On Chip Interconnects for Multiprocessor Turbo Decoding Architectures

M. Martina[a], G. Masera[a], H. Moussa[b], A. Baghdadi[b]

[a]*Dipartimento di Elettronica, Politecnico di Torino, 10129 Torino, Italy*
[b]*Electronics Department, TELECOM Bretagne, Technopole Brest Iroise, 29238 Brest, France*

Abstract

Turbo codes are among the most powerful and widely adopted error correcting codes in several communication applications. The high throughput requirements of current and future standards impose that parallel decoders composed by multiple interconnected processing elements are used at the receiver side to efficiently decode turbo codes. In this work, on chip interconnects for multiprocessor turbo decoding are investigated. Due to the dominant trend towards the design of flexible, multi-standard decoders, capable to support the decoding of several turbo codes, the network on chip approach is seen as a viable and promising solution, although the specific characteristics of the addressed application impose a drastic simplification in the network organization. Both indirect and direct network topologies are studied and experimental results show that a network-on-chip based decoder made of 16 processing elements can achieve a throughput of several hundreds of Mbps. Moreover, the area required by the network compares favorably with previously published works on flexible interconnect architectures for turbo decoding and the cost overhead of NOC based solutions with respect to a fully dedicated implementation is limited to 13%.

Keywords: NOC, Turbo decoder, MPSoC, VLSI

1. Introduction

The last few decades witnessed dramatic advancements in wireless communications, which have been improving at a fast rate in terms of both throughput and reliability. On the implementation side, this trend came with a continuous growth of the computational complexity of wireless receivers that has stimulated very relevant efforts in the search for efficient hardware solutions, capable of giving proper support to the demanded computation with limited Silicon area.

More recently, the introduction of a large number of continuously evolving wireless standards has raised the need for flexible hardware implementations, able to support on a unique component multiple standards and to dynamically adapt to different communication modes. Therefore, while throughput and area have been the dominant metrics driving the optimization of digital receivers for a specific application, today the need for flexible systems able to support different operative modes and standards has changed the perspective. In particular, the software defined radio (SDR) paradigm makes flexibility a fundamental property of future receivers [1]. To enable a single modem to service multiple different wireless systems, highly flexible solutions are needed. The cost of the so-called velcro approach, where multiple dedicated building blocks are separately optimized and simply placed end to end in the multi-standard receiver, becomes rapidly unacceptable with the number of standards to be supported. Flexibility can be also seen as a viable way to reduce development costs and extend products life.

One of the most demanding processing functions in a modern wireless receiver is channel decoding. Especially the decoding algorithms of binary and double-binary turbo codes have a very high computational complexity: for this class of error correcting codes, the decoding process contributes at least 40% to the total computational complexity of the physical layer of a wireless system, depending on the implementation platform [2] [3] [4]. Every standard uses multiple different codes and different configurations of the decoding algorithm. For example, the Turbo coding algorithm used in W-CDMA employs a different polynomial, block size, coding rate and termination method from that used in WiMAX. To show the widely heterogeneous set of turbo codes currently adopted in communication standards, Table 1 summarizes their main parameters as well as the required throughputs. Moreover, novel codes with enhanced properties with better overall performance are still investigated and these novel solutions might be adopted in future standards. Thus, flexible architectures able to support as much decoding options as possible seems the only viable solution for reducing overall costs.

The answer to the severe constraints of high processing throughput and full flexibility in terms of decoded code and executed decoding algorithm in clearly given by Multi-Processor System-on-Chip (MPSoC) architectures. This kind of solution is currently being widely investigated in order to accommodate the mentioned needs of high throughput and flexibility in emerging wireless communication standards. The software programmability of processors guarantees large flexibility in terms of computation. In order to obtain the same level of flexibility in terms of inter-processor communication, a proper interconnect structure must be used. Decoding algorithms for turbo codes involve iterative and intensive exchanges of data between the processing elements mapped onto different processors and the processed data typically show a very limited locality. Moreover, the inter-processor communication needs are different for different codes to be decoded. Thus, a really flexible communication network for an MPSoC based turbo decoder tends to be quite a complex structure with strict bandwidth and latency constraints. Conventional on-chip buses are known to become inefficient in large systems and are not adequate for channel decoders. On the contrary, the recently emerged Network-on-Chip (NOC) paradigm seems to be well applicable to implement inter-processor interconnects for channel decoders.

After the introduction of the NOC concept **[5, 6, 7, 8]** the scientific literature focused on general purpose NOC solutions; more recently the idea of Application Specific NOC (ASNOC) [9] was proposed as a method to improve efficiency, through a careful tailoring of the network features around the specific application to be supported. All ASNOC examples available in the literature are related to fairly complex applications, which involve heterogeneous processing tasks or IPs (Intellectual Property), occupy a fairly relevant physical area and make use of advanced methods for routing and congestion control. Following the classification given in [10], we call this kind of NOC InterIP-NOC. On the contrary, [10] defines as IntraIP-NOC any network whose domain area is restricted to be internal to a single IP: in this case, the NOC extend is typically much smaller, processing tasks are usually homogeneous and stringent constraints are posed on the NOC overhead, which is limited resorting to simple routing methods.

In this framework, the contribution of our work is twofold. i) To give results for a wide set of parameters of the NOC approach in turbo decoder architectures. This aspect has been partially investigated in some previous works, whereas in this paper it is treated in a unified way exploring both indirect and direct topologies. ii) To highlight the characteristics of a particular application, turbo decoding, that can take advantage of the NOC approach. In particular, dealing with the IntraIP-NOC concept, this work aims to focus the attention of the NOC community on a special case of Application Specific NOCs.

In the following of the paper, Section 2 briefly introduces the decoding algorithm together with its speed and flexibility needs, while Section 3 reviews the state of the art on the implementation of multi-standard or multi-mode parallel turbo decoders. Obtained results in the design of two different types of IntraIP-NOC for turbo decoding are presented in the two successive sections: specifically, Section 4 deals with some indirect interconnect structures designed for parallel turbo decoding, while an extensive study of the complexity/performance trade-offs offered by different direct NOCs, with different topologies, routing algorithms and node architectures, is given in Section 5. Finally, in Section 6 comparisons are presented and conclusions are drawn in Section 7.

2. Standard FEC Turbo Decoding Systems

A Turbo encoder is basically formed by two convolutional encoders and one interleaver. The interleaver can be seen as a memory where data are written in the natural order and later read in an interleaved order, according to a permutation law, which is a specific characteristic of the code and has the purpose of minimizing the correlation between original and interleaved data. The three units can be concatenated in a serial or parallel way and the uncoded symbols received by the turbo encoder can be either binary (e.g. UMTS case) or double-binary (like in the case of the WiMAX standard). In the parallel scheme, the first **convolutional coder (CC)** directly receives the uncoded sequence in the natural order, while the second encoder is fed by the interleaved sequence (see Fig. 1 (a)). The whole encoder operates on blocks of data having the same length as the interleaving sequence.

In the decoding process, a block of soft inputs (also called intrinsic information) are received from the demodulator and processed by a first component decoder (or SISO unit), which implements the BCJR algorithm **[11]**. This processing step is usually referred to as half iteration and generates at the SISO output results called extrinsic information. This sequence is interleaved and

Table 1: Turbo code based channel coding schemes in modern communication standards

Standard	Code	Rates	States	Block size	Throughput per channel
UMTS	binary	1/3	8	40 - 5114	up to 2 Mbps
HSDPA	binary	1/2 - 3/4	8	40 - 5114	up to 84 Mbps
CDMA-2k	binary	1/2 - 1/5	8	378 - 20736	up to 2 Mbps
IEEE802.16 (WiMAX)	double-binary	1/2 - 3/4	8	up to 648	up to 144 Mbps
Inmarsat	binary	1/2	16	up to 2608	up to 64 kbps
LTE	binary	1/3 - 7/8	8	40 - 6144	up to 326,4 Mbps

then processed by a second SISO unit (second half iteration) working as the component decoder for the second CC code: this unit also generates an output extrinsic sequence, which is de-interleaved and fed back to the first SISO. The decoding process is repeated several times, with the same intrinsic information and only extrinsic sequences are passed between the SISO. Extrinsic information is a measure of the reliability associated to possible estimations of uncoded symbols: thus, if the decoding process converges, the extrinsic values tend to evolve along iterations in the direction of increased reliability for the symbols that have been actually sent. The decoding is stopped after a predetermined number of iterations (typically in the range 4 to 8, depending on desired performance).

As the focus of this paper is on the IntraIP-NOC required to support communication needs among multiple processors that concurrently decode a turbo code, **the equations related to the decoding algorithm are not strictly necessary to understand the interconnect needs among processing elements: for this reason, we do not show the equations related to the BCJR algorithm and the reader can refer to [12], [13], [14] for more details.**

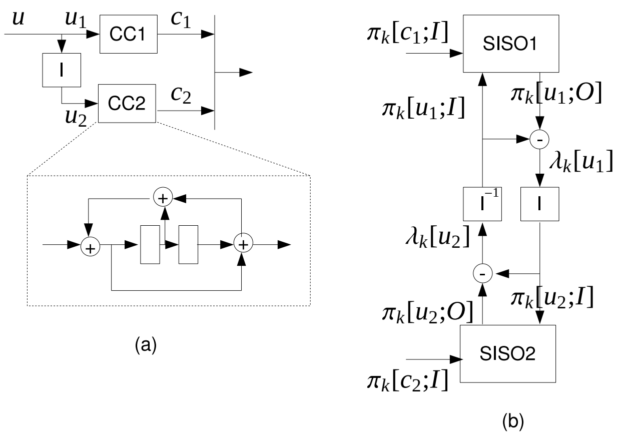

Figure 1: Example: turbo encoder (a) and turbo decoder (b)

The architecture of a turbo encoder with parallel concatenation of two CC is shown in Fig. 1 (a): a direct copy u_1 of the uncoded symbols u is applied to the first convolutional encoder CC_1, while a scrambled sequence, u_2 is received in parallel by the second encoder CC_2. The two generated outputs, c_1 and c_2 together form the coded symbol produced by the turbo encoder. An example of CC is also given in Fig. 1 (a). Part (b) of Fig. 1 describes the basic blocks of the corresponding decoder. **Here $SISO_1$ executes the BCJR algorithm on in-order data (corresponding to CC_1)**: it therefore receives $\pi_k[c_1;I]$ from the channel and $\pi_k[u_1;I]$ from the deinterleaver and generates outputs $\pi_k[u_1;O]$; extrinsic values $\lambda_k(u_1)$ are obtained according to [13]. Similarly $SISO_2$ operates on inputs $\pi_k[c_2;I]$ and $\pi_k[u_2;I]$ to compute $\pi_k[u_2;O]$; extrinsics $\lambda_k(u_2)$ are finally forwarded to the deinterleaver.

To limit the amount of memory required **by the decoding algorithm** the received blocks of data are usually partitioned in several separated windows and the BCJR algorithm is applied to each window. It has been proved that with a good choice of the window length, the performance loss due to this approach is negligible [15], [16].

Interleaving and deinterleaving components may be critical in a turbo decoder, both in terms of error correcting capabilities and in terms of implementation cost. The permutation function is usually implemented relying on a random access memory (RAM), where data are written in the natural order and read in the permuted order, or vice-versa. The generation of sequential addresses is obtained by means of a counter, but the scrambled addresses require more expensive circuits. A separate memory (RAM or ROM) can be used to store the scrambled addresses, but this solution tends to be expensive, especially when multiple turbo codes (and, consequently, multiple interleavers) need to be supported. Alternatively, when possible, the scrambled address sequence can be generated by means of a known and simple algorithm: in this case, the algorithm can be mapped onto proper hardware circuits, usually based on addition and shifting operations, which enable on the fly generation of scrambled addresses [17], [18].

3. Multiprocessor Architectures

The required limits of throughput in recent wireless standards show a tremendous rate of growing. While 2 Mbps is the throughput specification for the 3G Universal Mobile Telecommunications System (UMTS) [19], the enhanced telephony communications protocol known as High-Speed Downlink Packet Access (HSDPA) **with the "Dual Cell" technology** provides speeds of up to 84 Mbps in the downlink [20], and the WiMAX (Worldwide Interoperability for Microwave Access) standard IEEE802.16e requires a data rate of 144 Mbps [21]. Moreover, 3GPP Long Term Evolution (LTE) [22], which is considered a very promising 4G wireless technology, reaches a peak speed of 326.4 Mbps for a 4x4 antenna system, and 172.8 Mbps for a 2x2 antenna system. Data rate as high as 1 Gbps is currently expected in further evolution of LTE.

Therefore high throughput definitely is a key design issue for turbo code decoders. Single SISO architectures are unable to reach high throughput because of the iterative nature of the decoding algorithm, which results into long latency. Therefore the recourse to parallel architectures is inevitable [13].

A straightforward approach to the implementation of parallel turbo decoders is to handle concurrently several independent data blocks that are simultaneously decoded on different processing elements. This solution multiplies the costs (memories, area, and power consumption) with respect to the single processor decoder and improves the throughput by the same amount. More efficient parallel architectures can be obtained by exploiting the inherent parallelism of the iterative algorithm at different levels [23]. Parallel computation can also be introduced at the level of component decoders. In the original formulation of the whole decoding algorithm, the two allocated SISO do not work concurrently: the first component decoder operates in the domain of scrambled data, receiving them from the deinterleaver and writing them back into the interleaver. Thus, the second component decoder needs to wait until the block of data is completely processed and the interleaver memory filled with the new updated extrinsic values; at that point, it starts its processing reading from the interleaver and writing into the deinterleaver, while the first component decoder is idle. Alternatively the two component decoders can be arranged to run concurrently, leading to the shuffled decoding technique [24], where extrinsic values are exchanged as soon as they are available. This approach potentially doubles the decoding throughput, even if it may reduce the error correcting capability of the code for a fixed number of iterations. However, in [23] it is shown that for a target performance the shuffling requires in the worst case to increase the number of iterations by about 1.7.

Finally, the data block can be divided into several sub-blocks, each one decoded on an individual SISO processor. With P allocated processors, the same number of windows are processed at the same time and the throughput can be significantly improved. For a single SISO processor architecture, the throughput, T_{SS}, can be expressed as the product of the clock frequency f_{CK} times the ratio between the total number of decoded bits per data block, N, and the occupied number of clock cycles, n_{CK}. As the single SISO architecture for a binary code refines one bit reliability per clock cycle and $2It$ half-iterations are necessary before making a decision, n_{CK} should be equal to $2It \times N$. However, a slightly longer time is required to decode a block due to the internal latency of the SISO, L_{SISO}. The overall throughput is then given by

$$T_{SS} = \frac{N f_{CK}}{n_{CK}} = \frac{N f_{CK}}{2It \times (N + L_{SISO})}$$

For the multi-SISO decoder, the throughput is ideally derived observing that P SISO processors concurrently refine P bit reliability values:

$$T_{MS} = \frac{N f_{CK}}{n_{CK}} = \frac{N f_{CK}}{2It \times (N/P + L_{SISO})} \tag{1}$$

As an example, a decoder running at 300 MHz on blocks of 1024 data for 10 iterations and with a SISO latency of 5 clock cycles achieves a throughput as low as $T_{SS} = 15$ Mbps in the single SISO version, but can reach $T_{MS} =$ 115 Mbps in the multiple SISO version, with $P = 8$ processors.

However, (1) is a fairly optimistic estimation of the throughput, as it assumes that extrinsic data can be exchanged among allocated SISOs and interleaving and deinterleaving memories with no additional latency. This assumption is unrealistic, especially with large P: in this case, multiple extrinsic data need to be simultaneous scrambled and as a consequence the interleaver (and deinterleaver) memory has to be partitioned into P disjoint components. The partitioning must guarantee that P values can be accessed in parallel at each clock cycle, both in the natural order and the scrambled one. Due to the random nature of the permutation law, this partitioning is not always possible, and two or multiple accesses may collide into a single memory component.

There are a number of techniques to solve this problem [25], [26], [27], [28], [29], [30]. The collisions in the access to memories can be simply accepted and managed by inserting wait states in the SISO processors when a collision occurs: in other words, when two SISOs need to read a data from the same memory, two distinct access operations are scheduled and the decoding process is slowed down. This solution is rarely acceptable, due to its cost in terms of latency and throughput.

Another approach to the mentioned problem is to resort to hardware-aware interleavers, that are interleavers specifically designed to allow for simple partitioning up to a given degree of parallelism. Several techniques have been proposed to design such collision-free interleavers [25] [26] [27] [28] and recent standards specified codes with collision-free interleavers at least for some values of P (e.g. 3GPP LTE and WiMAX). However, other standards, such as HSDPA, use interleavers that generate collisions when partitioned and so a multi-standard implementation cannot take advantage of the collision-free property.

Finally, instead of constraining the interleaver design, the collision problem can be faced by designing proper interconnect networks capable to avoid or at least reduce conflicts in memory accesses. As this approach can handle any permutation law, its indubitable advantage is full flexibility with capability to implement decoders compliant with multiple standards and even future proof. Some papers have been published on this approach, proposing different buffering architectures. Two examples are given by the ring-shaped network proposed in [29] and by the joint space and time permutation network described in [30]. A considerable overhead is introduced in this approaches in terms of additional hardware resources, particularly if multiple interleavers must be supported.

In the context of a parallel multi-processor turbo decoder with capability of supporting generic interleavers with no collisions, the NOC (or better IntraIP-NOC [10]) is a very interesting option for the implementation of the flexible interconnect structure. NOCs bring to this specific kind of application several of their general advantages over traditional on-chip interconnects, such as enhanced scalability, separation between computation and communication, modularity, regularity of physical links and predictability of their delay.

In comparison to most InterIP-NOCs, networks adopted for turbo decoding exhibit some key differences.

1. Since the data block is partitioned into P sub-blocks of equal size and each SISO processor is assigned one sub-block, the same amount of data is exchanged by all network nodes.
2. To maximize throughput, idle times are minimized in SISO processor, so that the rate at which data are exchanged by a SISO processor is fairly constant.
3. Due to the random nature of the interleaver, the whole pattern of connections among allocated SISO processors shows little adjacency.
4. As a consequence of previous characteristics, the injected traffic load tends to be uniform.
5. Due to the iterative nature of the decoding algorithm, it is of utmost importance to minimize the latency of the single iteration.
6. The uniform traffic load and the homogeneous nature of nodes lead to simpler routing algorithms compared to the InterIP case. Moreover flow control is not required.
7. Given the permutation law of an interleaver, the traffic pattern can be derived by off-line analysis.

In order to take into account the variable delay contribution introduced for each data block by the NOC, the throughput expression of (1) must be corrected as follows

$$T_{MS} = \frac{N f_{CK}}{2It \times (N/P + L_{SISO} + L_{NOC})} \tag{2}$$

The additional latency term L_{NOC} depends on the specific code being decoded, as well as on the NOC characteristics, topology, routing algorithm and node or switch architecture. **Similarly, for double binary codes the single SISO refines two bit reliability values per clock cycle so the denominator in (2) becomes $2It\times(N/2P+L_{SISO}+L_{NOC})$. However, since the data block size for double binary codes is usually expressed in couples of bits, (2) for double binary codes becomes**

$$T_{MS} = \frac{2N_c f_{CK}}{2It \times (N_c/P + L_{SISO} + L_{NOC})} \tag{3}$$

where N_c is the number of couples per data block.

Both direct and indirect networks can be proposed for flexible parallel turbo decoders. Direct Networks typically consists of a set of nodes, each one being directly connected to small subset of other nodes in the network. The required connectivity is obtained by means of routers, which are components of the nodes and can decide the path for each data to be sent from a source node to a destination node. Instead of providing a direct connection between two nodes, indirect networks exploit switches to connects nodes.

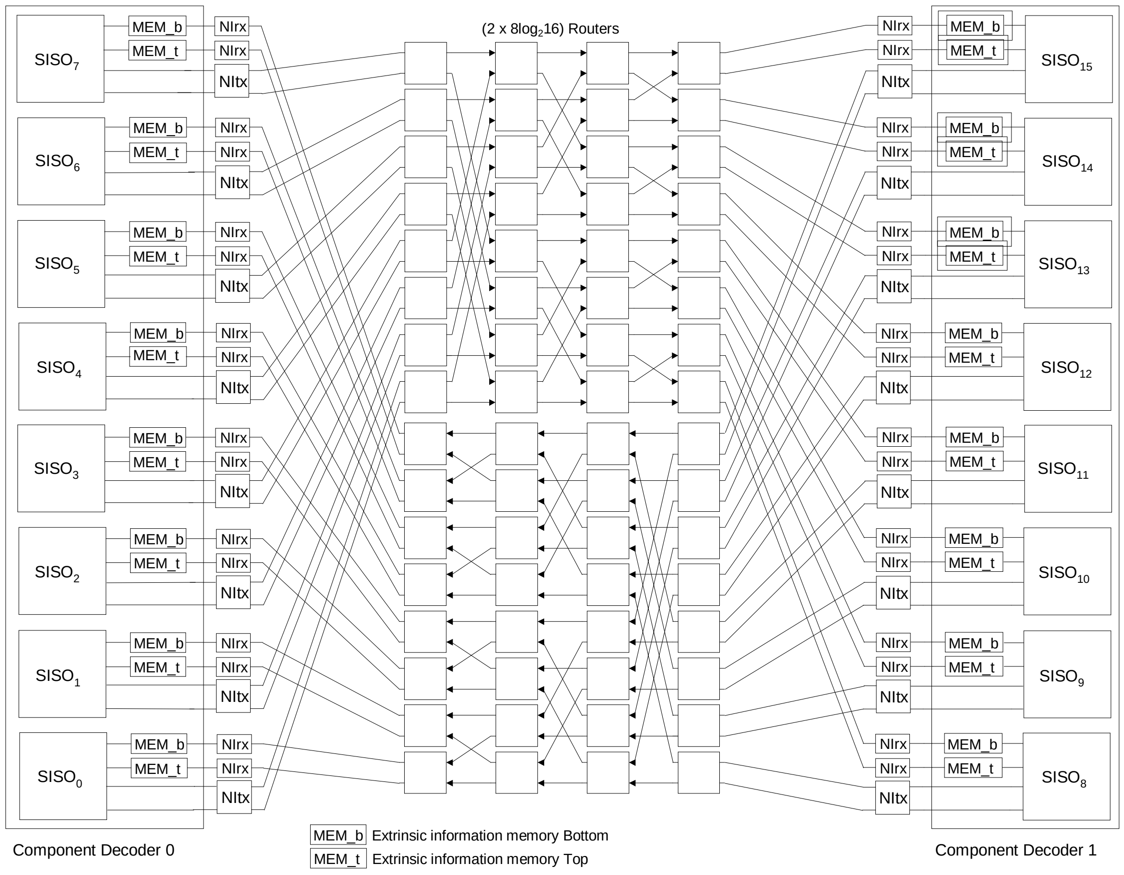

Figure 2: Schematic view of a complete multiprocessor turbo decoder with two Butterfly networks and 8 SISO by component decoder

In the following sections, both type of networks will be investigated as interconnect structures in parallel turbo decoders.

4. Indirect Networks

Indirect network topologies as Multistage Interconnection Networks (MIN) [31], have their name originated from their topology which includes several switch stages connected to each other and where the processing elements (PE) are connected on the first and last stages. From the connectivity point of view, MINs are positioned between shared bus networks and crossbars. Thus, it can happen that paths between two source/destination PEs cross, thus forming a conflict to be managed. Most of the MIN connecting P input processors to P output processors or memories are formed of $d = \log_2(P)$ stages, each one having $P/2$ switches (or routers) with 2 input and 2 output ports. Thus, the number of routers is out of $O(P \log_2(P))$ against $O(P^2)$ for crossbars.

4.1. Butterfly network

The Butterfly network [32] is a multistage interconnection network with 2-input 2-output routers and unidirectional links. The advantages of this topology are: first, the logarithmic diameter of the network ($\log_2 P$ with P the number of network input ports) which gives a number of routers equal to $\frac{P}{2} \log_2 P$; then the recursive structure of the network (a network of diameter d is obtained with two networks of diameter $d - 1$) which enables high scalability; and finally a very simple routing that uses directly the bits of the destination address for the selection of the output port at each stage of the network. However, this type of network does not have path diversity: there exists only one route between each source and each destination, which increases the risk of conflicts in the routers. To mitigate this problem, we chose to use queues to store the conflicting packets. Several architectural decisions were made in the proposed Butterfly network according to the specificities of the supported application. First of all, in the considered turbo decoder architecture, the interleaving (respectively deinterleaving) function of extrinsic information must be supported by the interconnection network. To this purpose, it is necessary that the network can vehicle any permutation of the inputs to its outputs, which is the case for the Butterfly network. Thus, through the packet addressing, the interleaving (respectively deinterleaving) of the data is performed thanks to the identifier of destination port and the destination memory write address containing extrinsic information. Assuming that the turbo decoder we consider supports the butterfly decoding scheme, a maximum of two packets will be generated by each SISO processor at the inputs of the network. This is why the number of input ports of the network is twice larger than the number of SISO processors. Figure 2 represents a Butterfly network that connects 8 SISOs each producing 2 packets for 8 memories attached to destination SISOs. Network Interfaces (NI) are used between network and processor/memories. The Destination-Tag routing technique is used. This deterministic routing uses a digit of the destination address in the header of the packets to select the output port at each router along the path from the source to the destination.

Packets integrate, besides the payload composed of the extrinsic information, a simple header field. The header includes the previously defined routing information and the destination memory address. The first field has a width of d bits, d being the diameter of the network and the second field a width of $\left\lfloor \log_2(\frac{N}{P}) \right\rfloor + 1$ bits, where N is the length of the frame to be decoded and P the number of SISOs in an interleaving domain.

Router architecture (Fig. 3.(a)) implements a simple switch with 2 input and 2 output ports with input FIFOs to store the conflicting packets. FIFO depth is determined with respect to the worst case: all input packets have the same network output port. Thus, FIFO depth is fixed to 2^i for all routers of stage i of the Butterfly network (i varies from 0 to $\log_2(P)$). The Routing and Arbitration block implements a round-robin queues serving policy. It generates the control signals of the switch matrix, FIFOs, and output packets.

4.2. Benes network

The Benes network [33] is another indirect network suitable for P to P permutation. It offers path diversity where P paths exist for each source/destination couple. Built from two Butterflies put back-to-back, its diameter is almost the double of that of Butterfly: $2\log_2 P - 1$. However, this topology avoids the conflicts if and only if all the paths have a different destination, which is not the case of parallel turbo decoding where interleaving (respectively deinterleaving) ends in potential conflicts. In order to avoid these conflicts, a time-division multiple access (TDMA)-based architecture, which implies all paths precalculation but avoids buffer usage in routers, is proposed and evaluated. In this approach, packets are scheduled at the input of the network so that at each cycle, there is at most one packet intended for a given network output port. For this purpose, a time slot is associated with each packet which defines the cycle when the packet will be sent in the network. Thus, the number of slots necessary to the packets scheduling corresponds to the maximum number of packets having the same destination. Of course, the slot assignment can be done in a random, arbitrary way or according to the application specificities. Regarding the routing algorithm, a Street-sign routing is used where the packets carry the identifier of router output of each crossed network stage. Moreover, contrary to the Butterfly network, which directly uses the network output port identifier, the routing information in Benes must be precalculated by an algorithm which determines the path of each packet. The reference algorithm for the Benes network, and more generally in the rearrangeable Clos networks, is the looping algorithm [34]. **This algorithm is executed off-line for the target interleaving law and the obtained routing information is carried by the packet header.**

4.3. Logic synthesis results

In order to evaluate the efficiency of the proposed indirect network architectures, a synthesizable RTL VHDL description of the two networks was developed. Logic synthesis was carried out with 130 nm standard cells technology. The performance metrics that we considered are area, maximum clock frequency, and throughput. Table 2 shows the results obtained in terms of area, frequency, and throughput for a network size of 16 input ports. **It is assumed that, each PE produces extrinsic information values with a rate r (injection rate): when $r = 1$, one extrinsic value per clock cycle is generated, whereas $r < 1$ means that more than one cycle is necessary to generate an extrinsic value. It is worth pointing out that, the use of r allows for modeling different clock frequencies for the PEs and the network. In particular, given the computational complexity of the BCJR algorithm [13], it is reasonable to assume that PEs can achieve a lower clock frequency than the network: $f_{PE} \leq f_{NOC}$. As a consequence, we consider in this paper only the case $r = f_{PE}/f_{NOC} \leq 1$. Impact of varying the injection rate on area and throughput are thoroughly discussed in the next section. For the designed indirect network**

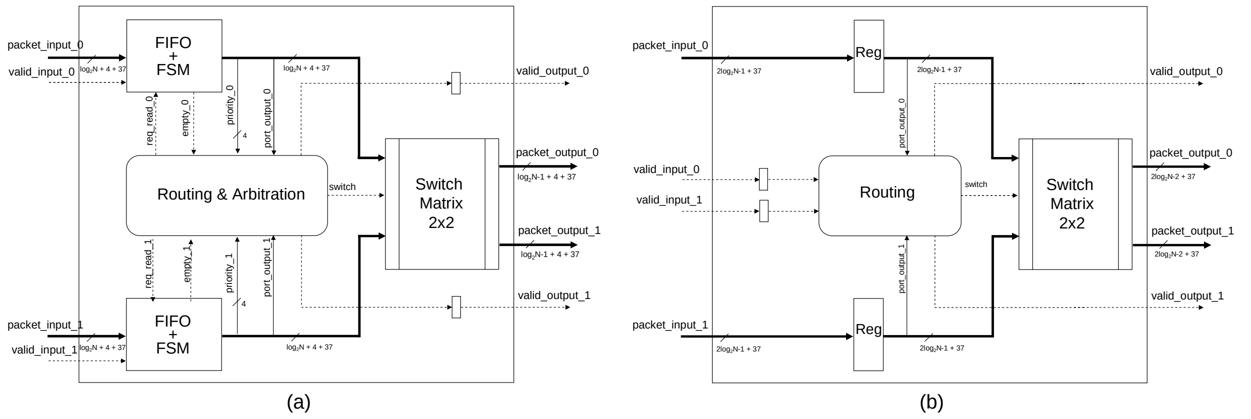

Figure 3: Router architecture: (a) Butterfly and (b) Benes

Table 2: Synthesis results of the Butterfly and Benes networks for a 130 nm standard cell technology with $P = 16$ and $r = 0.2$

	Butterfly	**Benes**
Frequency (*MHz*)	345	381
Area of one NI tx (mm^2)	0,0052	0,0098
Area of one NI rx (mm^2)	0,0045	0,0045
Area of one Router (mm^2)	0,0210	0,0065
Total Area (mm^2)	0,7506	0,4820
Throughput (*Mbps*)	138	152

topologies of Butterfly and Benes we fixed the injection rate to $r = 0.2$ **avoiding network congestion for the worst communication profile case.** We can see in table 2 how the lower complexity of Benes routers allows for higher maximum clock frequency (+10.4%) and thus higher throughput. Furthermore, total area of the Benes network is lower (−35.8%), although it comprises a higher number of routers.

5. Direct Networks

While two specific indirect networks have been analysed in the previous section, in the case of direct networks we provide a full investigation of the available design space, by analysing several combinations of choices for interleaver law, parallelism degree, node architecture, routing algorithm and topology.

A direct network based turbo decoder architecture relies on P nodes each of which contains both a processing element (PE) and a routing element (RE) (see Fig. 4). The i-th PE is a so called SISO module [35] that produces extrinsic information values with a rate r. When $r = 1$ at each time j each SISO outputs the new extrinsic information values $\lambda_{i,j}$ and stores into a local memory the extrinsic information received from the network $\lambda'_{i,j}$. On the other hand, the RE is built around a crossbar switch. It is known that node architectures can be classified in two main categories: input-queuing and output-queuing. According to [36], input-queuing leads to low implementation cost; as a consequence, it is a suited solution for direct network based turbo decoder architectures. Said M the number of input/output ports of each node, we have that the node is made of an $M \times M$ crossbar switch, whose inputs are connected to M FIFOs, and whose outputs, connected to M registers, become the node output ports (see Fig. 4). We assume that the input/output port labeled as $M - 1$ is devoted to receive/send data from/to the PE.

5.1. Node architecture

In order to send the extrinsic information on the network each node requires routing information and a routing algorithm (RA). In the case of turbo decoder architectures, the routing information is the destination node k and the memory location t where the extrinsic information will be written. This information is stored into an identifier memory and a location memory respectively. Thus, a message send by node i at time j is a packet made of a header and a payload. The header contains the destination node $k(i, j)$, whereas the payload contains $\lambda_{i,j}$ and the memory location $t(i, j)$ (see Fig. 5 (a)). In general, the RA drives the read enable of the input FIFOs, the crossbar configuration and the load signal of the output registers, based on the packet header information. In the case of single-binary turbo codes $\lambda_{i,j}$ is usually represented on a small number of bits, e.g. 8 bits. If we consider the HSDPA maximum block size $N = 5114$ and $P = 64$ we obtain that $k(i, j)$ and $t(i, j)$ require 6 and 7 bits respectively. Thus, we pay the network flexibility with a significant complexity overhead. However, the traffic generated by a multipro-

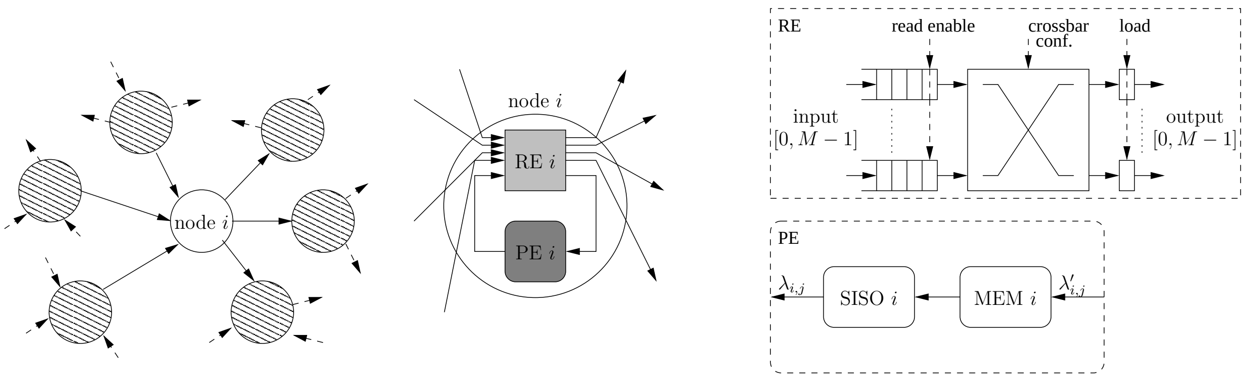

Figure 4: Node architecture block scheme

cessor turbo decoder is deterministic as it depends on the interleaving law. As a consequence, given a RA, we can compute off-line the commands for the RE and store them into a routing memory (RM). Furthermore, we can compute off-line the sequence of the memory locations at the destination side ($t'(i, j)$) and store it into a location memory. This solution leads to a packet with no routing information overhead (see Fig. 5 (b)). Unfortunately, the complexity overhead introduced by the RM can be relevant **and an external memory, e.g. a flash memory, is required to store the RE commands and $t'(i, j)$ sequences. As an example, to support all the interleavers specified by the HSDPA standard a 64 MB external flash memory is required.** To that purpose a third node architecture is derived where $t'(i, j)$ is precalculated and the packet contains only $k(i, j)$ as the header and $\lambda_{i,j}$ as the payload (see Fig. 5 (c)). In the following we will refer to the three architectures as: fully adaptive (FA) node architecture, all precalculated (AP) node architecture and partially precalculated (PP) node architecture respectively.

5.2. *Topologies*

Since the interleaver law tends to spread almost uniformly the extrinsic information among nodes, we consider topologies where all the nodes have the same degree $D = M - 1$. The same assumption has been made in previous works where ring, mesh and toroidal topologies were considered [37], [38], [39], [36]. However, since the number of clock cycles required to deliver a message have a significant impact on the turbo decoder throughput, topologies with short worst case distance among nodes are interesting candidates. To that purpose, generalized de-Bruijn [40] and generalized Kautz [41] topologies are attracting alternatives. Generalized de-Bruijn topologies are defined as follows: there is a connection between node i and node j if

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

Similarly, generalized Kautz topologies are defined as follows: there is a connection between node i and node j if

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

Thus, in this work we compare the performance and complexity of different topologies including ring (R) $D = 2$, honeycomb (H) $D = 3$ and toroidal mesh (T) $D = 4$ topologies as well as generalized de-Bruijn (B) and generalized Kautz (K) for $D = 2, 3, 4$ topologies. **As an example in Fig. 6 we show the generalized de-Bruijn ($D = 2$, $D = 3$) and the generalized Kautz ($D = 2$, $D = 4$) topologies for $P = 16$.**

5.3. *Routing algorithms*

It is worth pointing out that in turbo decoder architectures the delivery of messages impacts not only on the throughput but also on the error correction capability. In particular, message dropping to avoid collisions is not a viable solution as it can impair the decoder error correction performance. To that purpose shortest path based routing algorithms can be used both to obtain high throughput and to insure message delivery [42]. We can classify shortest path routing algorithms in two main categories: i) single shortest path (SSP) algorithms; ii) all-local shortest path (ASP) algorithms. SSP algorithms consider one shortest path from each source node i to each destination node k in the network, whereas ASP algorithms rely on the fact that more shortest paths can connect node i to node k. We call $\mathcal{N}^{i,k}$ the set of all nodes adjacent to i and placed on a shortest path between i and k. Both SSP and ASP routing algorithms can be coupled to different policies to

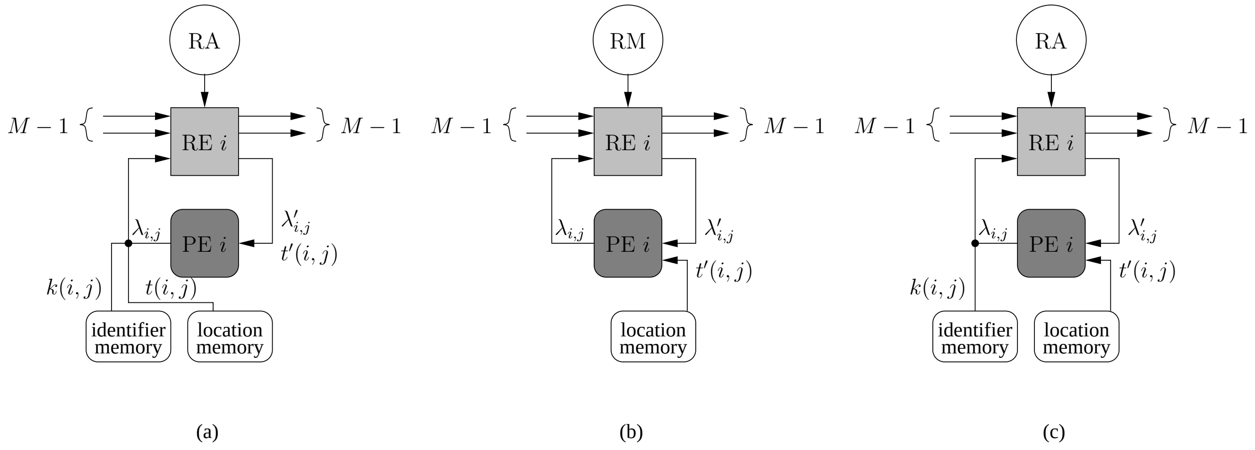

Figure 5: Node block schemes: FA (a), AP (b) and PP (c) node architectures

serve the input FIFOs. In this work we consider Round-Robin (RR) and FIFO-length (FL) serving policies with SSP and an improved version of FL with ASP. Moreover, the routing algorithms have been coupled with the node architectures as follows.

5.3.1. *FA/PP and SSP-RR*

This SSP routing algorithm is implemented with the Round-Robin circular serving policy on FA and PP node architectures.

5.3.2. *FA/PP and SSP-FL*

In this SSP routing algorithm the FL serving policy is used, namely based on the number of elements contained into each input FIFO, the longest FIFO is served first and the shortest one is served last. This routing algorithm has been implemented on FA and PP node architectures.

5.3.3. *AP ASP-FT*

This solution relies on a ASP routing algorithm coupled with an improved FL serving policy named FL with traffic-spreading (FT) that can be explained as follows. Let's define $\mathcal{I}_j^{i,l}$ as the set of input ports in a node $l \in \mathcal{N}^{i,k}$ that can receive a message from node i at time j. At time j the number of elements contained in the input FIFO associated to port $p \in \mathcal{I}_j^{i,l}$ with $l \in \mathcal{N}^{i,k}$ is $L_{j,p}^l$. The ASP-FT routing algorithm chooses $\hat{l} \in \mathcal{N}^{i,k}$ and $\hat{p} \in \mathcal{I}_j^{i,l}$ so that

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

The couples $\hat{l}$, $\hat{p}$ that satisfy (6) belong to the set $\mathcal{J}_{j,\hat{p}}^{i,\hat{l}}$. To choose only one couple in $\mathcal{J}_{j,\hat{p}}^{i,\hat{l}}$ we operate a traffic spreading based selection, namely our objective is to spread the traffic as much as possible over the network. To that purpose we use a set of counters (Q), where each counter $Q_{j,\hat{p}}^{i,\hat{l}}$ is incremented each time a message is sent from node i to node $\hat{l}$ through input port $\hat{p}$. Then, we select the couple $\tilde{l}$, $\tilde{p} \in \mathcal{J}_{j,\hat{p}}^{i,\hat{l}}$ that is associated to the least used path

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

Due to the complexity of this routing algorithm it has been used only with AP node architecture.

Unfortunately, none of the aforementioned algorithms prevents output contentions, namely two or more input FIFOs to be connected to the same output. We compare two possible approaches to solve this problem: i) delay colliding message (DCM), namely if an output port is already reserved, the colliding message is kept in its input FIFO and it is not sent; ii) send colliding message (SCM), if possible, a colliding message is sent to another output port. The aim of DCM is to reduce the number of hops to deliver a message, whereas SCM aims at reducing the size of input FIFOs.

5.4. *Experimental results*

The direct network based approach detailed in the previous sections has been tested on the design space represented in Fig. 7 for **an injection rate** $r = 0.33, 0.5, 1$. For each case we compared the throughput obtained for $f_{NOC} = 200$ **MHz** with eight iterations. **The throughput is calculated resorting to (2) and (3) for binary and double binary codes respectively where the whole n_{CK} values are obtained by SystemC cycle accurate simulations [43].** As far as

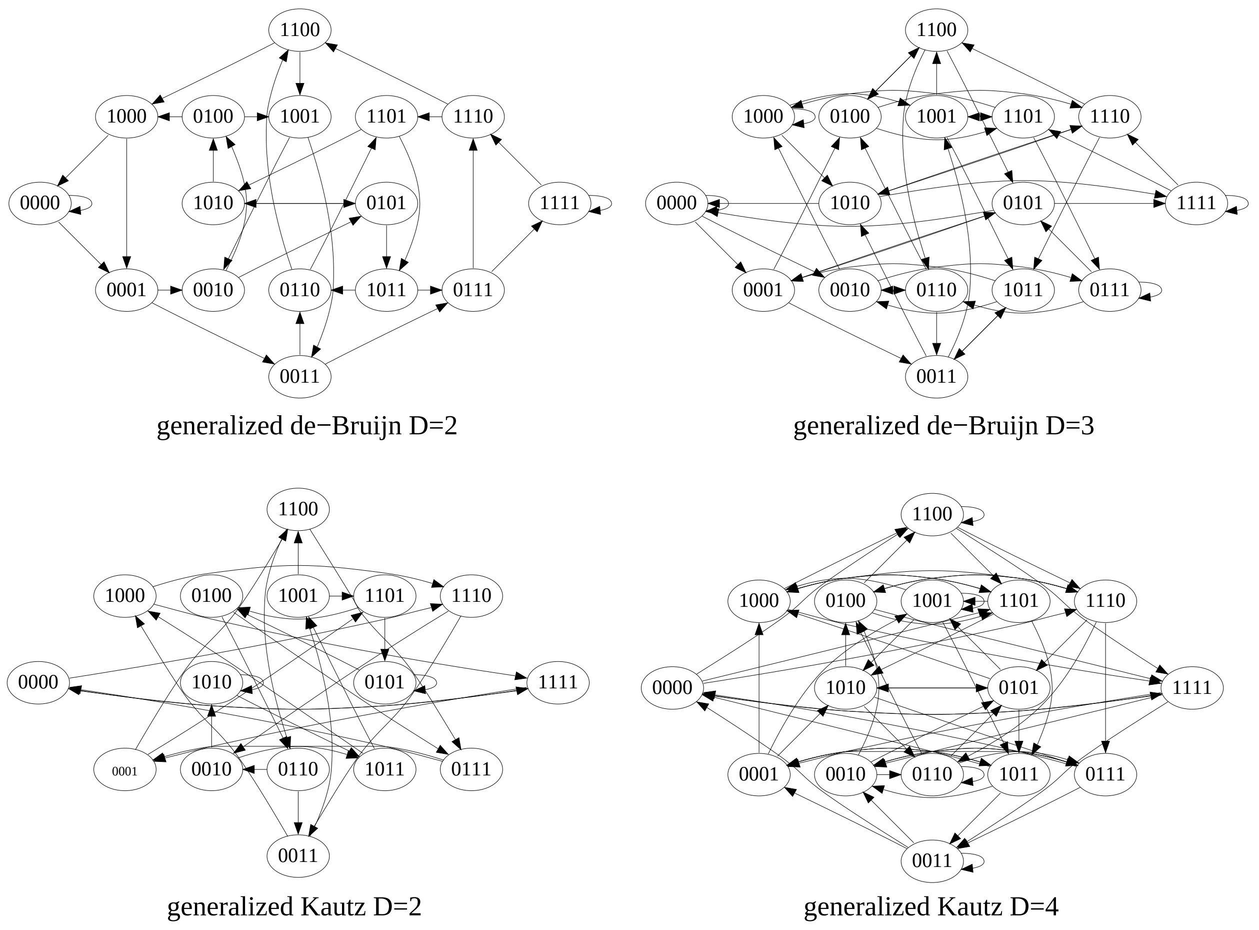

Figure 6: **Some examples of the considered topologies for $P = 16$**

the complexity is concerned, we described all the architectures as parametric blocks and we performed the logical synthesis on a 130 nm standard cells technology. Memories have been generated by means of a 130 nm memory generator. The area results concern all the nodes in the network where each node includes the blocks depicted in Fig. 5 except the PE. **It is worth pointing out that, $f_{NOC} = 200$ MHz is a conservative choice to limit buffering and to combat unreliability and reduce nonrecurrent costs. Moreover, it is in line with the clock frequency used for the indirect networks presented in section 4 and in previous works dealing with NOC based turbo decoder architectures as [36, 37, 39].**

The complexity characterization shown in this work does not consider post place and route area overhead. The actual routing overhead is expected to be quite limited for regular topologies, such as meshes and toroids, which can be implemented by means of a modular layout. For example in [44] and [45] it is shown that for regular topologies, at least at the 130 nm technology node, the area occupation of logic cells in the design gives a useful indication about the actual complexity of the NOC. Routing overhead in less regular topologies is somewhat more difficult to estimate before place and route. However, results are available in the literature showing that a low area layout can be generated for generalized de-Bruijn networks. For example in [46] and [47] 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.

The analysis of experimental results on the design space depicted in Fig. 7 shows that

1. PP node architectures achieve the same throughput as FA with a lower area.
2. Generalized de-Bruijn and generalized Kautz topologies achieve nearly the same results both in

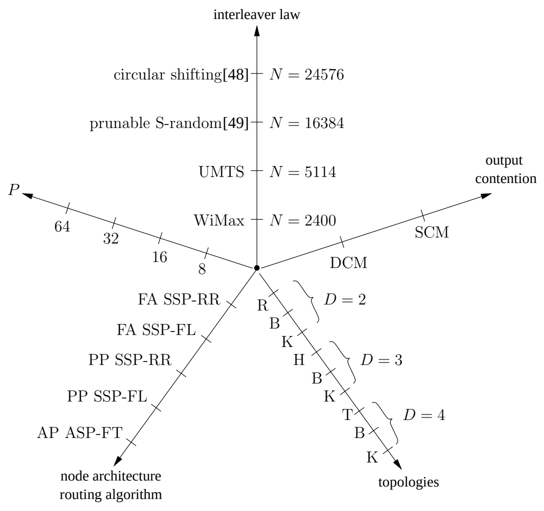

Figure 7: Direct network: design space representation

terms of throughput and complexity. Thus, for the sake of brevity in the following only Kautz results will be discussed.

3. DCM performs better that SCM, **as an example for the generalized Kautz topologies with** $D = 4$ **SCM requires the same or even greater area than DCM; moreover, SCM achieves a throughput that is about 30% less than DCM.**
4. Results tend to be clustered into two families, namely short interleavers (WiMax interleaver with $N = 2400$ and HSDPA interleaver with $N = 5114$) and long interleavers (prunable S-random interleaver with $N = 16384$ and circular shifting interleaver with $N = 24576$).

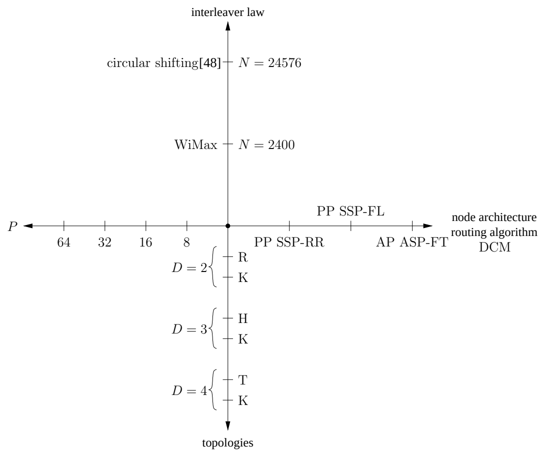

Figure 8: Direct network: pruned design space representation

As a consequence, for the sake of clarity in the following we will refer to the pruned design space defined by the aforementioned results and represented in Fig. 8. In particular, we will detail the experimental results obtained for PP node architectures with SSP routing algorithms and AP node architectures with ASP-FT routing algorithm with DCM. As far as short interleavers are concerned results referred to the HSDPA $N = 5114$ interleaver are shown; whereas for long interleavers we discuss the results obtained with the circular shifting $N = 24576$ interleaver.

Experimental results within the pruned design space are shown in Table 3 and 4 that refer to the HSDPA interleaver with $N = 5114$ and to the circular shifting interleaver with $N = 24576$ respectively. Each cell of the two tables gives the throughput in Mbps and the area in mm^2 obtained for different P and r values, routing algorithms and architectures with DCM. From the experimental results summarized in Table 3 and 4 we can observe that:

1. In most cases, topologies with D=4 achieve higher throughput with lower complexity overhead than topologies with D=2 when $r \rightarrow 1$. On the other hand, when $r < 1$ and $P > 8$ increasing D from 2 to 3 leads to a higher throughput and a smaller area, whereas increasing D from 3 to 4 leads to higher throughput but also to higher area.
2. The ASP-FT routing algorithm is the best performing solution both in terms of throughput and area when $r = 1$.
3. The routing memory overhead of the ASP-FT algorithm becomes relevant as r decreases and SSP solutions become the best solutions mainly for $P = 8$ and $P = 16$.
4. In most cases, generalized Kautz are the best performing topologies.
5. **For considered topologies with node degree D=2, the throughput tends to saturate when the interleaver size increases.**
6. **The NOC aggregate bandwidth increases with increasing values of** P**: as a consequence, the delivery time for a packet tends to be dominated by the number of hops in networks with large** P**. In this case, the topology choice tends to play a very important role, whereas the other design choices, such as routing algorithm, have a weaker impact on the throughput.**

To better highlight the results shown in Table 3 we depict in Fig. 9 the experimental results obtained with $r = 1$ **and ASP-FT routing algorithm for the HSDPA interleaver with** $N = 5114$ **where each point represents the throughput and the area achieved by**

Table 3: Throughput [Mbps]/area [mm^2] achieved for the HSDPA interleaver (N=5114) within the pruned design space

		D=2, ring				D=2, generalized Kautz			
		P=8	P=16	P=32	P=64	P=8	P=16	P=32	P=64
r=1.00	SSP-RR (PP)	60.68/1.54	70.52/3.60	79.36/5.64	84.33/7.26	54.43/1.89	74.59/3.31	105.57/4.96	159.41/7.00
	SSP-FL (PP)	60.88/1.54	72.11/3.18	80.46/4.44	82.43/5.62	58.09/1.78	81.64/3.14	117.19/4.47	171.15/6.20
	ASP-FT (AP)	68.70/1.33	77.81/2.59	83.45/3.60	83.18/4.90	58.09/1.53	81.64/2.51	117.19/3.40	171.15/4.51
r=0.50	SSP-RR (PP)	46.42/0.52	68.19/1.59	75.16/3.87	79.86/5.54	46.54/0.52	72.03/1.24	101.39/2.89	142.21/4.97
	SSP-FL (PP)	46.42/0.51	68.26/1.39	76.19/3.63	77.48/5.31	46.52/0.50	78.24/0.96	112.74/2.41	156.87/4.36
	ASP-FT (AP)	46.44/0.65	75.47/1.32	78.92/3.10	78.68/4.71	46.52/0.62	78.24/1.01	112.74/2.06	156.87/3.40
r=0.33	SSP-RR (PP)	31.12/0.50	57.72/0.78	71.27/2.90	75.21/5.10	31.16/0.49	58.33/0.74	92.91/1.58	129.53/3.53
	SSP-FL (PP)	31.11/0.49	57.62/0.73	72.35/2.64	72.35/4.95	31.14/0.49	58.30/0.71	101.63/1.28	140.96/2.92
	ASP-FT (AP)	31.11/0.65	57.75/0.98	74.64/2.47	74.29/4.50	31.14/0.62	58.30/0.88	101.63/1.36	140.96/2.54
		D=3, honeycomb				D=3, generalized Kautz			
		P=8	P=16	P=32	P=64	P=8	P=16	P=32	P=64
r=1.00	SSP-RR (PP)	60.65/1.58	102.86/2.13	98.96/4.98	192.84/6.23	84.06/0.82	111.95/1.85	188.01/3.17	279.76/5.28
	SSP-FL (PP)	60.91/1.57	105.40/2.09	101.23/4.27	186.37/5.81	90.61/0.74	126.96/1.72	194.30/3.07	291.89/5.14
	ASP-FT (AP)	68.55/1.34	140.49/1.44	144.30/3.11	240.32/4.04	90.61/0.75	142.85/1.34	207.89/2.37	298.02/3.71
r=0.50	SSP-RR (PP)	46.49/0.52	86.62/0.85	93.80/3.31	173.00/4.16	46.66/0.53	87.03/0.83	151.84/1.53	230.78/3.08
	SSP-FL (PP)	46.46/0.51	86.56/0.82	95.27/3.05	167.12/3.78	46.66/0.53	86.85/0.81	152.38/1.45	237.64/2.81
	ASP-FT (AP)	46.47/0.65	86.97/0.96	134.44/1.77	214.87/2.79	46.66/0.70	87.15/0.96	152.93/1.45	238.97/2.42
r=0.33	SSP-RR (PP)	31.14/0.50	58.38/0.79	88.54/2.04	150.94/2.97	31.20/0.52	58.51/0.78	103.86/1.39	167.56/2.63
	SSP-FL (PP)	31.12/0.49	58.38/0.77	91.45/1.83	150.59/2.76	31.19/0.52	58.49/0.77	103.86/1.36	167.12/2.54
	ASP-FT (AP)	31.14/0.64	58.49/0.95	101.39/1.55	163.07/2.44	31.19/0.66	58.49/0.93	103.94/1.43	167.56/2.33
		D=4, toroidal mesh				D=4, generalized Kautz			
		P=8	P=16	P=32	P=64	P=8	P=16	P=32	P=64
r=1.00	SSP-RR (PP)	66.73/1.09	92.71/1.92	104.97/3.94	181.61/5.58	75.29/0.92	156.49/1.32	191.11/2.84	356.13/4.49
	SSP-FL (PP)	70.40/1.03	93.46/1.83	113.14/3.40	190.82/5.21	83.24/0.87	163.70/1.21	199.45/2.83	372.74/4.30
	ASP-FT (AP)	90.23/0.74	162.04/1.15	228.30/2.22	341.84/3.49	90.61/0.74	163.70/1.12	246.81/1.99	372.74/3.31
r=0.50	SSP-RR (PP)	46.59/0.54	86.33/0.94	98.80/2.48	161.02/3.78	46.68/0.55	87.27/0.89	152.38/1.64	246.81/3.16
	SSP-FL (PP)	46.58/0.53	86.68/0.90	107.62/2.20	173.24/3.54	46.68/0.54	87.27/0.87	152.38/1.60	245.87/3.12
	ASP-FT (AP)	46.59/0.71	87.03/1.09	152.02/1.67	237.64/2.77	46.68/0.73	87.27/1.04	152.93/1.63	245.87/2.77
r=0.33	SSP-RR (PP)	31.17/0.52	58.41/0.87	94.91/1.68	147.29/3.14	31.21/0.54	58.57/0.85	103.94/1.54	170.24/2.94
	SSP-FL (PP)	31.16/0.52	58.43/0.85	98.80/1.59	154.04/3.04	31.21/0.53	58.57/0.84	104.11/1.51	170.01/2.92
	ASP-FT (AP)	31.17/0.66	58.49/1.03	103.35/1.63	166.25/2.71	31.21/0.69	58.57/1.01	104.11/1.59	170.01/2.69

Table 4: Throughput [Mbps]/area [mm^2] achieved for the circular shifting interleaver (N=24576) within the pruned design space

		D=2, ring				D=2, generalized Kautz			
		P=8	P=16	P=32	P=64	P=8	P=16	P=32	P=64
r=1.00	SSP-RR (PP)	62.56/6.58	72.23/15.73	81.22/24.45	87.04/30.37	56.62/8.43	77.26/14.10	116.01/20.56	169.96/26.72
	SSP-FL (PP)	62.57/6.84	73.89/13.90	82.98/18.67	88.25/22.11	59.52/8.09	83.52/13.50	125.31/18.55	183.79/23.53
	ASP-FT (AP)	71.48/5.93	81.57/11.29	88.17/15.12	91.14/19.36	59.52/6.94	83.52/10.74	125.31/13.90	183.79/16.74
r=0.50	SSP-RR (PP)	49.12/1.80	72.36/6.00	80.26/15.71	86.11/21.02	49.13/1.79	77.37/4.10	114.99/10.17	165.12/16.37
	SSP-FL (PP)	49.12/1.78	73.42/5.10	82.28/14.68	87.29/20.48	49.13/1.78	86.74/2.98	129.59/8.00	186.75/13.78
	ASP-FT (AP)	49.12/2.50	82.40/4.93	87.48/12.55	90.23/18.42	49.13/2.39	86.74/3.43	129.59/7.03	186.75/10.80
r=0.33	SSP-RR (PP)	32.78/1.76	63.67/2.09	79.52/11.40	85.06/19.10	32.78/1.76	63.83/2.06	111.61/4.13	162.20/9.53
	SSP-FL (PP)	32.78/1.76	63.68/2.04	81.51/9.65	86.27/18.43	32.77/1.75	63.81/2.01	123.57/2.66	186.58/6.51
	ASP-FT (AP)	32.78/2.51	63.67/3.37	86.71/9.32	88.90/17.17	32.77/2.44	63.81/2.99	123.57/3.71	186.58/6.45
		D=3, honeycomb				D=3, generalized Kautz			
		P=8	P=16	P=32	P=64	P=8	P=16	P=32	P=64
r=1.00	SSP-RR (PP)	63.28/6.51	107.23/8.07	103.96/20.19	219.19/21.14	87.61/2.86	118.27/6.64	210.05/10.39	332.65/16.23
	SSP-FL (PP)	64.03/6.67	109.73/8.31	106.61/16.64	214.53/20.29	97.37/2.06	135.54/6.08	220.22/10.86	350.29/16.37
	ASP-FT (AP)	72.48/5.74	152.42/5.28	160.67/11.92	313.79/13.45	97.37/2.29	153.26/4.55	239.53/8.07	375.55/11.66
r=0.50	SSP-RR (PP)	49.12/1.80	95.52/2.19	102.95/12.05	213.78/10.75	49.16/1.81	95.63/2.15	185.62/2.91	322.18/5.24
	SSP-FL (PP)	49.12/1.79	95.48/2.13	106.06/10.62	208.55/9.56	49.16/1.81	95.69/2.10	185.68/2.76	346.34/4.32
	ASP-FT (AP)	49.12/2.50	95.60/3.05	163.71/4.91	312.99/6.14	49.16/2.63	95.69/3.02	185.62/3.54	348.10/4.60
r=0.33	SSP-RR (PP)	32.78/1.77	63.84/2.08	102.14/5.83	205.69/5.13	32.79/1.79	63.89/2.08	124.30/2.68	235.31/3.96
	SSP-FL (PP)	32.78/1.76	63.84/2.06	108.17/4.43	216.03/4.54	32.79/1.79	63.89/2.07	124.27/2.62	235.58/3.78
	ASP-FT (AP)	32.78/2.50	63.85/2.99	123.62/4.18	233.35/5.14	32.79/2.47	63.89/2.95	124.30/3.62	235.58/4.68
		D=4, toroidal mesh				D=4, generalized Kautz			
		P=8	P=16	P=32	P=64	P=8	P=16	P=32	P=64
r=1.00	SSP-RR (PP)	70.18/4.23	97.20/6.89	110.05/13.77	202.77/16.59	77.99/3.34	174.15/3.71	215.50/8.33	493.89/10.22
	SSP-FL (PP)	73.67/4.04	96.02/6.58	117.84/11.43	214.68/15.00	86.09/3.21	184.17/2.97	232.38/8.38	516.74/9.82
	ASP-FT (AP)	96.57/2.36	184.12/3.13	275.76/6.43	471.89/9.19	97.11/2.35	184.17/3.06	298.83/5.09	516.74/7.61
r=0.50	SSP-RR (PP)	49.14/1.82	95.26/2.32	109.34/7.13	198.32/7.77	49.16/1.82	95.75/2.19	185.62/2.95	350.48/4.47
	SSP-FL (PP)	49.14/1.80	95.48/2.22	119.74/6.10	213.85/7.03	49.16/1.82	95.75/2.17	185.96/2.89	350.89/4.28
	ASP-FT (AP)	49.14/2.66	95.61/3.34	185.17/4.03	347.12/5.17	49.16/2.72	95.75/3.15	185.90/3.83	350.89/4.96
r=0.33	SSP-RR (PP)	32.78/1.79	63.85/2.17	110.01/3.59	196.55/5.02	32.79/1.81	63.91/2.15	124.37/2.82	236.04/4.17
	SSP-FL (PP)	32.78/1.78	63.85/2.15	123.10/2.97	216.80/4.56	32.79/1.80	63.92/2.14	124.40/2.78	235.94/4.06
	ASP-FT (AP)	32.78/2.42	63.85/3.03	124.00/3.95	234.15/5.26	32.79/2.51	63.92/3.02	124.40/3.72	235.94/4.96

Table 5: **Power consumption [mW] results: average power consumption per node**

		D=2, R	D=2, K	D=3, H	D=3, K	D=4, T	D=4, K
	SSP-RR (PP)	8.02	7.71	7.43	6.37	7.16	5.94
r=1.00	SSP-FL (PP)	6.17	6.79	6.98	6.23	6.92	5.87
	ASP-FT (AP)	7.87	8.49	8.62	8.16	8.46	8.16
	SSP-RR (PP)	6.09	5.42	5.11	3.88	5.14	4.44
r=0.50	SSP-FL (PP)	5.81	4.73	4.69	3.59	5.02	4.51
	ASP-FT (AP)	7.53	6.45	6.27	5.67	6.91	6.91
	SSP-RR (PP)	5.44	3.67	3.75	3.33	4.33	4.10
r=0.33	SSP-FL (PP)	5.41	3.11	3.53	3.26	4.38	4.23
	ASP-FT (AP)	7.13	4.83	5.46	5.37	6.60	6.67

a topology for a given P. Similarly, in Fig. 10 and 11 we show the experimental results obtained with $r = 1$, $P = 64$ and $r = 0.33$, $P = 16$ respectively for the HSDPA interleaver with $N = 5114$. Moreover, in Table 5 we show the average power consumption per node obtained for each topology. As it can be observed, AP architectures are almost always the most power demanding ones, due to the presence of RMs. On the contrary PP architectures show a lower power consumption; this reduction is particularly significant when topologies with short distances among nodes, as the generalized Kautz ones, are employed. As it can be observed, generalized Kautz

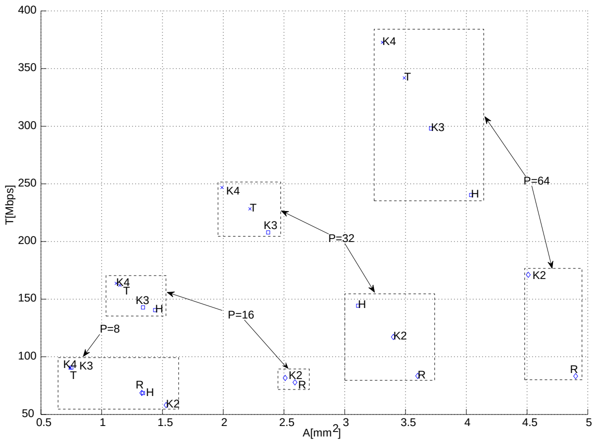

Figure 9: **HSDPA with N = 5114 throughput/area comparison of different topologies for the case r = 1, ASP-FT routing algorithm, DCM approach**

topologies with $D = 4$ are always the best solutions to achieve high throughput with minimum area overhead. Moreover, we observe that increasing D has the effect of reducing network congestion, which has two positive consequences: higher throughput and smaller size of input FIFOs. However, the saved area benefit is partially mitigated by the increased complexity of the routing algorithm, as highlighted in the first part of Table 6 for PP node architectures: for example, K topology requires 23.81 mm^2 for FIFOs and 0.11 mm^2 for RA/RM when

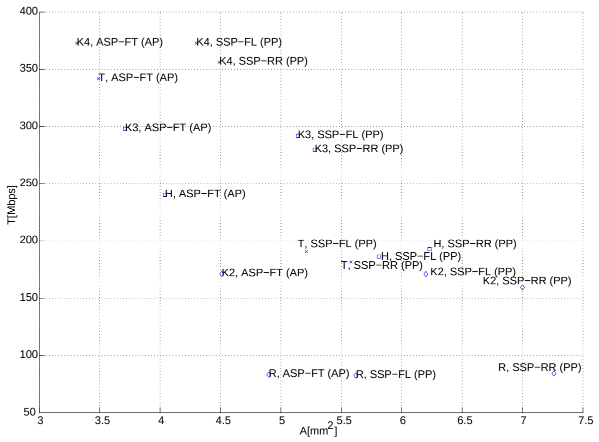

Figure 10: **HSDPA with N = 5114 throughput/area comparison of different topologies for the case r = 1, P = 64, DCM approach**

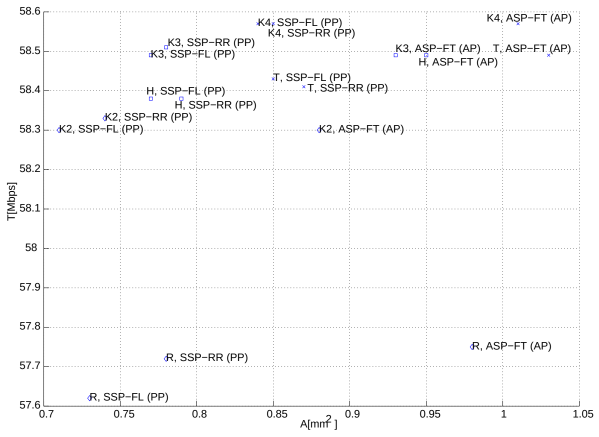

Figure 11: **HSDPA with N = 5114 throughput/area comparison of different topologies for the case r = 0.33, P = 16, DCM approach**

$D = 2$, whereas areas of 6.86 mm^2 and 0.36 mm^2 are occupied when $D = 4$. On the contrary, if an AP architecture is used for switching nodes, the routing memory size progressively becomes smaller with increasing D, due to the higher number of packets that can be switched at each clock cycle. For example, in the second part of Table 6 (AP node architecture), the area occupied by the routing memory (RA/RM) reduces with increasing D. We can further observe that for $r = 1$ the ASP-FT routing algorithm is the best solution, whereas for $r < 1$ SSP routing algorithms, implemented as PP node architectures, tend to achieve the same performance as the ASP-FT routing algorithm with lower complexity overhead. An interesting phenomenon that arises increasing the interleaver size is the performance saturation that can be observed in the Table 4 for $D = 2$ topologies, namely the throughput tends to saturate and increasing r has the effect of augmenting the area with a negligible increase or even with a decrease of throughput. As an example, the generalized Kautz topology with $P = 64$ and ASP-FT routing algorithm achieves more than 180 Mbps with $r = 1$, $r = 0.5$, $r = 0.33$. However, the solution with the smallest area is the one obtained with $r = 0.33$. The throughput flattening of low D topologies can be explained by observing that high values of r tend to saturate the network. Furthermore, high values of r lengthen the input FIFOs as highlighted in Table 7, where the total area of the network is given as the breakdown of the building blocks, namely the input FIFOs, the crossbars (CB), the output registers, the routing algorithm/memory (RA/RM), the identifier memory (IM) and the location memory (LM) is given for some significant cases: the highest throughput (light-gray), the highest area (mid-gray), and lowest area (dark-gray) points for each D value in Table 4.

Table 6: Area occupied by input FIFOs and routing algorithm/memory (RA/RM) for different topologies and routing algorithms with $P = 64$, circular shifting interleaver ($N = 24576$), $r = 1$ and DCM

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

6. Comparisons

Given the different solutions presented in the two previous sections as methods to interconnect multiple processing elements in a multi-standard parallel turbo decoder, it is very interesting to compare them with other approaches towards single standard implementations or even architectures designed to support a limited number of codes. However such a comparison tends to be quite difficult to carry out, as the flexibility in the computation to be supported always comes at a relevant cost, both in terms of occupied area and achievable throughput. Direct comparisons between architectures with different degrees of flexibility risk then to be unfair. We report in Table 8 some of the analyzed NOC based architectures together with a few previous implementations of known turbo decoders. For each case, occupied area and achieved throughput is given and all data have been converted to a 130 nm CMOS technology: for the sake of simplicity, when starting from values related to a technology with feature size α, rough scaling factors $(130/\alpha)^2$ and $\alpha/130$ have been used in the conversion for area and throughput respectively. The purpose of the presented data is twofold:

1. To compare NOC based solutions studied in this work with other interconnect architectures used in parallel turbo decoders (first and second parts of the Table); in this case, comparisons can be made directly on area and throughput data.
2. To obtain an estimation of the flexibility cost associated to the use of NOC structures (first and third parts of the Table). In this case, direct area comparisons do not make sense, as the complexity values given in Table 8 for NOC based decoders are referred to only the interconnect structures and do not include the area contribution of processing elements. On the contrary, the whole occupied area is shown for considered decoders. However one useful lesson that can be learned from those date is the overhead associated to the NOC structure.

In the first part of Table 8, four NOCs with parallelism degree equal to 16 are compared in terms of achieved throughput for 8 decoding iterations, occupied area and throughput to area ratio: Butterfly and Benes indirect networks, toroidal mesh and generalized Kautz among direct networks. For the two latter cases, the reported synthesis data refer to $D = 4$, $r = 1$, $N = 5114$ and ASP-FT configuration. All four cases provide fairly high throughput at a limited cost of occupied area and throughput to area ratio is quite high, with the highest value shown by the Benes topology. Two different interconnect architectures are considered in the second part of the Table. These two structures are able to adapt to multiple codes, with different interleaving laws, but they achieve significantly lower throughput to area ratios. Finally in the third part of the Table, we give implementation data for three implementations with no, or limited, flexibility. In these cases, area figures refer to the whole

Table 7: Hardware resources breakdown for the circular shifting interleaver with N=24576 and DCM: some significant points

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

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

Table 8: Implementation results of some flexible interconnect structures and complete turbo decoders. Interconnect architectures have parallelism degree P = 16; the two direct networks have D = 4, r = 1, N = 5114 and ASP-FT configuration. Throughput is for 8 decoding iterations. All data have been converted to a 130 nm CMOS technology.)

NOC architecture (degree 16)	clock freq. (MHz)	NOC area (mm^2)	Throughput (8 it.) (Mbps)	Throughput to Area Ratio
Butterfly	345	1.5	138	92
Benes	381	0.96	152	158
toroidal mesh	200	1.2	162	135
generalized Kautz	200	1.1	164	149
Interconnect architecture (degree 16)	**clock freq. (MHz)**	**interconnect area (mm^2)**	**Throughput (8 it.) (Mbps)**	**Throughput to Area Ratio**
[29]	184	1.5	19.6	13
[50]	-	9.8	91	9.2
Decoder architecture	**clock freq. (MHz)**	**decoder area (mm^2) (mm^2)**	**Throughput (8 it.) (Mbps)**	**NOC overhead (gen. Kautz area over decoder area)**
[51]	150	8.4	75	13%
[52]	250	10.7	187	10%
[53]	352	10	352	11%

decoders and the last column reports, for each decoder, the ratio between the NOC area occupied by the Kautz topology and the decoder area. This ratio is a measure of the potential overhead introduced by the NOC with respect to the global cost of a decoder: it represents the percentage increment of complexity that would be caused by the use of the NOC as a method to make the considered decoder flexible, that is able to handle any interleaving law. In the three considered cases, this overhead is not higher than 13%, which makes a NOC a very promising solution for the implementation of flexible interconnect structures in parallel turbo decoders.

7. Conclusions

In this paper we presented two classes of solutions to achieve high throughput and flexibility in multiprocessor turbo decoder architectures. Both solutions rely on the network on chip approach. One solution investigates indirect networks, as butterfly and Benes topologies. The other solution relies on direct networks, proposing generalized Kautz topologies as an effective alternative. Experimental results show that a network-on-chip based decoder made of 16 processing elements achieves a throughput of 164 Mbps, while the area required by the network compares favorably with previously published works.

Acknowledgement

This work has been supported by NEWCOM++ NoE.

References

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

[2] Y. Lin, H. Lee, M. Woh, Y. Harel, S. Mahlke, T. Mudge, C. Chakrabarti, K. F. Soda, A low-power architecture for software radio, in: 33rd International Symposium on Computer Architecture, 2006, pp. 89–101.

[3] C. Pan, N. Bagherzadeh, A. Kamalizad, A. Koohi, Design and analysis of a programmable single-chip architecture for DVB-T base-band receiver, in: Design, Automation and Test in Europe Conference and Exhibition, 2003, pp. 468–473.

[4] M. Hosemann, R. Habendorf, , G. P. Fettweis, Hardware-software codesign of a 14.4mbit - 64 state - viterbi decoder for an application- specifi c digital signal processor, in: IEEE Workshop on Signal Processing Systems, 2003, pp. 45–50.

[5] P. Guerrier, A. Greiner, A generic architecture for on-chip packet-switched interconnections, in: Design, Automation and Test in Europe Conference and Exhibition, 2000, pp. 250–256.

[6] W. J. Dally, B. Towels, Route packets, not wires: On-chip interconnection networks, in: Design Automation Conference, 2001, pp. 684–689.

[7] L. Benini, G. D. Micheli, Networks on chips: a new soc paradigm, IEEE Computer 35 (1) (2002) 70–78.

[8] S. Kumar, A. Jantsch, J. P. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, A. Hemani, A network on chip architecture and design methodology, in: IEEE Computer Society Annual Symposium on VLSI, 2002, pp. 105–112.

[9] L. Benini, Application specifi c NoC design, in: Design, Automation and Test in Europe Conference and Exhibition, 2006, pp. 1330–1335.

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

[11] L. Bahl, J. Cocke, F. Jelinek, J. Raviv, Optimal decoding of linear codes for minimizing symbol error rate, IEEE Transactions on Information Theory 20 (3) (1974) 284–287.

[12] G. Masera, Turbo Code applications: a journey from a paper to realization, Springer, 2005, Ch. 14 - "VLSI for turbo codes", pp. 347–382.

[13] E. Boutillon, C. Douillard, G. Montorsi, Iterative decoding of concatenated convolutional codes: Implementation issues, Proceedings of the IEEE 95 (6) (2007) 1201–1227.

[14] S. Papaharalabos, P. T. Mathiopoulos, G. Masera, M. Martina, On optimal and near-optimal turbo decoding using generalized max* operator, IEEE Communications Letters 13 (7) (2009) 522–524.

[15] S. Benedetto, D. Divsalar, G. Montorsi, F. Pollara, Algorithm for continuous decoding of turbo codes, IET Electronics Letters 32 (4) (1996) 314–315.

[16] A. J. Viterbi, An intuitive justifi cation and a simplifi ed implementation of the MAP decoder for convolutional codes, IEEE Journal on Selected Areas in Communications 16 (2) (1998) 260–264.

[17] Z. Wang, Q. Li, Very low-complexity hardware interleaver for turbo decoding, IEEE Transactions on Circuits and Systems II 54 (7) (2007) 636–640.

[18] M. Martina, M. Nicola, G. Masera, Hardware design of a low complexity, parallel interleaver for wimax duo-binary turbo decoding, IEEE Communications Letters 12 (11) (2008) 846–848.

[19] http://www.3gpp.org.

[20] http://www.3gpp2.org.

[21] IEEE Std 802.16, part 16: air interface for fi xed broadband wireless access systems (Oct. 2004).

[22] http://www.3gpp.org/highlights/lte/lte.htm.

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

[24] J. Zhang, M. P. C. Fossorier, Shuffled iterative decoding, IEEE Transactions on Communications 53 (2) (2005) 209–213.

[25] C. Douillard, C. Berrou, Turbo codes with rate-m/(m+1) constituent convolutional codes, IEEE Transactions on Communications 53 (3) (2005) 268–278.

[26] J. Kwak, K. Lee, Design of dividable interleaver for parallel decoding in turbo codes, IET Electronics Letters 38 (22) (2002) 1362–1364.

[27] A. Nimbalker, T. Blankenship, B. Classon, T. Fuja, D. C. Jr., Inter-window shuffle interleavers for high throughput turbo decoding, in: 3rd International Symposium on Turbo Codes and Related Topics, 2005, pp. 335–358.

[28] R. Dobkin, M. Peleg, R. Ginosar, Parallel interleaver design and VLSI architecture for low-latency MAP turbo decoders, IEEE Transactions on VLSI 13 (4) (2005) 427–438.

[29] F. Gilbert, M. Thul, N. Wehn, Communication centric architectures for turbo-decoding on embedded multiprocessors, in: Design, Automation and Test in Europe Conference and Exhibition, 2003, pp. 356–361.

[30] A. Tarable, S. Benedetto, Mapping interleaving laws to parallel turbo decoder architectures, IEEE Communications Letters 8 (3) (2004) 162–164.

[31] R. Bergevin, Architectures et traitement parallles, www.gel.ulaval.ca/~bergevin/enseignement.html (2007).

[32] J. W. Cooley, J. W. Tukey, An algorithm for the machine calculation of complex fourier series, Mathematics of Computation 19 (90) (1965) 297–301.
URL `http://www.jstor.org/stable/2003354`

[33] V. E. Benes, Mathematical Theory of Connecting Networks and Telephone Traffic, Academic Press, 1965.

[34] D. C. Opferman, N. T. Tsao-Wu, On a class of rearrangeable switching networks, Bell System Technical Journal 50.

[35] S. Benedetto, D. Divsalar, G. Montorsi, F. Pollara, Soft-input soft-output modules for the construction and distributed iterative decoding of code networks, European Transactions on Telecommunications 9 (2) (1998) 155–172.

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

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

[38] M. J. Thul, F. Gilbert, N. Wehn, Concurrent interleaving architectures for high-throughput channel coding, in: IEEE International Conference on Acoustics, Speech and Signal Processing, 2003, pp. 613–616.

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

[40] M. Imase, M. Itoh, Design to minimize diameter on building-block network, IEEE Transactions on Computers 30 (6) (1981)

439–442.

[41] M. Imase, M. Itoh, A design for directed graphs with minimum diameter, IEEE Transactions on Computers 32 (8) (1983) 782–784.

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

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

[44] F. Angiolini, P. Meloni, S. M. Carta, L. Raffo, L. Benini, A layout-aware analysis of networks-on-chip and traditional interconnects for MPSoCs, IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems 26 (3) (2007) 421–434.

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

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

[47] M. Hosseinabady, M. R. Kakoee, J. Mathew, D. K. Pradhan, De Bruijn graph as a low latency scalable architecture for energy efficient massive NoCs, in: Design, Automation and Test in Europe Conference and Exhibition, 2008, pp. 1370–1373.

[48] L. Dinoi, S. Benedetto, Variable-size interleaver design for parallel turbo decoder architectures, IEEE Transactions on Communications 53 (11) (2005) 1833–1840.

[49] S. Dolinar, D. Divsalar, Weight distributions for turbo codes using random and nonrandom permutations, TDA Progress Report 42-122 (1995) 56–65.

[50] I. Ahmed, C. Vithanage, Dynamic reconfiguration approach for high speed turbo decoding using circular rings, in: ACM Great Lakes symposium on VLSI, 2009, pp. 475–480.

[51] M. May, T. Ilnseher, N. Wehn, W. Raab, A 150 Mbit/s 3GPP LTE turbo code decoder, in: Design, Automation and Test in Europe Conference and Exhibition, 2010, pp. 1420–1425.

[52] J. H. Kim, I. C. Park, A unified parallel radix-4 turbo decoder for mobile wimax and 3gpp-lte, in: IEEE Custom Intergrated Circuits Conference, 2009, pp. 487–490.

[53] P. Urard, L. Paumier, M. Viollet, E. Lantreibecq, H. Michel, S. Muroor, B. Coates, M. Gupta, A generic 350 Mb/s turbo-codec based on a 16-states SISO decoder, in: IEEE International Solid-State Circuits Conference, 2004, pp. 424–536.

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