guarantees shorter delivery time, and shortest path algorithm, which can be implemented with a very low complexity. In spite of these choices the NoC is responsible for a relevant efficiency gap between NoC-based highly flexible decoders and dedicated or partially flexible solutions, such as for example implementations in [33,35]: the NoC guarantees virtual connectivity among all nodes and great flexibility, but packet latencies and intermediate storage, along with routing logic and memories, increase power consumption and decrease throughput with respect to dedicated and partially flexible decoders.

General power reduction techniques, like clock gating and dynamic voltage scaling, can be applied to channel decoders. Additionally, specific features of LDPC and turbo decoding can be exploited to reduce power dissipation. As an example, since LDPC and turbo decoding are iterative processes, early stopping of iterations criteria have been proposed over the years [23,25]: these techniques rely on the observation of a metric to decide if it is worth or not

to perform additional iterations, avoiding unnecessary energy consumption. The usage of power reduction techniques is in many cases very effective, however, since they can be introduced in any decoding architecture, the power consumption gap between NoC-based decoders and dedicated architectures is still large.

This paper proposes new power reduction techniques for flexible channel decoders. These techniques reduce and optimize the traffic due to messages exchanged among PEs, which account for a significant percentage of the consumed energy. Therefore the proposed methods are particularly effective with NoC-based decoders. **Preliminary contributions in the direction of traffic reduction have been made in [31] for turbo codes, and in [15] for LDPC codes, both dealing with the evaluation of the usefulness of exchanged information: this work refines and extends them, while proposing new traffic optimization techniques.** The performance of the proposed techniques has been extensively evaluated and compared to alternative methods, while the most promising one has been implemented as an application example on a fully characterized decoder [14]. Area overhead and power gain have been obtained and compared to the state of the art to evaluate the effectiveness of the solution. Both multi-standard decoders [6, 19] and single-standard, optimized implementations [16, 34] are taken in account. The comparisons show how the proposed solutions greatly improve the energy efficiency of NoC-decoders and help to reduce the gap between flexible and dedicated implementations. For example, the proposed multi-standard LDPC/turbo decoder consumes 99.2 mW, against the 51.6 mW of the decoder in [16], regardless of its support being limited to WiMAX LDPC codes only and being optimized for ultra low power performance.

The rest of the paper is organized as follows: Section 2 introduces LDPC and turbo decoding and analyzes the problems arising in parallel implemen-

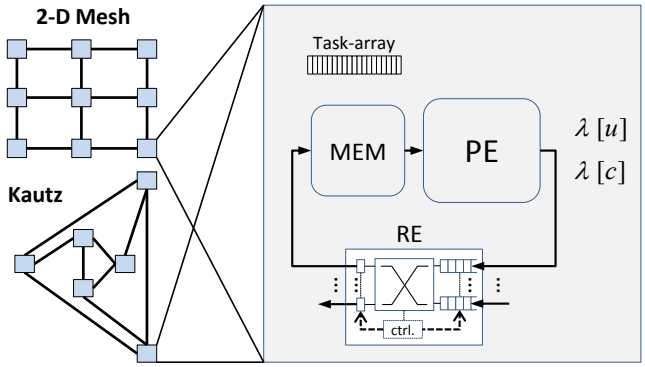

Fig. 1 NoC-based parallel decoder structure

tation. Section 3 proposes different solutions to these problems, while their performance is compared against alternative approaches in Section 4. Section 5 describes the hardware architectures of the proposed techniques, and in Section 6 the implementation results of the best-performing method are compared to the state of the art. Conclusions are drawn in Section 7.

2 NoC-based decoding: practical issues

LDPC codes [18] are described by a sparse matrix with M rows and N columns. Each word x that satisfies $\mathbf{H} \cdot x = 0$ is considered as valid codeword. LDPC decoding is based on the Tanner graph representation of $\mathbf{H}$, composed of Variable Nodes (VNs, the columns of $\mathbf{H}$) and Check Nodes (CNs, the rows of $\mathbf{H}$). The *Belief Propagation* (BP) algorithm is the most common algorithm for LDPC decoding, especially with the efficient layered scheduling [20]. In a layered decoder, parity-check constraints are grouped in layers of unconnected $\mathbf{H}$ rows: extrinsic information is passed from one layer to a subsequent one [20], reiterating the process up to the desired level of reliability. Let the Logarithmic Likelihood Ratio (LLR) of bit c in column k of $\mathbf{H}$ be represented with $\lambda_k[c]$, and initialized to the corresponding received soft value. For all parity

constraints l in a given layer, VN k receive the extrinisic information $\lambda_k^{old}[c]$ from the previous layer, and produce an updated version $\lambda_k^{new}[c]$, sent to the following layer. The update is based on the metric R_{lk}^{new}, computed by CN l and stored for usage as R_{lk}^{old} in the next iteration.

Convolutional turbo codes are typically specified as the parallel concatenation of two Convolutional Code (CC) encoders. The decoder is consequently made of two different Soft-In-Soft-Out (SISO) decoders, linked by an interleaver Π and a de-interleaver Π^{-1}. Each SISO decoder relies on the BCJR algorithm [8]. With each CC represented with a trellis, let k be a trellis step and u an uncoded symbol. Each decoder computes

$$\lambda_k[u] = \lambda_k^{apo}[u] - \lambda_k^{apr}[u] - \lambda_k[\mathbf{c}^u] \tag{1}$$

where $\lambda_k^{apo}[u]$ and $\lambda_k^{apr}[u]$ are the *a-posteriori* and *a priori* information respectively, and the systematic component of the intrinsic information is represented by $\lambda_k[\mathbf{c}^u]$.

In parallel decoders, the decoding process of the received frame is typically partitioned among P PEs. Messages are exchanged among PEs by means of an interconnection structure, that is usually deterministic, and guarantees fixed and uniform latency. To increase the degree of flexibility of the decoder, recent works have proposed NoCs as interconnection structures [14, 21, 27, 30]. **Different techniques are available in the state of the art to partition the received frame and the decoding process among PEs: they all rely on the assignment of a set of variable nodes (in the LDPC case) and trellis steps (in the turbo case) to each processing element. Even though random partitioning has been proven to grant acceptable results, a graph coloring approach that minimizes the inter-PE communication has been employed in this case [15].** Fig. 1

shows the basic structure of a NoC-based decoder. While in turbo decoders the PEs are concurrent SISOs executing (1) on different sub-blocks, in LDPC decoders each PE updates the LLRs involved in a certain set of parity check constraints. Consequently, the task array of each PE, i.e. the sequence of processes to be performed within an iteration, is a continuous set of trellis steps in the turbo case, and a uniform selection of parity checks from all $\mathbf{H}$ layers in the LDPC case. Each PE is connected to a Routing Element (RE), which in turn is linked to a number of other routers, with input buffers at every port. If every RE has an attached PE the NoC has a direct topology (like the 2-D mesh and the generalized Kautz [24] shown in Fig.1), while in indirect NoCs (e.g. Benes [10]) some REs are only used as intermediate communication nodes. The NoC traffic is constituted of $\lambda_k[u]$ and $\lambda_k[c]$ values for turbo and LDPC codes respectively, as shown in Fig. 1: messages are injected in the NoC directly by the PEs, while information received from the channel is stored in a memory for further use. The communication pattern with a NoC is deterministic, but the delays introduced are nonuniform and can vary consistently from code to code and with time. This nonuniform distribution of delays is basically due to the uneven distance among PEs. Choices like number of PEs, NoC topoly ans routing algorithm have a significant impact on achievable throughput and implementation complexity, as extensively studied in [14, 27]. In general, a NoC used to support turbo and LDPC decoding is a particular type of NoC, characterized by very low complexity and reduced functionalities in comparison to usually considered NoCs. For example, since exchanged messages are usually six to ten bit values, NoC packets are single phits, sent using source routing. Moreover, the inter-PE communication patterns exhibit a pseudo-random nature, because of the limited adjacency of both interleavers used in turbo codes and parity check matrices associated to

LDPC codes. Thus, the optimal mapping of processing tasks onto PEs does not provide any relevant benefit in terms of NoC traffic [26].

Additional techniques to optimize the NoC traffic are available. In particular, Quality-of-Service (QoS) oriented networks can be considered for turbo and LDPC decoding. In [9], an area- and energy-wise breakdown of different router architectures is presented, providing a comprehensive overview of NoC designs. The CS network and the GuarVC network both target QoS improvement: the first one is a circuit-switched network that relies on simplicity and static allocation of resources, while the second makes use of multiple virtual channels and flow control: priorities are assigned to packets and the traffic flow is optimized. However, CS networks are not effective for channel decoding, because the traffic pattern between PEs is not regular enough. Moreover, both circuit-switched and virtual channel-based NoCs tend to introduce large area and power consumption overheads. For example, a 22-PE NoC (the same choice made in Section 6) implemented as a CS network would be slightly larger than the whole flexible decoder in [19] and its power consumption would be higher by a 2.4 factor. The same NoC, implemented as a GuardVC network, leads to 1.3 times larger area and 2.9 times higher power consumption. Therefore, usual techniques to reduce traffic in NoCs are not effective in the case of NoC-based decoders and dedicated solutions are required.

The complex interactions between the topology of the NoC, the number of PEs, the code and the performance of the decoder have been analyzed with a dedicated bit-true and cycle-accurate tool (JANoCS) [12]. This cycle-accurate tool allows to simulate the processing and communication phases of the decoding process concurrently, observing the state of the network and the traffic-related issues together with Bit Error Rate (BER) performance. From extensive simulations it has been possible to observe how the on-time delivery of messages is of fundamental importance for the decoding process: to analyze

this concept, let us define the Normalized Delivery Time (NDT) as

$$\mathrm{NDT} = \frac{t_d - t_0}{t_u - t_0} \tag{2}$$

where t_0 is the sending time of a message, t_d the arrival time at its destination PE, and t_u the instant when the considered message has to be used at the destination PE. Given a certain decoder architecture, t_u depends on the target throughput while t_d is related to the NoC clock frequency. Messages that are delivered on time, i.e. that reach their destination before t_u, are characterized by NDT≤ 1. Late messages have $t_d > t_u$, leading to NDT> 1. The condition NDT< 1 tends to be very restrictive for codes used in current standards, because of the already mentioned low degree of adjacency in typical inter-PE communication patterns [36]. As a consequence, also a smart mapping of tasks on PEs does not allow for any useful clustering and t_d values have a strongly nonuniform distribution.

In order to guarantee NDT< 1 for the totality of messages, either the throughput must be reduced (which means reducing the PEs clock frequency) or the NoC clock frequency should be increased (without altering the PEs clock frequency) [14].

An ideal situation for a decoder is shown in Fig. 2, that plots the number of messages with respect to the NDT for a WiMAX code of block size 2304 and rate 1/2, decoded with a 16-PE decoder mapped on a Kautz network: in this decoder, to have NDT< 1 for all messages, the NoC frequency is 420 MHz, while the PEs only run at 280 MHz.

Fig. 3 gives the message delivery time distribution for the same code, with the difference that both PE and NoC frequency are 280 MHz. Here, a consistent percentage of messages has NDT> 1. Late messages are extremely disruptive for the performance of the decoder. In case a message has not been

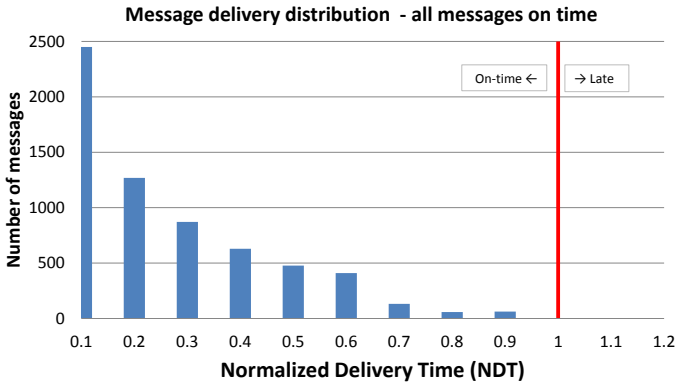

Fig. 2 Message normalized delivery time distribution - all messages reach the destination on time

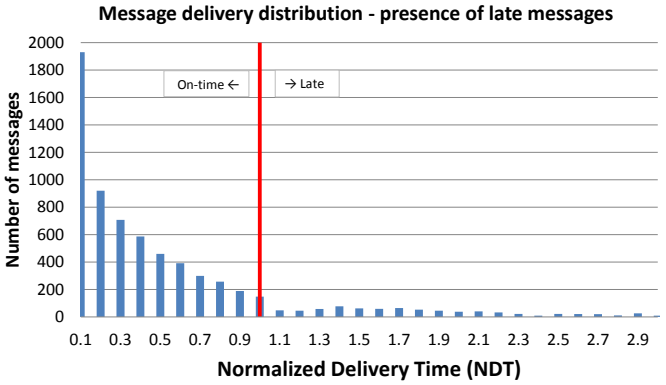

Fig. 3 Message normalized delivery time distribution - some messages do not reach the destination on time

delivered at t_u, the destination PE can use the value computed at the previous iteration, but this leads to additional errors. Fig. 4 plots the BER curves of WiMAX turbo code of size 960, rate 1/3 and WiMAX LDPC code of size 2304, rate 1/2 under different percentages of late messages for illustration. These are obtained by changing the frequency of the 16-PE Kautz NoC used in Fig. 2 and 3. Simulations have been performed on an AWGN channel, with BPSK modulation and fixed point precision (10 bits, 3 of fractional part). It can be seen that very small amounts of late messages ($< 1\%$) degrade the

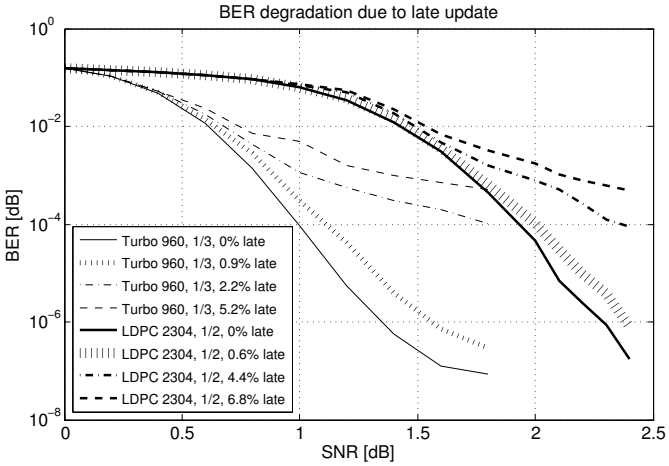

Fig. 4 BER degradation due to late information update

decoder performance, while larger percentages completely compromise the decoder error correction capabilities. It is clear that late messages can not be simply discarded, however efficient methods to reduce and optimize the traffic are introduced in the following Section.

3 Traffic reduction and optimization

Reducing the number of messages traveling on the NoC is bound to speed-up the delivery of those remaining, with shorter queues and fewer collisions. Two techniques have been devised and tested towards these goals: they are based on the general concept that not all information messages (which are of a probabilistic nature) traveling on the network are essential for the success of the decoding. These two techniques (sub-Section 3.1 and 3.2) can be used either alone or combined. Two additional methods to optimize the NoC traffic are described in sub-Section 3.3.

3.1 Hard importance

The Hard Importance (HI) method allows to refrain from sending messages that are estimated to be of low impact on the decoding process. **A similar approach was considered in [31] and [15].** In the LDPC decoding case, the messages traveling on the NoC are different updates of $\lambda_k[c]$. The HI checks are performed once per iteration to each of them. Consider the following comparisons:

$$sgn(\lambda_k^n[c]) = sgn(\lambda_k^{n-1}[c]) \tag{3}$$

$$|\lambda_k^n[c]| \geq |\lambda_k^{n-1}[c]| \tag{4}$$

$$|\lambda_k^n[c]| \geq \texttt{Thr}^{HI} \cdot \max(\lambda_k[c]) \tag{5}$$

where n expresses the n^{th} iteration and $\max(\lambda_k[c])$ is the maximum possible value of $\lambda_k[c]$ given the number of bits assigned to its representation, while $0 \leq \texttt{Thr}^{HI} \leq 1$ expresses the percentage of $\max(\lambda_k[c])$ involved in the comparison. If all above three conditions are verified, $\lambda_k[c]$ is flagged as `unimportant` and the bit LLR is not updated anymore for the rest of the decoding process. The first two comparisons check the presence of a monotonic divergence from zero in the LLR value, while the third requires a large enough absolute value to be satisfied. Compliance with all three checks confirms that the information is already reliable, and that a change of sign (and consequently a bit flip) is extremely unlikely. Since with layered scheduling a $\lambda^k[c]$ is updated many time within each iteration, a single `unimportant` LLR will result in a traffic reduction of several messages.

For turbo decoding, the choice of stopping or not a message can be made by modifying the Symbol Reliability Difference (SRD) criterion proposed in [31]. Defining

$$\delta_m^{(i)}(d_k) = L_m^{(i)}(d_k = d_k^1) - L_m^{(i)}(d_k = d_k^2) \tag{6}$$

as the difference between the logarithmic extrinsic probabilities of the first and second most probable symbols, the original SRD criterion proposes, for each symbol d_k

$$\phi^{(i)}_{m,m'}(d_k) = |\delta^{(i)}_m(d_k) - \delta^{(i)}_{m'}(d_k)| \tag{7}$$

where m' refers to the metrics at the input of the SISO, coming from the previous half-iteration, and m at the output. If condition

$$\phi^{(i)}_{m,m'}(d_k) \leq \mathtt{Thr}^{HI}_{Abs} \cdot \max(\phi^{(i)}_{m,m'}(d_k)) \tag{8}$$

is satisfied, the message can be stopped. Applying this method as is, however, led to unsettling results. This is due to the fact that it is not taken in account that (7) could give very low $\phi^{(i)}_{m,m'}(d_k)$ also if both $\delta^{(i)}_m(d_k)$ and $\delta^{(i)}_{m'}(d_k)$ are very close to 0. In this case $\phi^{(i)}_{m,m'}(d_k)$ expresses just uncertainty about symbols, instead of agreement between SISOs, and the message should not be stopped. For this reason, an additional control has been added for the message to be stopped:

$$|\delta^{(i)}_m(d_k)|, |\delta^{(i)}_{m'}(d_k)| \geq \mathtt{Thr}^{HI}_{Diff} \cdot \max(\delta^{(i)}(d_k)) \tag{9}$$

where $\mathtt{Thr}^{HI}_{Diff}$ assures a degree of reliability on the symbol.

The performance of HI is of large interest, since this method can be applied also to other types of decoders beside NoC-based ones. HI acts on messages by deciding if it is worth updating a value or not, and can be effective also in absence of a NoC. It can consequently be exploited in decoders that rely on shared memory banks: in this case, energy is saved by reducing the number of memory write operations.

3.2 Soft importance

The Soft Importance (SI) technique evaluates the state and the evolution of the information exchanged through the NoC, flagging non-essential messages as `expendable`. In case of collisions, i.e. messages that need to be routed through the same output port, the router arbiter will forward a message and discard all the `expendable` messages that were not granted priority. In LDPC decoding, (3)-(5) are applied in SI with a more relaxed $\texttt{Thr}^{SI}$, while the metrics considered are not $\lambda_k^n[c]$ and $\lambda_k^{n-1}[c]$, but $\lambda_k^{new}[c]$ and $\lambda_k^{old}[c]$. Each PE monitors the evolution of the LLR locally, before and after the processing: if all three comparisons are verified, the message is `expendable`. Since in turbo decoding the HI method already affects each message separately, the SI method can be efficiently implemented with exactly the same mechanism as HI. The `expendable` flag is applied to messages satisfying the modified SRD conditions with more relaxed thresholds $\texttt{Thr}_{Abs}^{SI}$ and $\texttt{Thr}_{Diff}^{SI}$.

3.3 Urgency and Buffer reordering

Smarter and more efficient communication on the NoC can be obtained through the identification of urgent and less urgent messages: traffic optimization deals with the late message issue by prioritizing the former against the latter. Priority-based routing is a well-explored path to guarantee QoS: multiple virtual channels are often assigned different priorities to differentiate traffic flows [17, 26]. The concept of priority is applied here in an original way, by using a single channel with a reordering buffer.

The Urgency technique (U) implements a priority-based collision management policy. In case of collision, priority is given to the most urgent message, i.e. the message which is needed by its destination PE sooner. To allow this

Table 1 Percentages of late messages (% late) and stopped or not sent messages (% stop) for no traffic handling (No TH) and combinations of the proposed methods on 16-PE and 32-PE generalized Kautz and 2D-Mesh NoCs, at BER= 10^{-5}

NoC	Code, length, rate	No TH %	HI %		SI %		HI + SI %		U %	U+ BR %	HI+SI+ U+BR %
		late	late	stop	late	stop	late	stop	late	late	late
16-PE Kautz SP	LDPC 2304, 5/6	9.2	4.8	17.5	8.0	12.3	1.6	23.8	8.5	0.8	0.1
	LDPC 576, 5/6	28.8	19.4	16.9	24.3	12.4	15.4	22.3	25.1	11.1	8.2
	LDPC 1440, 1/2	14.5	6.3	19.5	10.9	13.0	5.5	27.0	12.0	2.6	1.1
	LDPC 864, 1/2	22.3	14.2	18.4	19.2	12.6	11.5	26.1	20.7	4.1	3.0
	Turbo 2400, 1/3	3.3	2.0	20.7	2.8	13.9	1.7	22.8	3.0	0.2	0.0
	Turbo 960, 1/3	5.2	2.9	18.3	4.3	12.8	2.6	23.4	4.6	0.3	0.1
	Turbo 6144, 1/3	4.8	3.0	20.1	3.4	12.9	2.1	22.2	3.2	0.2	0.0
32-PE Kautz SP	LDPC 2304, 5/6	13.6	7.2	17.4	10.3	14.3	3.7	25.1	9.5	1.1	0.3
	LDPC 576, 5/6	42.6	30.0	16.7	35.3	14.5	25.9	24.3	37.5	18.8	16.8
	LDPC 1440, 1/2	19.9	10.1	19.3	14.8	16.0	8.0	28.3	17.1	6.2	2.9
	LDPC 864, 1/2	31.5	22.2	18.7	27.0	15.8	20.1	27.3	26.2	9.9	8.0
	Turbo 2400, 1/3	6.1	4.2	20.7	5.0	15.5	3.5	23.8	4.8	0.8	0.2
	Turbo 960, 1/3	9.7	5.9	18.3	6.9	15.0	4.4	25.2	7.1	1.2	0.4
	Turbo 6144, 1/3	8.8	6.2	20.0	7.1	14.4	4.6	24.9	6.9	0.8	0.1
16-PE Mesh X-Y	LDPC 2304, 5/6	10.2	5.4	17.3	8.7	12.8	3.3	23.9	8.8	1.4	0.3
	LDPC 576, 5/6	32.6	21.0	17.0	26.0	13.1	18.2	24.4	25.8	12.6	8.9
	LDPC 1440, 1/2	16.0	6.9	19.5	11.5	13.3	6.1	28.1	13.2	2.6	1.1
	LDPC 864, 1/2	24.1	15.5	18.4	20.2	13.1	12.4	26.8	22.1	4.1	3.0
	Turbo 2400, 1/3	3.9	2.3	20.7	3.3	14.4	2.1	23.6	3.4	0.5	0.1
	Turbo 960, 1/3	6.2	3.4	18.3	4.7	13.0	3.2	23.9	5.9	0.9	0.3
	Turbo 6144, 1/3	5.9	3.8	20.1	3.8	13.6	3.0	23.7	3.8	0.4	0.2
32-PE Mesh X-Y	LDPC 2304, 5/6	15.5	9.4	17.4	11.3	14.7	4.2	25.6	9.8	1.1	0.3
	LDPC 576, 5/6	44.8	30.4	16.8	37.7	15.1	27.5	25.3	38.4	18.8	16.8
	LDPC 1440, 1/2	22.3	11.0	19.6	16.0	16.3	9.0	29.1	18.0	6.2	2.9
	LDPC 864, 1/2	33.2	23.3	18.5	27.9	16.0	21.3	27.7	26.9	9.9	8.0
	Turbo 2400, 1/3	7.3	5.3	21.0	5.4	16.1	3.8	25.0	5.0	0.9	0.3
	Turbo 960, 1/3	10.4	7.0	18.4	7.1	15.4	4.7	26.4	7.5	1.5	0.6
	Turbo 6144, 1/3	10.1	6.8	19.9	7.6	15.0	5.0	26.2	7.2	1.0	0.3

kind of decision, an urgency field must be added to the message during sending, initialized with an estimate of the number of clock cycles available before the message is needed by another PE. The field must be updated by the routers, taking in account the wait cycles spent in input buffers, and a message is discarded if its urgency reaches zero, avoiding unnecessary switching activity for late messages. With the LDPC case, each PE can perform an estimation based on local knowledge of the instant in which the outgoing message is going to be needed. The precision of the estimate strongly depends on the regularity of the partitioning of the **H** matrix among the PEs. On the contrary, in the

turbo case, since the interleaving rule is known to all PEs, the measure can be exact.

In Buffer Reordering (BR) a fast lane can be created by arranging the messages in the input buffers not in arrival order, but according to the urgency field. The most urgent message in a buffer will consequently always be the first one to be pulled out, increasing its chances of arriving on time.

4 Performance results

The impact of each of the proposed techniques, alone and in combination with one another, has been evaluated with the JANoCS tool [12]. Extensive simulations have considered a wide range of codes, NoC topologies, number of PEs, routing algorithms, PE and RE architectures. Table 1 lists the percentages of late and stopped messages for a set of LDPC and turbo codes taken from WiMAX [3] and 3GPP-LTE [5] standards, considering a decoder implementing different combinations of the proposed techniques. The codes have been mapped on 16-PE and 32-PE NoCs with generalized Kautz [24] and two-dimensional mesh topologies. Meshes are a common topology for middle-sized NoCs, and the simple X-Y routing algorithm [28] joins good performance with ease of implementation. Kautz networks have been proven effective for turbo and LDPC decoding in [27] and [13] respectively also in presence of REs of degree three: routers implement a shortest path routing algorithm [27]. With all the considered NoCs, each PE produces one $\lambda_k[c]$ message per clock cycle in the LDPC case, and one $\lambda_k[u]$ message in the turbo case. While with single-binary turbo codes $\lambda_k[u]$ consists of a single value, three values are necessary in double-binary turbo codes: the width of the simulated channel is adapted accordingly.

HI and SI show high percentages of stopped messages (up to 29%), with substantial reduction in late messages (down to 1.6%), especially for HI and HI+SI. Results for HI and SI were obtained at the SNR point for which BER= 10^{-5} with 10 maximum iterations for LDPC codes and 8 maximum iterations for turbo codes. Iterations are stopped as soon as the codeword is correct, and the number of stopped messages is averaged over all performed iterations. When this kind of early stopping criterion is not present, HI can work as a valid substitute. In fact, if a codeword is correct and its decoding continues, the magnitude of all LLRs keeps growing and HI effectively prevents all messages from being sent. HI and SI inherently introduce some BER degradation, for which careful threshold calibration is necessary: if set too low, the stopped messages can still carry information about uncertain bits, introducing new errors. A wide range of possible threshold values have been considered and simulated, with the final choice representing a good tradeoff between method effectiveness (i.e. number of stopped messages) and BER degradation. The decoder can incur in additional errors in case a metric update is stopped too early by HI or SI: however, threshold calibration allows for results similar to those shown in Fig. 7, where the impact of HI on BER for an LDPC code and a turbo code are presented. The percentages of stopped messages assumed in these simulations are consistent (17% and 18% respectively), but both the LDPC and the turbo code show negligible performance losses. In the plots of Fig. 5 the thresholds have been set as $\mathtt{Thr}^{HI} = 0.2$ and $\mathtt{Thr}^{SI} = 0.1$, while in Fig. 6 as $\mathtt{Thr}^{HI}_{Abs} = 0.25$, $\mathtt{Thr}^{HI}_{Diff} = 0.15$, $\mathtt{Thr}^{SI}_{Abs} = 0.15$ and $\mathtt{Thr}^{SI}_{Diff} = 0.05$. These thresholds are also used to derive the "Ideal THR" curves in Fig. 8 and 9, that show how variations in the threshold values affect the BER performance.

The results given by the urgency U method alone are not satisfying: since its effects are mostly appreciated in case of collisions, which can involve also non-critical messages, its effectiveness alone is limited. However, as soon as

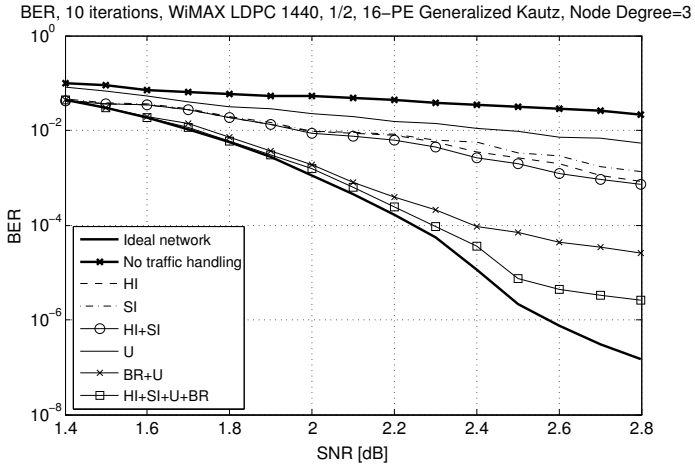

Fig. 5 Impact of the proposed techniques and their combinations - LDPC codes

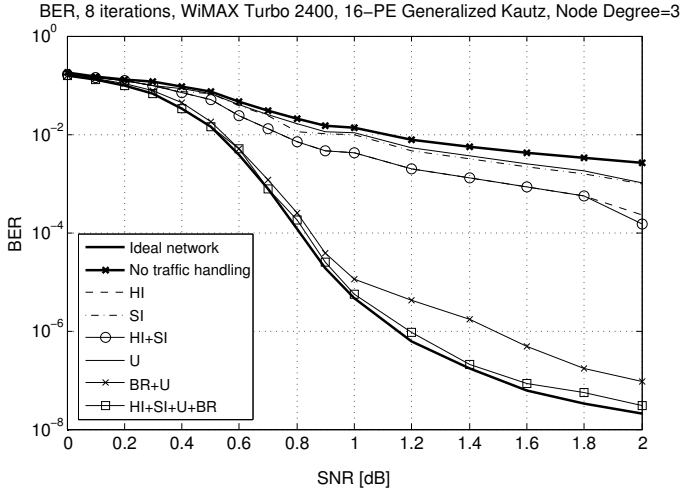

Fig. 6 Impact of the proposed techniques and their combinations - turbo codes

a message is identified as late it is discarded: this means that the percentage of stopped messages for which U contributes is equal to the percentage of late messages. The U+BR urgency-based buffer reordering, which allows non-urgent messages to be delayed in favor of critical ones, drastically reduces the occurrence of late messages (from 11.1 % to 0.2%). Its effectiveness can be improved further by combining the two traffic reduction methods with the traffic

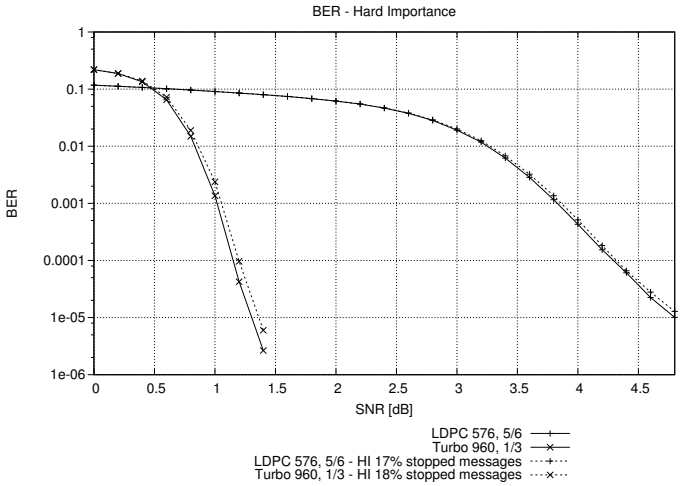

Fig. 7 Impact of HI on BER

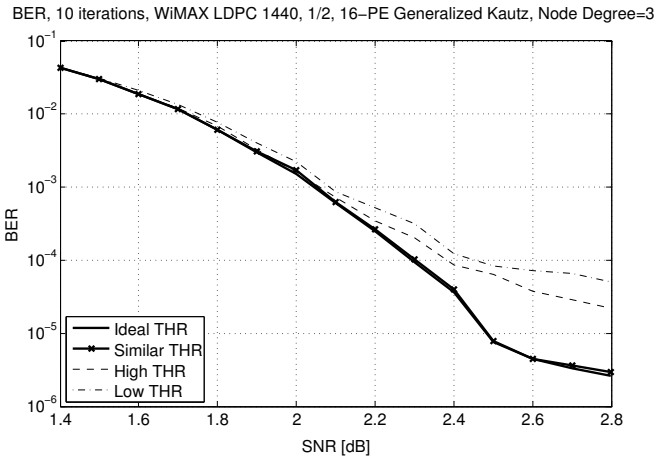

Fig. 8 Impact of different threshold choices on HI+SI+U+BR - LDPC codes

optimization ones. The joint application of all four techniques (HI+SI+U+BR) guarantees a late message percentage close to zero in most cases, while substantially reducing their impact in the remaining cases.

From Table 1 it is possible to make some important observations. As expected from previous analysis [13, 27], the Kautz topology performs better than 2D-mesh when targeting LDPC and turbo codes. Moreover, turbo codes suffer less from late messages w.r.t. LDPC codes (No TH column), thanks to

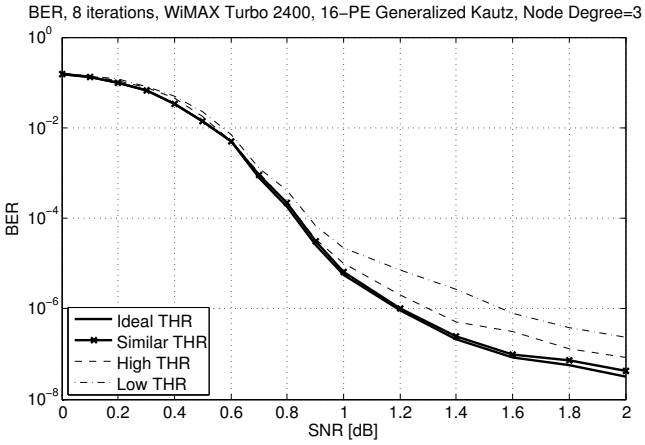

Fig. 9 Impact of different threshold choices on HI+SI+U+BR - turbo codes

their less critical communication phase. It can also be noticed how the size and the rate of the code affect the number of late messages, especially when related to the size and topology of the NoC. A small LDPC code has small **H** layers and a small turbo code has short half-iterations: consequently, the available message delivery times are limited. Moreover, small codes mapped on a large NoC suffers from a large number of late arrivals, since the distance between PEs dominates the transmission times. This is the case of the WiMAX LDPC 576, rate 5/6 in Table 1: similar effects are encountered with larger codes when mapped on the 32-PE NoCs. On the contrary, queues and collisions are the main sources of delay in case of large codes mapped on small NoCs. These limit cases (e.g. LDPC 576, rate 5/6 in Table 1) cannot be completely solved with the implementation of the proposed traffic handling techniques, and need to work in conjunction with alternative techniques: for example, the code can be mapped on a smaller portion of the NoC, and the unused part of the decoder can be deactivated to save energy.

The percentage of late messages is directly reflected on the decoder performance, as shown in Fig. 5 and 6: **a high percentage of late messages**

results in an unreliable decoding process. The effects of the different combinations considered in Table 1 on the BER of an LDPC and turbo code are plotted, using the same NoC and PE architectural choices. As an example, Fig. 5 shows the BER results for a WiMAX LDPC code of rate 1/2 and block size 1440, with 10 maximum iterations; the duo-binary WiMAX turbo code in Fig. 6 has an information block size of 2400 symbols and rate 1/3, decoded with 8 iterations. However, the behaviors observed in Fig. 5 and 6 do not depend on the choice of the codes, but only on the percentage of late messages. In both plots, the "ideal network" curve represents an ideal decoding process, where the interconnection structure does not introduce any latency (lower bound), while the "no traffic handling" curve shows the BER in case of a real decoding process in which no one of the methods is applied (upper bound). It can be noticed how all the performance curves of the proposed solutions span the interval between the upper and lower bound. HI, SI and HI+SI curves do not provide substantial performance improvement, though giving interesting results in terms of traffic reduction, and consequently reducing the switching activity. The U curve is still very close to the "no traffic handling" one, reflecting its limited effectiveness shown in Table 1. The U+BR curve shows the performance in presence of the powerful buffer reordering, with a further step towards the reference curve made by the HI+SI+U+BR curve, that combines all four methods. Though the proposed techniques behave coherently in both cases, they yield slightly better results in the turbo case, as expected from Table 1.

Fig. 8 and 9 show the impact of different threshold values on the BER performance of HI+SI+U+BR applied under the same conditions and to the same codes of Fig. 5 and 6. As mentioned earlier in this section, the "Ideal THR" curves have been obtained by simulating an extensive set of possible threshold values. The final choice has been made by selecting the threshold

values that maximize the number of stopped messages without degrading the BER performance, and the obtained percentages of stopped messages are those reported in Table 1. The choice of the threshold is not very critical, as shown by curves labeled as "Similar THR" in Fig. 8 and 9, which have been derived by rising each "Ideal THR" threshold by 10%: it can be seen how the curves are almost superimposed, and similar minor fluctuations are observed in the number of stopped messages as well. Larger threshold variations have much more influence the BER: the thresholds used in the "High THR" curves are obtained by tripling the "Ideal THR" thresholds. Very high threshold values result in a very small percentage of stopped messages, and in the ineffectiveness of both HI and SI. In fact, "High THR" BER curves are very similar to the U+BR curves of Fig. 5 and 6, where HI and SI are not applied. On the contrary, very low thresholds as the ones used in "Low THR" curves (half of "Ideal THR" thresholds) dramatically increase the number of stopped messages. However, a large number of messages carrying useful information are stopped as well, causing consistent BER degradation.

5 Hardware architecture

The similarity of the calculations involved in HI and SI, and the necessity for controls at each RE for SI, U and BR allow for efficient resource sharing. The multi-mode decoder described in [14] has been taken as reference architecture: among the different implementations, the one denoted **A** in the paper has been selected and called $\mathbf{A}_{ref}$. It relies on a 22-PE Generalized Kautz NoC, and complies with WiMAX, HPAV [2] and DVB-RCS [1] standards, with limited support for WiFi [4].

The HI method can be easily implemented in the SISO. At the beginning of each trellis step one or more read operations from the

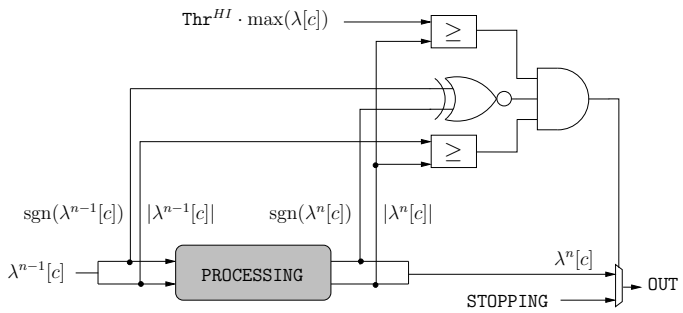

Fig. 10 HI implementation for LDPC codes - `STOPPING` message

data memory are required, and they provide the data needed to calculate the $\delta^{(i)}_{m'}(d_k)$ member of Eq. (7). At the end of the trellis steps, also the data needed to calculate $\delta^{(i)}_{m}(d_k)$ are ready, thus making it possible to compute Eq. (7), (8) and (9).

A memory bit is required for each trellis step to signal if the outgoing messages are `unimportant` and must not be sent. For LDPC codes, since the considered metrics are $\lambda^{n}_{k}[c]$ and $\lambda^{n-1}_{k}[c]$, implementation of HI is less straightforward.The main issue is the fact that the storage of $\lambda^{n}_{k}[c]$ is performed by replacing $\lambda^{n-1}_{k}[c]$. Eq. (3) to (5) can however be executed without any additional memory for the storage of $\lambda^{n-1}_{k}[c]$ by configuring the data memory as Read-Before-Write. When a message is flagged as `unimportant`, the corresponding memory bit is set, and the `unimportant` flag must be propagated to all other PEs. A dedicated `STOPPING` message is sent in place of the `unimportant` message. Fig. 10 shows the circuit responsible for the HI check and eventual creation of the `STOPPING` message: the `PROCESSING` block executes the computations necessary to update each $\lambda_k[c]$. When a `STOPPING` message is received, the `unimportant` memory bit is set in the destination PE, and the `STOPPING` message is propagated. If a PE receives a `STOPPING` message when the `unimportant` memory bit has already been set, the `STOPPING`

message is not sent anymore. The `STOPPING` message is mapped to the lowest negative value that can be represented with the allocated number of bits: for example, with 9 bits, the data dynamic range is mapped to the interval (-255, +255), and the value -256 is recognized as a `STOPPING` message.

The HI method can also be applied to save energy by simply reducing the number of memory write operations at the destination PEs. When a given message is received by its destination PE, the writing into the internal memory can be avoided if the message is recognized as unimportant. The implementation of this functionality still exploits the `STOPPING` message, which is used to control the write enable signal of the memory and prevent write operation. However, in this case, the `STOPPING` message must be sent to the memory at every iteration.

The implementation of SI follows that of HI in the turbo case, only requiring two additional comparators for the different thresholds in (8) and (9). It is instead much simpler than HI for LDPC codes, since both $\lambda_k^{new}[c]$ and $\lambda_k^{old}[c]$ are available during each parity check computation, and there is no need for flag propagation. Together with the simple computational logic in the PEs, a flag bit signaling if a message is `expendable` or not must be added also in all NoC FIFOs and channels, while changes in the write and read pointers (`WPTR` and `RPTR`) of each FIFO are forced by the RE arbiter in case of collisions.

The U method requires, for the initialization of the `URGENCY` field of outgoing messages, the estimation of the available delivery time. This measure can be obtained, in the turbo case, thanks to the current trellis step together with the globally known interleaving rule. Since each SISO processes a sequential set of trellis steps, the destination memory address is a precise identifier of the time instant a message will be needed. The `URGENCY` field of each outgoing

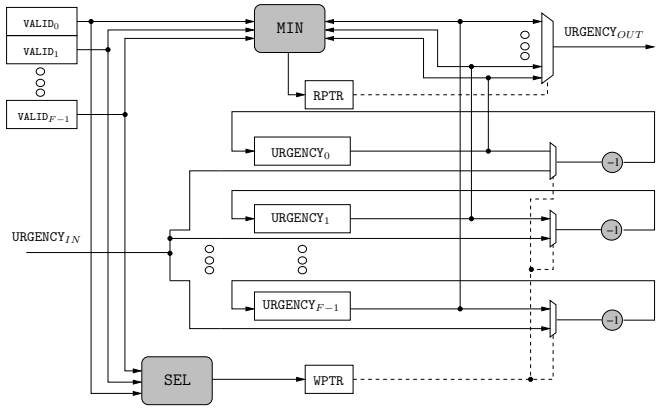

Fig. 11 Urgency-based buffer reordering

message can be initialized as

$$U_{turbo} = T_{half} - t_{send} + t_{need} \tag{10}$$

where T_{half} is the duration of a half iteration, t_{send} is the time stamp of the sending instant and t_{need} equals to the destination memory address multiplied by the number of cycles needed to complete each trellis step. The destination address is also used in the initialization of U in the LDPC case. By multiplying it by the minimum row degree of $\mathbf{H}$, a lower bound of t_{need} is obtained, thus leading to the following equation:

$$U_{LDPC} = t_{need} - t_{send} \tag{11}$$

The urgency field requires additional bits in all the NoC FIFOs and channels, together with the simple initialization logic at each PE. Moreover, each FIFO of length F needs F adders to update U at each clock cycle, while `WPTR` and `RPTR` must be updated in case the urgency field reaches zero, and the corresponding message discarded. All the FIFO memory elements must be available for writing at each clock cycle: the FIFO consequently must be implemented

with registers, and not with a RAM. Finally, the priority of the RE arbiter is changed from being FIFO-length-based [14] to urgency-based.

The implementation of BR requires all the modifications described for U, plus a novel method for the update of WPTR and RPTR. The RE input buffers in fact lose their FIFO nature, since the input order is not guaranteed to be the output order. Fig. 11 shows the simplified structure of the proposed buffer reordering mechanism; white blocks represent registers, while gray blocks indicate additional computation elements. Along with the URGENCY field, each buffer element requires an additional VALID field. Read and write operations on the buffer take in account external signals (PUSH, POP) and the internal state of the buffer (IS_EMPTY, IS_FULL). Every time a write operation is performed, the VALID field of the corresponding buffer element is set, while it is cleared with a read operation. The VALID fields are necessary to keep track of the free and occupied elements, since the irregular input and output orders prevent WPTR and RPTR from being used for this purpose. During write operation, the urgency field from the incoming message $\texttt{URGENCY}_{IN}$ is substituted according to WPTR to one of the stored $\texttt{URGENCY}_X$ during the U update process. Thus, the updated value of $\texttt{URGENCY}_X$ is $\texttt{URGENCY}_{IN} - 1$ instead of $\texttt{URGENCY}_X - 1$. In a concurrent read operation a $\texttt{URGENCY}_X$ value is selected as the buffer output $\texttt{URGENCY}_{OUT}$ according to RPTR. The SEL module chooses the update value of WPTR among the elements with cleared VALID field with fixed priority. The MIN module, instead, updates RPTR as the pointer to the minimum $\texttt{URGENCY}_X$ among the VALID ones. The whole operation occurs within a single clock cycle, and correct functionality of the circuit has been tested in 90 nm CMOS technology for up to 10 buffer elements, with 200 MHz as target throughput. The area overhead introduced by the additional operations in the reordering buffer has been evaluated with respect to a typical FIFO buffer. For example, a reordering buffer with five buffer elements accounts for a little more than

Table 2 Effect of traffic handling on area occupation (CMOS 90 nm technology, post-layout results)

	$\mathbf{A}_{ref}$ Area		$\mathbf{A}_{new}$ Area		Overhead
	[mm^2]	%	[mm^2]	%	%
Core Memory	1.46	53%	1.53	49%	4.8%
SISO Logic	0.42	15%	0.45	15%	7.1%
LDPC PE Logic	0.31	11%	0.38	12%	22.6%
NoC	0.56	21%	0.75	24%	33.9%
Total	2.75	100%	3.11	100%	13.1%

Table 3 Effect of traffic handling on power consumption (CMOS 90 nm technology, 1.0 V supply)

	$\mathbf{A}_{ref}$		$\mathbf{A}_{new}$		Power Gain
	Pow	f_{clk}	Pow	f_{clk}	
	[mW]	[MHz]	[mW]	[MHz]	%
PEs	68.0	200	70.1	200	+3.1%
NoC	48.6	333	29.1	200	-40.2%
Total	116.6	333-200	99.2	200	-15.0%

2000 μm^2, against the 850 μm^2 of a regular FIFO, with a $\times$2.35 area increment factor.

6 Implementation

To help a fair comparison with the state of the art, all the modifications have been applied to the $\mathbf{A}_{ref}$ decoder, creating a new implementation $\mathbf{A}_{new}$, targeting 90 nm CMOS technology: starting from VHDL models designed after the exploration performed with JANoCS, synthesis has been carried out with Synopsys Design Compiler, functional simulation and validation with Mentor Graphics ModelSIM, and place and route with CADence SoC Encounter. Table 2 dissects the post place and route area occupation of various components of the decoder before and after the implementation of the proposed methods. The small increase in memory occupation is due to the extra memory bit for the `unimportant` flag related to HI, shared between SISOs and LDPC PEs.

The simple logic required for both HI and SI in the turbo case results in an additional 7.1% area occupation for SISOs, while the more complex operations involved in the LDPC PE for HI lead to a higher area overhead. The widths of NoC buffers and channels have been increased to accommodate the `URGENCY` field (five bits), the `VALID` bit of BR and the `expendable` bit of SI. This, together with the additional logic for U and BR, heavily affects the NoC area, with an overhead exceeding 30%. With a total area of 3.11 mm^2, the modified decoder is 13.1% larger than $\mathbf{A}_{ref}$. However, as mentioned in Section 2, $\mathbf{A}_{ref}$ deals with the late message issue with a NoC clock frequency higher than the PE clock frequency: with the introduction of the traffic reduction and optimization techniques, however, this is not necessary anymore, and the decoder can be clocked with a single frequency. Table 3 details the worst case power consumption Pow for $\mathbf{A}_{ref}$ and $\mathbf{A}_{new}$ architectures: in the original decoder the clock frequency f_{clk} is set to 200 MHz for the PEs and to 333 MHz for the NoC. The global power consumption is 116.6 mW, with the NoC accounting for 41.7% of the total. Total power consumption in $\mathbf{A}_{new}$ is reduced of 15% w. r. t. $\mathbf{A}_{ref}$, while the power gain on the NoC alone reaches a very consistent reduction of 40.2%. This result basically derives from the lower clock frequency of the NoC with respect to architecture $\mathbf{A}_{ref}$, but the clock frequency reduction is made possible thanks to the adoption of the described traffic reduction techniques. A final measure of the ratio between costs and advantages in reducing the NoC power consumption can be obtained as

$$(Pow^{NoC}_{\mathbf{A}_{new}} + Pow^{PE}_{\Delta} - Pow^{NoC}_{\mathbf{A}_{ref}})/Pow^{NoC}_{\mathbf{A}_{ref}} = -35.8\% \tag{12}$$

where Pow^{PE}_{Δ} is the power consumption increment in PEs due to the contribution of HI, SI and U initialization. The implementation of HI+SI+U+BR,

Table 4 Performance and energy consumption comparison (CMOS 90 nm technology, 1.0 V supply)

Code		$\mathbf{A}_{ref}$	$\mathbf{A}_{new}$	$\mathbf{A}_{red}$
LDPC	$n_{it}^{(max)}$	10	10	9
2304	ΔSNR @ BER=10^{-5} [dB]	0.0	0.02	0.09
5/6	E_{frame} [μJ]	2.03	1.73	1.83
Turbo	$n_{it}^{(max)}$	8	8	7
6144	ΔSNR @ BER=10^{-5} [dB]	0.00	0.00	0.15
1/3	E_{frame} [μJ]	7.82	6.65	6.84

taking in account all the introduced overheads, brings a power reduction on the NoC equal to 35.8% of $Pow_{\mathbf{A}_{ref}}^{NoC}$.

The actual impact of HI on the energy consumption of centralized decoders can also be estimated. In [7] the energy breakdown of a decoder based on memory banks sharing is given. The energy consumption of the γ-memory accounts for 70% of total dynamic energy for WiMAX LDPC code of size 576 and rate 5/6. For this particular code if HI is applied the percentage of stopped messages, and consequently of avoided write operations, is around 17% (Table 1). Since write operations contribute for approximately 50% of the energy expenditure, the implementation of HI leads to a 6% reduction in the total decoder energy consumption.

A simple direct way to reduce the traffic on the NoC is to reduce the number of iterations of the channel decoder. Table 4 compares the impact of such method with respect to the proposed techniques in terms of energy efficiency and BER degradation. In this table we consider the BER performance and energy consumption of $\mathbf{A}_{red}$, i.e. $\mathbf{A}_{ref}$ with a reduced number of maximum iterations, and compares it with $\mathbf{A}_{new}$ and the original $\mathbf{A}_{ref}$, for two code examples. The number of performed iterations is represented by $n_{it}^{(max)}$, $\mathbf{E}_{frame}$ expresses the energy spent per decoded frame and ΔSNR shows the perfor-

Table 5 LDPC/Turbo architectures comparison: CMOS technology process (Tp), total area occupation (A_{tot}, normalized area occupation for 90nm technology (An_{tot}), clock frequency (f_{clk}), peak power consumption (Pow), energy efficiency (E_{eff}), maximum number of iterations ($n_{it}^{(max)}$), code length (N) and rate (r), interleaver size (K) and throughput (T), Area efficiency (A_{eff})

Decoder		[a] $\mathbf{A}_{ref}$	[a] $\mathbf{A}_{new}$	[6]	[19]	[a] [16]	[a] [34]
Tp [nm]	LDPC	90	90	65	45	65	-
	CTC					-	130
Supply Voltage [V]	LDPC	1.1	1.1	1.1	N/A	1.1	-
	CTC					-	1.2
A_{tot} [mm^2]	LDPC	2.75	3.11	0.62	0.9	2.32	-
	CTC					-	3.57
[c] An_{tot} [mm^2]	LDPC	2.75	3.11	1.19	3.6	4.45	-
	CTC					-	1.71
f_{clk} [MHz]	LDPC	[b] 333-200	200	400	150	40	-
	CTC					-	302
Pow [mW]	LDPC	116.6	99.2	76.8	86.1	29.5	-
	CTC					-	788.9
[c] E_{eff} [$\frac{nJ}{bits}$]	LDPC	0.166	0.141	0.085	1.208	0.008	-
	CTC	0.079	0.067	2.193	1.176	-	0.121
$n_{it}^{(max)}$	LDPC	10	10	10	8	10	-
	CTC	8	8	5	8	-	5.5
N, r	LDPC	2304, 1/2	2304, 1/2	2304, 5/6	N/A	2304, 5/6	-
K	CTC	2400	2400	2400	N/A	-	6144
T [Mb/s]	LDPC	70	70	237.8	71.05	1152	-
	CTC	183	183	37.2	73.46	-	390.6
[c] A_{eff} [$\frac{bits}{mm^2 \cdot kcycles}$]	LDPC	1272	1125	4995	1051	64719	-
	CTC	2662	2354	391	1088	-	4160

[a] post–layout results
[b] f_{clk}^{NoC}
[c] Power and area normalization factors are $(Tp1/Tp2)^3$ and $(Tp1/Tp2)^2$

mance degradation with respect to $\mathbf{A}_{ref}$ when BER=10^{-5}. In $\mathbf{A}_{red}$ the reduced number of iterations (by only one iteration) leads to a proportional reduction of $\mathbf{E}_{frame}$, but non-negligible performance degradation is present. It is especially noticeable in the turbo case, that relies on a smaller $n_{it}^{(max)}$. The proposed implementation outperforms the iteration reduction in terms of energy savings, while at the same time affecting the BER performance only marginally.

Finally Table 5 compares the results of $\mathbf{A}_{new}$ with $\mathbf{A}_{ref}$ and few recent related state-of-the-art LDPC and turbo decoders. The energy efficiency has been introduced to help a fair comparison, and is defined as $E_{eff} = \mathrm{Pow}/(T \cdot n_{it}^{(max)})$, where T is the achieved through-

put and $n_{it}^{(max)}$ is the maximum allowed number of iterations, while the power consumption has been normalized to the CMOS 90 nm process. This measure expresses the energy spent for each decoded bit. The area efficiency, relates the area occupation normalized to the CMOS 90 nm process ($\mathbf{An}_{tot}$) to T and the clock frequency f_{clk}, and is defined as $\mathbf{A}_{eff} = (T \cdot n_{it}^{(max)}/f_{clk}) \cdot (1000/\mathbf{An}_{tot})$, where the 1000 multiplication factor changes the unit of measure from $[\frac{bits}{mm^2 \cdot cycles}]$ to $[\frac{bits}{mm^2 \cdot kcycles}]$. The +13.1% in $\mathbf{A}_{tot}$ that $\mathbf{A}_{new}$ exhibits w. r. t. $\mathbf{A}_{ref}$ leads to a lower $\mathbf{A}_{eff}$. The effects of the proposed methods, though, can be really appreciated by observing Pow and E_{eff}. The reduction of the NoC clock frequency to 200 MHz overcompensates the increased peak power consumption caused by the additional logic, leading to improved E_{eff} values.

The multi-standard LDPC and turbo presented in [6] has a very small area occupation, with uneven throughput between LDPC and turbo mode. This situation leads to very high maximum $\mathbf{A}_{eff}$ and $\mathbf{E}_{eff}$ in LDPC mode. $\mathbf{A}_{new}$, instead, is far more efficient in turbo mode, and yields better results also in [6] LDPC mode worst case operating conditions (N=672, $r = 1/2$, 20 iterations). The flexible LDPC/turbo decoder designed in [19] and $\mathbf{A}_{new}$ achieve comparable throughputs when decoding LDPC codes, with [19] having a larger $\mathbf{An}_{tot}$. This leads to $\mathbf{A}_{new}$ having slightly better $\mathbf{A}_{eff}$ and $\mathbf{E}_{eff}$: the gap is much larger in turbo mode.

The high parallelism, single-standard WiMAX LDPC decoder presented in [16] guarantees very high throughput with a 40 MHz frequency, that allows for reduced power consumption and great efficiencies. Though [16] has been designed for ultra-low power consumption, and $\mathbf{A}_{new}$ targets multiple code types and standards, the

normalized power gap between them is minimal. This work fares even better when compared to the dedicated Application-Specific Integrated Circuit (ASIC) targeting 3GPP-LTE turbo codes in [34], that yields good A_{eff} and throughput. The estimated normalized Pow of 261 mW with the 90 nm node is still higher than A_{new}, with lower efficiency.

7 Conclusion

The paper proposes novel power reduction techniques for NoC-based channel decoders: extensive traffic and performance analysis are performed, while the hardware implementation allows to assess the impact on a relevant design example and to obtain accurate power consumption. By dealing with the late message delivery problem through traffic reduction and optimization, the proposed techniques allow to avoid power expensive solutions **while at the same time guaranteeing a reliable decoding process**. Moreover, one of the methods can be applied to any channel decoder architecture. Simulation results show the performance of different combinations of the presented techniques: all of them are effective, in particular HI+SI+U+BR, that allows to reach the target throughput with minor BER degradation also in presence of a late message percentage of more than 20% of the total. The implementation of this method on a known decoder allows for a 15% total power reduction (40.2% NoC power reduction): it is presented and compared to the state of the art. Post place and route results show small area occupation and very low power consumption, with good efficiency measures w.r.t. even much less flexible decoder.

References

1. Digital video broadcasting (DVB); interaction channel for satellite distribution systems (2005)
2. Homeplug AV specification (2005)
3. Ieee standard for local and metropolitan area networks part 16: Air interface for fixed and mobile broadband wireless (2006)
4. Ieee standard for information technology–telecommunications and information exchange between systems–local and metropolitan area networks (2009)
5. Multiplexing and channel coding (2012)
6. Alles, M., Vogt, T., Brehm, C., Wehn, N.: FlexiChaP: A dynmically reconfigurable ASIP for channel decoding for future mobile systems pp. 293–314 (2010). DOI 10.1109/TURBOCODING.2008.4658677
7. Amador, E., Pacalet, R., Rezard, V.: Optimum LDPC decoder: A memory architecture problem. In: ACM/IEEE Design Automation Conference, pp. 891–896 (2009)
8. Bahl, L.R., Cocke, J., Jelinek, F., Raviv, J.: Optimal decoding of linear codes for minimizing symbol error rate. IEEE Trans. on Information Theory **20**(3), 284–287 (1974)
9. Banerjee, A., Wolkotte, P., Mullins, R., Moore, S., Smit, G.J.M.: An energy and performance exploration of network-on-chip architectures. IEEE Trans. on Very Large Scale Integration (VLSI) Systems **17**(3), 319–329 (2009). DOI 10.1109/TVLSI.2008.2011232
10. Benes, V.E.: Mathematical Theory of Connecting Networks and Telephone Traffic. Academic Press (1965)
11. Benini, L.: Application specific NoC design. In: Design, Automation and Test in Europe Conference and Exhibition, pp. 1330–1335 (2006)
12. Condo, C., Baghdadi, A., Masera, G.: A joint communication and application simulator for noc-based custom socs: Ldpc and turbo codes parallel decoding case study. In: Digital System Design (DSD), 2013 Euromicro Conference on, pp. 168–174 (2013). DOI 10.1109/DSD.2013.26
13. Condo, C., Martina, M., Masera, G.: A network-on-chip-based turbo/LDPC decoder architecture. In: IEEE Design, Automation Test in Europe Conference Exhibition, pp. 1525 –1530 (2012)
14. Condo, C., Martina, M., Masera, G.: VLSI implementation of a multi-mode turbo/LDPC decoder architecture. IEEE Trans. on Circuits and Systems I **60**(6), 1441–1454 (2013). DOI 10.1109/TCSI.2012.2221216

15. Condo, C., Masera, G.: A flexible NoC-based LDPC code decoder implementation and bandwidth reduction methods. In: Conference on Design and Architectures for Signal and Image Processing, pp. 1 –8 (2011). DOI 10.1109/DASIP.2011.6136889

16. Cui, Y., Peng, X., Chen, Z., Zhao, X., Lu, Y., Zhou, D., Goto, S.: Ultra low power QC-LDPC decoder with high parallelism. In: IEEE International SOC Conference, pp. 142 –145 (2011). DOI 10.1109/SOCC.2011.6085136

17. Dally, W.: Virtual-channel flow control. In: International Symposium on Computer Architecture, pp. 60–68 (1990). DOI 10.1109/ISCA.1990.134508

18. Gallager, R.G.: Low density parity check codes. IRE Trans. on Information Theory **IT-8**(1), 21–28 (1962)

19. Gentile, G., Rovini, M., Fanucci, L.: A multi-standard flexible turbo/LDPC decoder via ASIC design. In: International Symposium on Turbo Codes & Iterative Information Processing, pp. 294–298 (2010)

20. Hocevar, D.: A reduced complexity decoder architecture via layered decoding of LDPC codes. In: IEEE Workshop on Signal Processing Systems, pp. 107 – 112 (2004). DOI 10.1109/SIPS.2004.1363033

21. Hu, W.H., Bahn, J.H., Bagherzadeh, N.: Parallel LDPC decoding on a Network-on-Chip based multiprocessor platform. In: International Symposium on Computer Architecture and High Performance Computing, pp. 35 –40 (2009). DOI 10.1109/SBAC-PAD.2009.9

22. Hung, W., Addo-Quaye, C., Theocharides, T., Xie, Y., Vijakrishnan, N., Irwin, M.: Thermal-aware IP virtualization and placement for networks-on-chip architecture. In: IEEE International Conference on Computer Design: VLSI in Computers and Processors, pp. 430 – 437 (2004). DOI 10.1109/ICCD.2004.1347958

23. Hunt, A., Crozier, S., Gracie, K., Guinand, P.: A completely safe early-stopping criterion for max-log turbo code decoding. In: International Symposium on Turbo Codes and Related Topics, pp. 1–6 (2006)

24. Imase M.; Itoh, M.: A design for directed graphs with minimum diameter. IEEE Trans. on Computers **C-32**(8), 782–784 (Aug. 1983)

25. Li, J., You, X.H., Li, J.: Early stopping for LDPC decoding: convergence of mean magnitude (CMM). IEEE Communications Letters **10**(9), 667 –669 (2006). DOI 10.1109/LCOMM.2006.1714539

26. Lo, S.H., Lan, Y.C., Yeh, H.H., Tsai, W.C., Hu, Y.H., Chen, S.J.: QoS aware BiNoC architecture, year=2010, pages=1-10, keywords=, issn=1530-2075,. In: IEEE International Symposium on Parallel Distributed Processing

27. Martina, M., Masera, G.: Turbo NOC: A framework for the design of Network–on–Chip–based turbo decoder architectures. IEEE Trans. on Circuits and Sistems I **57**(10), 2776 – 2789 (2010)
28. de Mello, A.V., Ost, L.C., Moraes, F.G., Calazans, N.L.V.: Evaluation of routing algorithms on mesh based NoCs. In: Faculdade de Informatica, Pontifcia Universidade Catolica do Rio Grande do Sul, Tech. Rep, vol. 40 (2004)
29. Moussa, H., Baghdadi, A., Jezequel, M.: Binary de Bruijn interconnection network for a flexible LDPC/turbo decoder. In: IEEE International Symposium on Circuits and Systems, pp. 97–100 (2008)
30. Moussa, H., Baghdadi, A., Jezequel, M.: Binary De Bruijn onchip network for a flexible multiprocessor LDPC decoder. In: ACM/IEEE Design Automation Conference, pp. 429–434 (2008)
31. Muller, O., Baghdadi, A., Jezequel, M.: Bandwidth reduction of extrinsic information exchange in turbo decoding. IET Electronics Letters **42**(19), 1104–1105 (2006)
32. Neeb, C., Thul, M., Wehn, N.: Network-on-chip-centric approach to interleaving in high throughput channel decoders. In: Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on, pp. 1766 – 1769 Vol. 2 (2005). DOI 10.1109/ISCAS.2005.1464950
33. Rovini, M., Gentile, G., Rossi, F., Fanucci, L.: A scalable decoder architecture for ieee 802.11n ldpc codes. In: IEEE Global Telecommunications Conference, pp. 3270–3274 (2007). DOI 10.1109/GLOCOM.2007.620
34. Studer, C., Benkeser, C., Belfanti, S., Huang, Q.: Design and implementation of a parallel turbo-decoder ASIC for 3GPP-LTE. Solid State Circuits, IEEE Journal of **46**(1), 8– 17 (2011)
35. Urard, P., Paumier, L., Viollet, M., Lantreibecq, E., Michel, H., Muroor, S., Coates, B., Gupta, B.: A generic 350 mb/s turbo-codec based on a 16-states siso decoder. In: IEEE International Solid-State Circuits Conference, pp. 424–536 Vol.1 (2004). DOI 10.1109/ISSCC.2004.1332775
36. Vacca, F., Moussa, H., Baghdadi, A., Masera, G.: Flexible architectures for LDPC decoders based on network on chip paradigm. In: Euromicro Conference on Digital System Design, pp. 582–589 (2009)