Exploiting generalized de-Bruijn/Kautz topologies for flexible iterative channel code decoder architectures

Exploiting Generalized de-Bruijn/Kautz Topologies for Flexible Iterative Channel Code Decoder Architectures

Carlo Condo[a], Maurizio Martina[a,*], Massimo Ruo Roch[a], Guido Masera[a]

[a] *Electronics and Telecommunications Department, Politecnico di Torino, C.so Duca degli Abruzzi 24, 10129 Torino, Italy*

Abstract

Modern iterative channel code decoder architectures have tight constrains on the throughput but require flexibility to support different modes and standards. Unfortunately, flexibility often comes at the expense of increasing the number of clock cycles required to complete the decoding of a data-frame, thus reducing the sustained throughput. The Network-on-Chip (NoC) paradigm is an interesting option to achieve flexibility, but several design choices, including the topology and the routing algorithm, can affect the decoder throughput. In this work logarithmic diameter topologies, in particular generalized de-Bruijn and Kautz topologies, are addressed as possible solutions to achieve both flexible and high throughput architectures for iterative channel code decoding. In particular, this work shows that the optimal shortest-path routing algorithm for these topologies, that is still available in the open literature, can be efficiently implemented resorting to a very simple circuit. Experimental results show that the proposed architecture features a reduction of about 14% and 10% for area and power consumption respectively, with respect to a previous shortest-path routing-table-based design.

Keywords: Network on Chip, low latency, de-Bruijn, Kautz, turbo codes, LDPC codes

1. Introduction

Flexible and scalable interconnection systems have become of paramount importance in modern System-on-Chip (SoC) designs. In this context, the Network-on-Chip (NoC) approach [1, 2, 3, 4] has been proposed as an interesting solution. Several research efforts have been spent in the last years not only to improve NoC reliability and efficiency, but also to reduce complexity and save power consumption by means of several different techniques, such as [5, 6, 7] and including wireless NoCs [8] that exploit Ultra-Wide-Band technology [9]. Several design parameters, as NoC topology, Routing Algorithm (RA)

*Maurizio Martina
Email address: maurizio.martina@polito.it (Maurizio Martina)

and scheduling policy [10, 11, 12] affect the complexity and the power consumption of the NoC. Most of the works published in the literature (e.g. [13, 14]) highlight that 2D mesh and 2D mesh-like topologies are among the best topologies for tile-based ASIC implementation. However, some recent publications have shown that the general NoC approach can not be straightforwardly applied to high throughput and low latency applications. Indeed, in [15] application-specific NoCs are proposed, i.e. the NoC used to interconnect different IPs is tailored around the application. This concept is further developed in [16], where intra-IP NoCs are described. The intra-IP NoC is proposed as a very low complexity interconnection structure to build flexible IPs, whose computation relies on a parallel architecture made of several processing elements (PEs). Examples of high throughput and low latency architectures that can take advantage of intra-IP NoCs are mainly in the field of baseband processing for telecommunications, such as baseband multiple-input-multiple-output systems [17] and iterative channel code decoders [18]. Both applications require flexibility to support different coding modes and standards. In particular, in [18] both turbo and Low-Density-Parity-Check (LDPC) codes from several different standards are supported. It is worth noting that since the throughput of iterative channel code decoders is inversely proportional to the latency of the architecture, reducing the latency of the NoC is a viable solution to increase the throughput. In these cases previous works [19, 20] show by simulation that the minimization of the number of clock cycles spent to send a message from the source node to the destination node is of paramount importance and the maximum distance between two nodes, i.e. the diameter of the topology, should be minimized to reduce the delivery time. Recently, logarithmic diameter topologies, such as de-Bruijn [21] and Kautz [22] topologies, have gained popularity in the research community [20, 23, 24, 25, 26, 27] as viable alternatives to the well known mesh and mesh-like solutions. Indeed, logarithmic-diameter topologies like de-Bruijn and Kautz manage to guarantee short minimum-distance path while showing a regular structure, similar to that of the simpler mesh. Furthermore, [28] shows an algorithm to build optimal VLSI layout for de-Bruijn topologies. Another important characteristic of de-Bruijn and Kautz topologies is the self-routing property [29], that is exploited in [30] to develop a shortest-path RA to connect any pair of nodes. Even if [30] solved the problem of building an optimal shortest-path RA for these topologies, low complexity hardware implementations are still of interest in the context of NoC [23, 24, 25, 26, 27]. A well known flexible solution to support most topologies and RAs employs routing-tables. However, it has been shown that this approach does not scale in terms of latency and area. This problem has been faced in [31] and further developed in [14, 32], where Logic-Based-Distributed-Routing (LBDR) for mesh-derived topologies is proposed. For a survey on LBDR the reader can refer to [33].

Inspired by the LBDR approach, this work proposes to exploit the optimal shortest-path RA described in [30] for de-Bruijn and Kautz topologies to implement a distributed RA that features lower complexity than the routing-table-based architecture. The proposed solution is applied to the flexible architecture for iterative channel code decoding proposed in [18] extending the

results presented in [34]. Even if optimized interconnection structures can be custom designed for one channel code decoder, or for a limited number of cases, this approach is not practical when high flexibility is required, e.g. when the decoder supports more codes and communications standards [18, 35, 36]. Thus, the contributions of this work are i) to show that the optimal shortest-path RA described in [30] can be implemented as a distributed RA, obtaining a very low complexity circuit, ii) to employ this circuit in the design of flexible iterative channel code decoder architectures, achieving better results than the ones shown in state of the art [18].

The rest of the paper is organized as follows. Section 2 summarizes related work and Section 3 presents generalized de-Bruijn and Kautz topologies, highlighting the main characteristics that can be exploited in NoC design. In Section 4 the optimal shortest-path RA proposed in [30] is summarized and a very simple circuit for implementing it is proposed. This circuit is finally exploited in Section 5 to reduce the complexity of the intra-IP-NoC based architecture proposed in [18] for Turbo and LDPC code decoding. Finally, in Section 6 conclusions are drawn.

2. Related work

In [23] the authors show that de-Bruijn and Kautz topologies achieve higher performance and consume less power than meshes in general NoC-based systems. A similar result is shown in [24] where the routing is based on the optimal shortest-path RA described in [30]. However, [24] targets virtual channels with a general packet structure and stores in the header flit tag bits. The meaning of these tag bits will be clarified in Section 4, where we will show that tag bits can be removed from the packet. The work in [25] proposes two RA based on shift direction. The most complex one, referred to as shortest shifting based RA, relies on shortest-path routing. However, since [25] does not target a specific application, the packet structure is more complex than the one used in this current work. In [27] a simple RA for de-Bruijn topologies, based on shift and compare operations as in [25], is proposed. However, it is applied only to binary de-Bruijn topologies and no area results are given. In [26] a Kautz NoC is used to design a very high speed neocortical computing processor for visual recognition. The properties of the Kautz topology, in particular its logarithmic diameter, are exploited to enable fast communication among the processing cores and to perform distributed low-radix routing.

On the other hand, the works presented in [18, 19, 20] propose de-Bruijn and Kautz topologies for iterative channel code decoder architectures. In particular, results in [19, 20] confirm that for iterative channel code decoder architectures, the throughput achieved by de-Bruijn and Kautz topologies is from 10% to 50% higher than the one obtained with ring, honeycomb and mesh topologies with nearly the same area. These results are exploited in [18] to design a NoC based decoder architecture that supports both turbo and LDPC codes from several different standards. All the works in [18, 19, 20] exploit shortest-path routing to minimize the latency and maximize the throughput, however they to not use

the optimal RA described in [30]. Moreover, [18, 19] rely on routing tables, whereas [20] is limited to binary de-Bruijn topologies and it does not provide details on the circuit to implement the shortest-path RA.

Finally, we expect that other applications could benefit from the use of de-Bruijn and Kautz topologies. As an example NoC-based architectures for video processing have been considered in [37, 38]. Moreover, decoders for the H.264 standard for video compression are included in the MSCL NoC benchmark suite [39]. In particular, H.264 decoders have a number of tasks and links that is comparable with the turbo/LDPC decoder case of study. It is worth pointing out that, as argued in [24], de-Bruijn and Kautz topologies are scalable, meaning that the general architecture to implement the node (and the RA) remains the same independently of the number of nodes.

3. De-Bruijn and Kautz topologies

In this section De-Bruijn and Kautz topologies are presented. The notation used to describe the topologies and to introduce their characteristics is summarized in Table 1.

Table 1: Notation

$\mathcal{A}$	Alphabet of symbols
l	Number of symbols in $\mathcal{A}$
v, w	Source and destination nodes
$\mathbf{v}$, $\mathbf{w}$	Representation of v and w as a sequence of digits taken from $\mathcal{A}$
q	Length of $\mathbf{v}$ and $\mathbf{w}$
v_i, w_i	i-th digit of $\mathbf{v}$ and $\mathbf{w}$
D	Degree of the topology
P	Number of nodes
s	Number of self-loops

De-Bruijn and Kautz topologies are obtained by building directed graphs according to the following definitions.

Definition 1. *A de-Bruijn sequence is an array of q elements, where each element is taken from an alphabet $\mathcal{A}$ with l symbols.*

Thus, a de-Bruijn graph is made of nodes labeled with de-Bruijn sequences. Let $\mathbf{v} = v_{q-1}, \ldots, v_0$ and $\mathbf{w} = w_{q-1}, \ldots, w_0$ be the labels (expressed as de-Bruijn sequences) of two nodes v and w in a de-Bruijn graph, where v and w are numbers in the base l (the number of symbols in the alphabet $\mathcal{A}$) and $v_i, w_i \in \mathcal{A}$ with $0 \leq i \leq q-1$ are the elements of the digit array symbols in the alphabet $\mathcal{A}$. There is an arc from node v to node w if $w_i = v_{i-1}$ for $1 \leq i \leq q-1$, that is $\mathbf{w}$ is obtained by left-shifting $\mathbf{v}$ and by placing in the rightmost position a symbol from $\mathcal{A}$. As a consequence, each node is connected to l nodes. Thus, the graph is regular with degree $D = l$ and the number of nodes is $P = l^q$ (Fig. 1 (a)). Unfortunately, de-Bruijn graphs have self-loops (one node connected to itself). Self-loops can be avoided using Kautz graphs.

Definition 2. *A Kautz sequence is an array of q elements, where each element is taken from an alphabet $\mathcal{A}$ with l symbols avoiding sequences with equal symbols in consecutive positions.*

From this definition we infer that a Kautz graph is made of nodes labeled with Kautz sequences and there is an arc from node v to node w if $w_i = v_{i-1}$ for $1 \leq i \leq q-1$ (with $w_1 \neq w_0$). Equivalently, there is an arc from node v to node w if $\mathbf{w}$ is obtained by left-shifting $\mathbf{v}$ and by placing in the rightmost position a symbol from $\mathcal{A}$, subject to the constraint that the result is a Kautz sequence. As a consequence, each node is connected to $l-1$ nodes and the graph is regular with degree $D = l - 1$; the number of nodes is $P = l \cdot (l-1)^{q-1}$ (Fig. 1 (b)).

As it can be inferred from above definitions, de-Bruijn and Kautz graphs for given P and D not always exist. This limitation is overcome by generalized de-Bruijn and generalized Kautz graphs, proposed in [40, 41, 42].

Definition 3. *A generalized de-Bruijn graph has an arc from node v to node w if (1) holds true:*

$$w = (D \cdot v + r) \bmod P, \tag{1}$$

with $0 \leq v \leq P-1$ and $0 \leq r \leq D-1$ (Fig. 1 (c)).

Definition 4. *A generalized Kautz graph has an arc from node v to node w if (2) holds true:*

$$w = -(D \cdot v + r) \bmod P, \tag{2}$$

with $0 \leq v \leq P-1$ and $1 \leq r \leq D$, or equivalently

$$w = [D \cdot (P - 1 - v) + r] \bmod P, \tag{3}$$

with $0 \leq v \leq P-1$ and $0 \leq r \leq D-1$ (Fig. 1 (d)).

Unfortunately, both generalized de-Bruijn and generalized Kautz graphs can have self-loops. However, as shown in [29], the numbers of self-loops s in generalized de-Bruijn and generalized Kautz graphs are $D \leq s \leq 2 \cdot D - 2$ and $0 \leq s \leq D$ respectively. Besides, in generalized Kautz graphs, $s = 0$ is achieved when

$$P \bmod (D+1) = 0, \tag{4}$$

that is $D+1$ is a divider of P [29]. Moreover, as detailed in [41, 42], generalized de-Bruijn and generalized Kautz graphs have logarithmic diameter. This property is particularly interesting in the NoC context as it ensures that the length of shortest paths connecting any two nodes (the number of hops) grows as the logarithm of P. In particular, the diameter of generalized de-Bruijn and generalized Kautz graphs are $\lceil \log_D(P) \rceil$ and $\lceil \log_D(P \cdot (D-1) + D) \rceil - 1$, where the latter one is the lower bound for directed Moore graphs [43] and $\lceil x \rceil$ is the minimum integer not smaller than x. As a consequence, generalized Kautz graphs not only have less self-loops than generalized de-Bruijn ones, but they are optimal from the diameter size point of view i.e. they minimize the number of hops connecting two nodes on a shortest path. According to [29] the number of self-loops in generalized Kautz topologies is $s = b \cdot \lfloor D/b \rfloor$, where $b = \gcd(P, D+1)$ and $\lfloor x \rfloor$ is the maximum integer not larger than x.

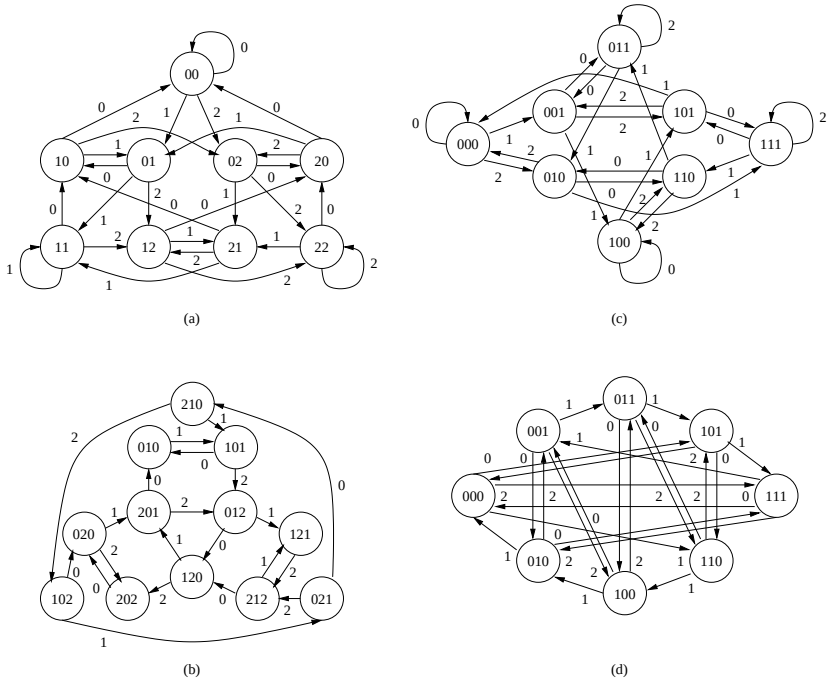


Figure 1: Example of logarithmic diameter graphs: (a) $q = 2, l = 3$ ($D = 3, P = 9$) de-Bruijn graph, (b) $q = 3, l = 3$ ($D = 2, P = 12$) Kautz graph, (c) $D = 3, P = 8$ generalized de-Bruijn graph, (d) $D = 3, P = 8$ generalized Kautz graph.

4. Optimal shortest path routing: algorithm and implementation

In this section the optimal shortest path RA, proposed in [30], is briefly summarized. Then, a simple circuit to implement it as a distributed RA is proposed.

4.1. Optimal shortest path routing algorithm for generalized Kautz topologies

One of the most interesting results shown in [29] is that generalized de-Bruijn and generalized Kautz graphs have self-routing property. Moreover, there exist a path of length $m = \lceil \log_D(P) \rceil$ that connects any pair of nodes in the network. Besides, in [30] these properties are exploited to propose a shortest path RA to connect any pair of nodes. For each pair of nodes v and w the RA computes a tag t that is exploited to derive the shortest path. Namely, the tag is converted into an array of D-ary elements, whose length $z \leq m$ is the length of the path and the routing path is derived as follows:

$$y_{n-1} = [D \cdot (P - 1 - y_n) + t_{n-1}] \bmod P, \tag{5}$$

where $n = 1, \ldots, z$ and $y_z = v$ and $y_0 = w$.

Algorithm 1 Optimal shortest-path RA for generalized Kautz topologies [30].

1: **if** $v = w$ **then**
2: $z \leftarrow 0$
3: **else**
4: $z \leftarrow 1$
5: $found \leftarrow$ FALSE
6: **while** NOT $found$ **do**
7: **if** z is odd **then**
8: $g \leftarrow [w + (v + 1) \cdot D^z] \bmod P$
9: **else**
10: $g \leftarrow [w - v \cdot D^z] \bmod P$
11: **end if**
12: **if** $g < D^z$ **then**
13: $found \leftarrow$ TRUE
14: **for** $n = 1$ to z **do**
15: **if** n is odd **then**
16: $t_n \leftarrow D - 1 - g_n$
17: **else**
18: $t_n \leftarrow g_n$
19: **end if**
20: **end for**
21: **else**
22: $z = z + 1$
23: **end if**
24: **end while**
25: **end if**

The steps to compute t are shown in Algorithm 1, where v and w are the source and the destination node respectively. This routing algorithm can be implemented both in lumped form - namely the source node computes all the path from u to v - and in distributed form. In the second case, each node computes only one step of the routing, namely the step required to go one hop ahead to reach the destination. These concepts are detailed in the following paragraphs.

4.2. Digital circuit for distributed optimal shortest path routing

The shortest path from node v to node w can be computed exploiting the recurrence in (5). As it can be observed, this is a first order recurrence so there is no need to keep trace of the previous steps. Thus, when a packet arrives at node y_n, only w and y_n are required to compute y_{n-1}. That is the only information required to compute y_{n-1} is i) the current node y_n and the destination node w. This property permits to reuse Algorithm 1 simply replacing v with y_n. Moreover, the *for* loop at line 14 in Algorithm 1 reduces to the computation of

t_{z-1} only. Another simplification is obtained by rewriting (5) as

$$y_{n-1} = (\chi_n + t_{n-1}) \bmod P, \tag{6}$$

where $\chi_n = [D \cdot (P - 1 - y_n)] \bmod P$ is only a function of y_n. As a consequence, for each node we can precalculate χ_n and store it in a register. Then, once t_{n-1} is computed, y_{n-1} is calculated via a modulo P adder (see the right part of the dark gray shaded box in Fig. 2). Besides, t_{n-1} depends on w and y_n (Algorithm 1) so it can be implemented with some constants and few logic as detailed in the following paragraphs (see the light gray shaded box in Fig. 2).

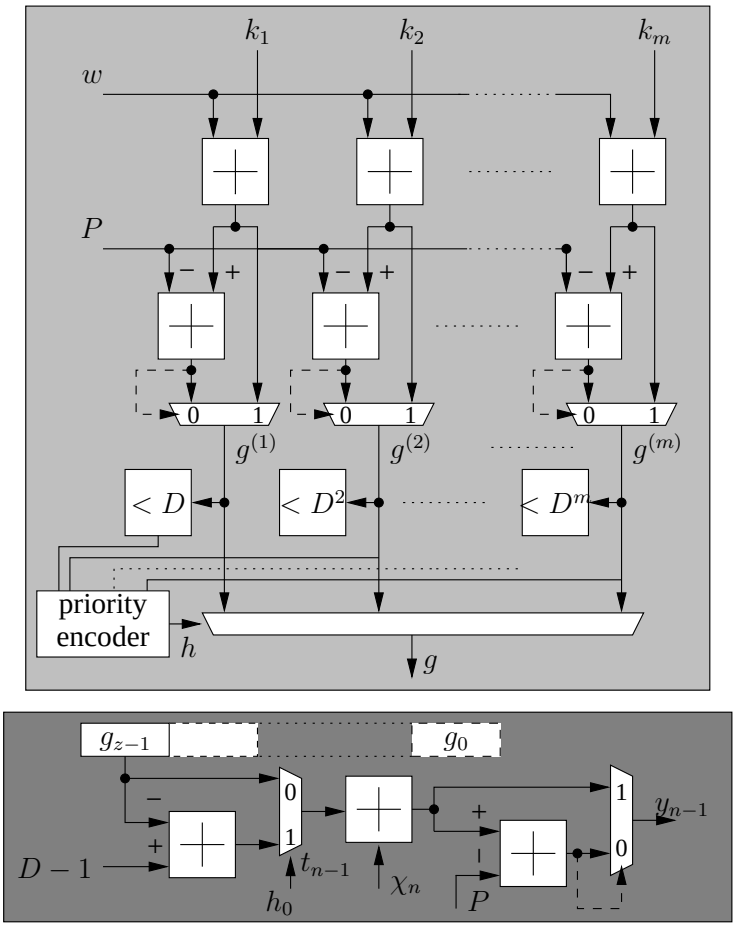


Figure 2: General architecture to implement the RA described in Algorithm 1.

The circuit that implements the shortest path routing relies on two steps: i) computing g, ii) representing g as D-ary array, whose elements are g_n, and implementing (6). The first step is implemented observing that the *while* loop at line 6 in Algorithm 1 is devoted to find both z, i.e. the length of the shortest

path from node v to node w, and g, required to compute the tag t. The same holds true if we substitute v with y_n. Since $z \leq m = \lceil \log_D(P) \rceil$, the search of z can be performed in parallel. Thus, the proposed circuit identifies m candidates for g, where the i-th candidate ($g^{(i)}$), according to lines 8 and 10 of Algorithm 1, is

$$g^{(i)} = \begin{cases} \left[w + (y_n + 1) \cdot D^i\right] \bmod P & \text{if i is odd} \\ \left[w - y_n \cdot D^i\right] \bmod P & \text{otherwise} \end{cases}, \tag{7}$$

where v has been replaced with y_n. Each of the candidates is compared to a threshold (D^i) and a priority encoder selects $g = g^{(z)} = \min_i\{g^{(i)} < D^i\}$. As it can be inferred from (7), we can compute $g^{(i)}$ via a modulo P adder where the first operand is w and the second one is

$$k_i = \begin{cases} (y_n + 1) \cdot D^i] \bmod P & \text{if i is odd} \\ P - [(y_n \cdot D^i) \bmod P] & \text{otherwise} \end{cases}, \tag{8}$$

which can be precalculated and stored in a register inside node y_n. As shown in the light gray shaded box of Fig. 2, each mod P adder relies on a subtracter and a multiplexer driven by the sign of $w + k_i - P$: if the sign is positive then $g^{(i)} = w + k_i - P$, otherwise $g^{(i)} = w + k_i$.

The second step, depicted in the dark gray shaded box in Fig. 2, relies on representing g as an array of at most m D-ary elements, where the element at position $h = z - 1$, that is g_{z-1}, is used to compute t_{z-1} as in lines 16 and 18 of Algorithm 1. Then, h_0, which is the least significant bit of h, is used to select

$$t_{z-1} = \begin{cases} D - 1 - g_{z-1} & \text{if h is odd} \\ g_{z-1} & \text{otherwise} \end{cases}. \tag{9}$$

Finally, with a modulo P operation we obtain

$$y_{n-1} = \begin{cases} (\chi_n + t_{n-1}) - P & \text{if } (\chi_\mathrm{n} + \mathrm{t_{n-1}}) - \mathrm{P} \geq 0 \\ \chi_n + t_{n-1} & \text{otherwise} \end{cases}. \tag{10}$$

As it can be inferred from Algorithm 1 and Fig. 2, significant complexity reduction can be achieved if both D and P are powers of two. Indeed, in this case both the generation of D-ary elements and modulo P operations are implemented with no hardware cost exploiting binary representation. An example of the architecture achieved for $D = 4$ and $P = 64$ is shown in Fig. 3 where i) both comparators and priority encoder are implemented with few logic gates and ii) modulo P operations are obtained by letting the data wrap (when an overflow occurs). However, being both D and P powers of two, then (4) cannot be satisfied and some self-loops have to be tolerated. The impact of this choice will be discussed in section 5, where experimental results are shown.

5. Case of study: intra-IP NoC for a turbo/LDPC decoder architecture

Turbo and LDPC codes are characterized by almost uniform traffic patterns both in time and space [44]. As a consequence, regular logarithmic diameter

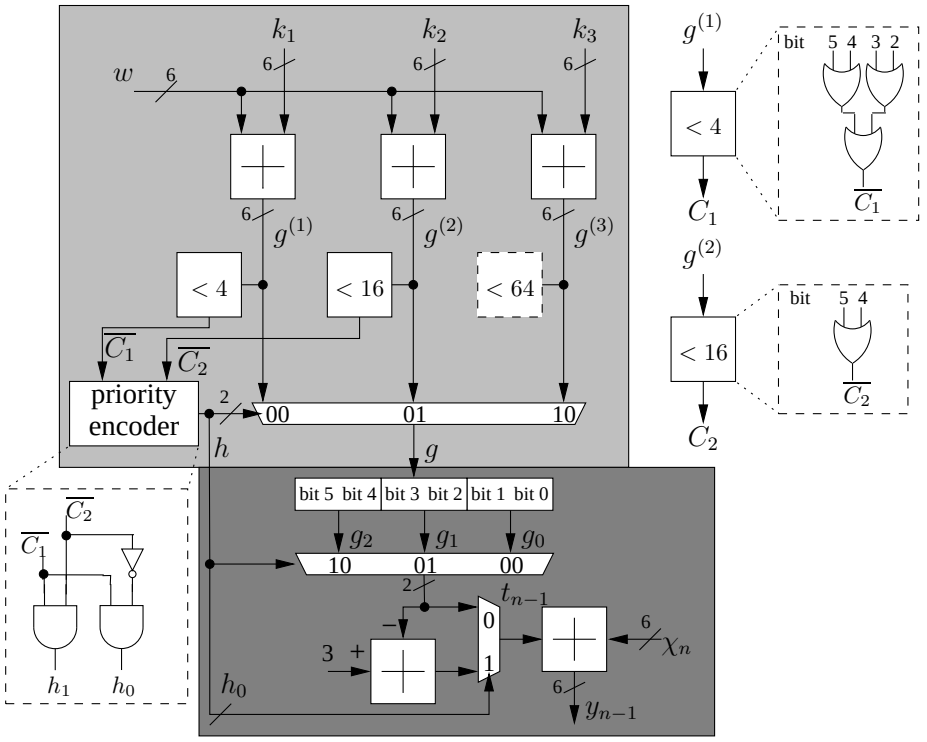


Figure 3: Architecture of the RA in Algorithm 1 when $D = 4$ and $P = 64$.

topologies, such as generalized Kautz topologies, that are able to minimize the distance between two nodes are an interesting option [19]. Besides, NoC based turbo and LDPC code decoder architectures are characterized by a deterministic traffic, imposed by the structure of the interleaver and parity check matrix respectively. The interleaving operation in turbo codes is scrambling the processing order of data within a block and relies on a permutation with almost uniform distribution [44]. The parity check matrix of LDPC codes is a very sparse binary matrix and the position of the '1' defines the mapping pattern among the PEs. These characteristics can be exploited precalculating all the routing and scheduling information by the means of a cycle accurate simulator, such as [45], which performs RTL simulations exploiting the SystemC simulation kernel. The simulator, that has been modified to support both turbo and LDPC codes, requires the following information: i) the topology description in the form of an adjacence matrix, ii) the traffic pattern (permutation law for turbo codes or parity check matrix for LDPC codes) iii) the routing algorithm and iv) PE timing description. The simulator outputs both cycle by cycle results and global results, including the configuration and status information of each RE and the total number of clock cycles to deliver all the messages. Cycle by cycle information can be converted in a sequence of commands for each

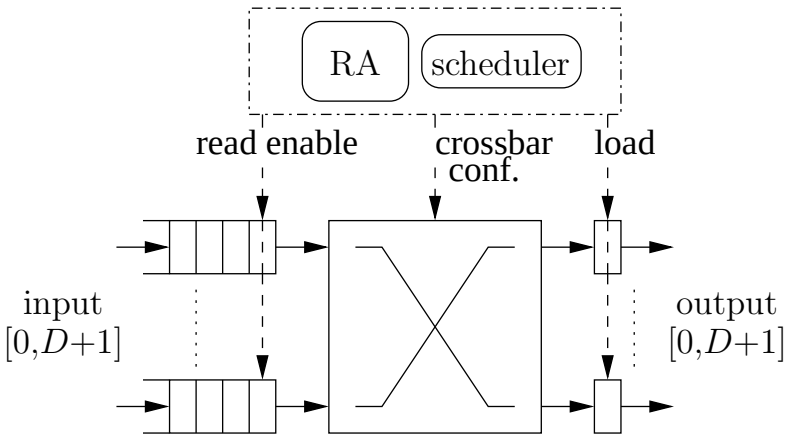


Figure 4: Architecture of a node for intra-IP NoCs.

routing element and stored in memories. Unfortunately, as shown in [46], the amount of memory required to implement such an approach is very large, being almost impractical in some real cases, such as for the HSDPA standard. As a consequence, routing and scheduling information have to be computed algorithmically on-the-fly, minimizing complexity and delay. Even if routing-tables can be used for this class of applications, the proposed low complexity circuit is a scalable alternative.

The node architecture proposed in [18] relies on a PE, that performs the data computation, and a Routing Element (RE), that sends and receives data to/from the network. Each RE is made of a $(D+1) \times (D+1)$ crossbar, $D+1$ input FIFOs and output registers, that are managed by an RA, and a scheduler as depicted in Fig. 4. As discussed in [19] shortest-path RAs coupled with Round-Robin (RR) and Longest-Queue-First (LQF) scheduling policies are the most suited solutions for channel code decoding applications.

In this work the intra-IP NoC-based multi-mode turbo/LDPC decoder architecture proposed in [18] is taken as a case of study. In particular, this architecture i) supports all WiMAX LDPC codes, that in the worst case imply a frame size $N = 2304$ and a code rate $r = 0.5$, and ii) relies on a $D = 3$, $P = 22$ nodes generalized Kautz topology where the shortest-path RA is distributed and performed via routing tables. It is worth noting that the frame size N corresponds to the number of packets sent over the network during an iteration. In the following, the architecture presented in [18] is extended to the case $D = 4$, $P = 32$ and it will be referred to as Routing-Table-based NoC (RT-NoC) architecture. The RT-NoC architecture is then modified replacing the routing tables with the circuit-based RA described in Section 4.2. This new architecture will be referred to as RA-NoC. Both architectures have been described in VHDL, simulated with Modelsim and implemented on a CMOS 90 nm standard cell technology for a 200 MHz target clock frequency [47] with Synopsys Design Compiler and Cadence Encounter. As an example the post

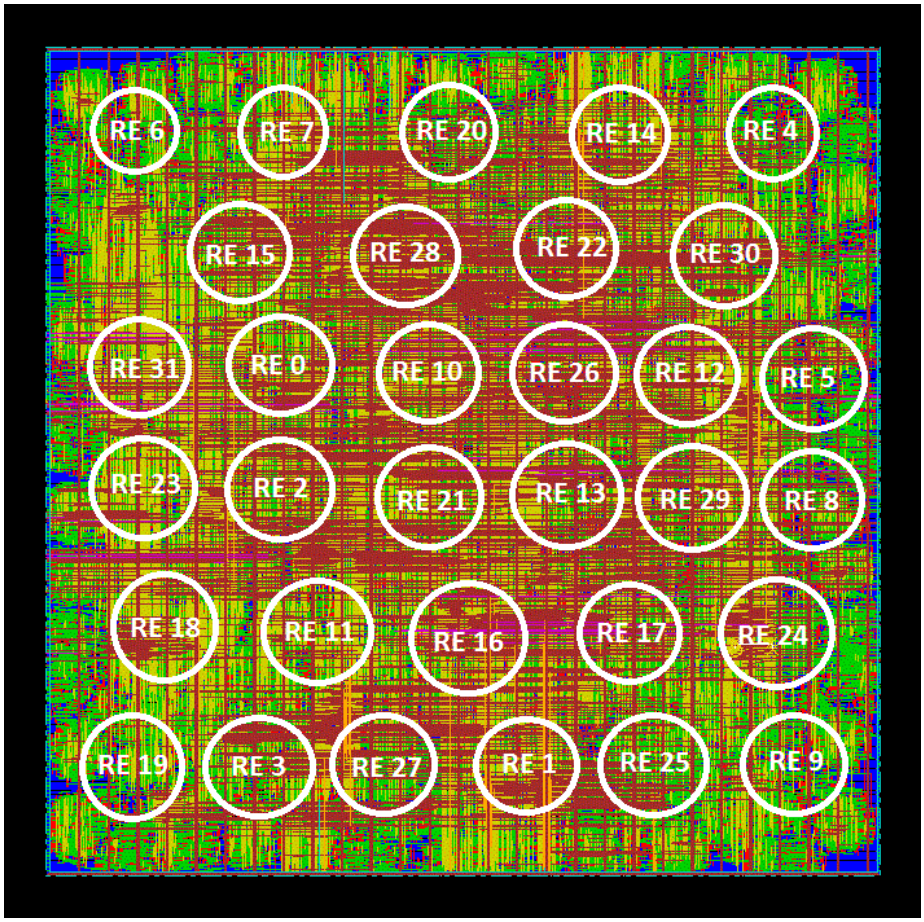


Figure 5: Post place and route layout of the proposed intra-IP RA-NoC for $D = 4$, $P = 32$.

place and route layout of the proposed intra-IP RA-NoC for $D = 4$, $P = 32$ is shown in Fig. 5. Stemming from the design in [18], the FIFOs in each node have been conservatively sized to eight locations to prevent slowing the PEs. The structure of the packet is tailored around the application and relies on a header, containing the destination node ($\lceil \log_2 P \rceil = 5$ bits), and a payload, containing one datum (6 bits) and the memory location where the datum will be stored at destination node ($\lceil \log_2(N \cdot r/P) = 6$ bits). Thus, the data width of the NoC is 17 bits and, since each packet is made of one flit only, then the system is deadlock-free.

Post place and route area and switching-activity-based power consumption are summarized for both architectures in Table 2. As it can be observed the proposed RA-NoC features an area and a power consumption reduction of about 14% and 10% with respect to the RT-NoC.

Table 2: Area (A) and power (Pow) consumption comparison between RT-NoC and RA-NoC. CMOS 90 nm standard cell technology, clock frequency 200 MHz, 8-element FIFOs, $P = 32$, $D = 4$.

	A		Pow	
	[mm^2]	% reduction	[mW]	% reduction
RT-NoC	0.807	-	75.1	-
RA-NoC	0.691	-14.4%	67.2	-10.5%

Table 3: Area (A), Power consumption (Pow) and clock cycles (cycles) comparison between two RA-NoCs with $P = 30$ and $P = 32$. CMOS 90 nm standard cell technology, clock frequency 200 MHz, 8-element FIFOs, $D = 4$, WiMAX LDPC code $N = 2304$, $r = 0.5$.

	nodes with 3 connection	A [mm^2]	Pow [mW]	cycles
$P = 30$	0	0.650	58.2	5028
$P = 32$	4	0.691	67.2	4880

As discussed in section 4.2, choosing D and P as powers of two leads to a simplification in the RA circuit, but introduces self-loops. Since self-loops are not used, they have been removed in the implemented architecture. This simplification can reduce the decoder throughput as some nodes have less connections, namely with $D = 4$ and $P = 32$ there are twenty-eight nodes with 4 connections and four nodes with 3 connections. To explore this direction we observed that, if D is not a power of two, then converting g to an array of D-ary elements becomes complex. Moreover, if P is not a power of two, then modulo operations can be performed with a limited complexity, indeed they require a subtracter and a multiplexer (see Fig. 2). Thus, we fixed $D = 4$ and $P = 30$, so that (4) is satisfied and self-loops are avoided, leading to a topology where all nodes have 4 connections, and implemented the corresponding NoC in the same conditions used for the case $D = 4$, $P = 32$. As it can be inferred from Table 3 a fair comparison is not straightforward. Indeed, experimental results in Table 3, highlight that the $D = 4$, $P = 30$ architecture is slightly smaller and consumes less power than the $D = 4$, $P = 32$ one. On the contrary, the number of clock cycles needed to complete the message exchange phase for the WiMAX LDPC code with frame size $N = 2304$ and code rate $r = 0.5$ is larger for the case $D = 4$, $P = 30$ than for the case $D = 4$, $P = 32$. The area clock-cycles product (ACP) is a possible figure of merit to have a fair comparison between the two solutions. The ratio between the two ACP values is $\mathrm{ACP}_{D=4,P=30}/\mathrm{ACP}_{D=4,P=32} = 0.97$, meaning that the solution $D = 4$, $P = 30$ performs better than the one with $D = 4$, $P = 32$, even if the difference is very small.

In [19] it is shown that logarithmic diameter topologies achieve higher throughput than the well known mesh topologies. However, this result was obtained using the same clock frequency for both topologies. It is largely recognized that mesh topologies, being highly regular can benefit from short and uniform interconnect delays and so they can reach higher clock frequencies than other topologies, such as Kautz ones. As a consequence, even if mesh topologies have

Table 4: Area occupation (A), Power consumption (Pow), maximum clock frequency (f) and clock cycles (cycles) comparison between Kautz RA-NoC and toroidal mesh (t-mesh) RT-NoC. CMOS 90 nm standard cell technology, 8-element FIFOs, $D = 4$, $P = 32$, WiMAX LDPC code $N = 2304$, $r = 0.5$.

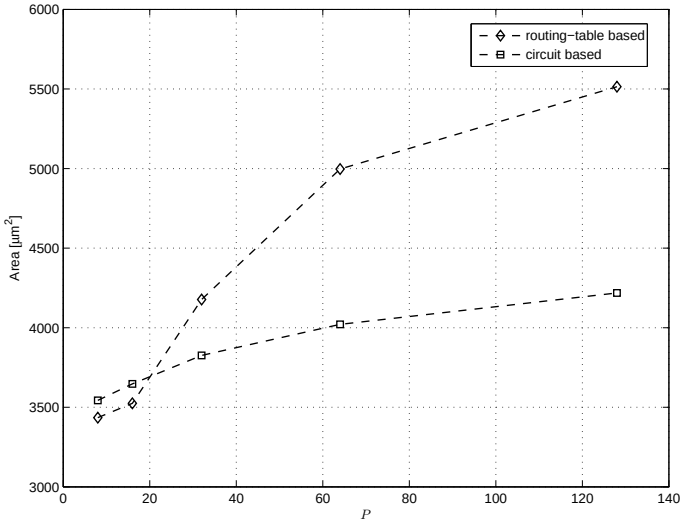
	A [mm^2]	Pow [mW]	f [MHz]	cycles
kautz	0.712	162.39	500	4880
t-mesh	0.717	174.24	555	6131

larger diameter than Kautz topologies, they can be advantageous in terms of throughput because they can run faster. Thus, we implemented a toroidal mesh topology with $D = 4$ and $P = 32$ and we derived the maximum achievable clock frequency for the generalized Kautz topology and the toroidal mesh topology respectively. Experimental results, shown in Table 4, confirm that the toroidal mesh topology can run at a higher clock frequency than the generalized Kautz one. On the contrary, the generalized Kautz topology requires less clock cycles than the toroidal mesh topology for the decoding of the WiMAX LDPC code with frame size $N = 2304$ and code rate $r = 0.5$. As it can be inferred from Table 4, the time to complete the message exchange phase for WiMAX LDPC code $N = 2304$, $r = 0.5$ with the generalized Kautz topology and the toroidal mesh topology are 9.7 μs and 11μs respectively. This last result confirms the effectiveness of logarithmic topologies, such as the generalized Kautz ones, for low latency applications, as iterative channel code decoder architectures.

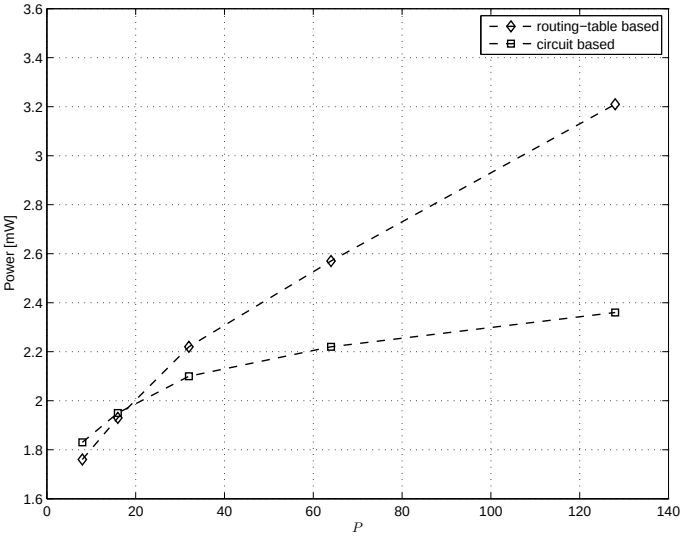
Finally, in Fig. 6 the area and the power consumption of one routing-table-based RE is compared with the area and the power consumption of one circuit-based RE as a function of P in the same test conditions employed for the full decoders. As it can be observed, as long as P increases the advantage of the circuit-based RE becomes larger, e.g. when $P = 64$ the area and power reductions are about 20% and 14% with respect to one table-based RE. It is worth noting that, with a target clock frequency of 200 MHz the propagation delay of the routing decision calculation in one RE is 3.3 ns. However, the circuit can be pushed to run up to 950 MHz, accepting to increase the area by 2.5 times.

6. Conclusion

In this work a circuit to implement the optimal shortest-path RA proposed in [30] for generalized Kautz topologies is presented. The proposed circuit features lower complexity and power consumption than routing-table-based shortest-path RAs. Experimental results on the NoC-based design proposed in [18] show that the complexity of the NoC is reduced by about 14% and the power consumption by about 10%. Compared with the well known toroidal mesh topology the proposed solution achieves higher throughput, being an interesting approach for high throughput and low latency applications.



(a) Area



(b) Power

Figure 6: Area and power consumption of one routing-table-based RE and one circuit-based RE as a function of

Acknowledgment

This work has been partially funded by the Newcom# project.

References

[1] 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.

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

[3] 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.

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

[5] S. V. Tota, M. R. Casu, M. R. Roch, L. Rostagno, M. Zamboni, MEDEA: A hybrid shared-memory/message-passing multiprocessor NoC-based architecture, in: Design, Automation and Test in Europe Conference and Exhibition, 2010, pp. 45–50.

[6] W. Xiohang, M. Palesi, Y. Mei, J. Yingtao, M. C. Huang, L. Peng, Low latency and energy efficient multicasting schemes for 3D NoC-based SoCs, in: IEEE International Conference on VLSI and System-on-Chip, 2011, pp. 337–342.

[7] M. Daneshtalab, M. Ebrahimi, J. Plosila, H. Tenhunen, CARS: Congestion-aware request scheduler for network interfaces in NoC-based manycore systems, in: Design, Automation & Test in Europe Conference & Exhibition, 2013, pp. 1048–1051.

[8] D. Zhao, Y. Wang, SD-MAC: design and synthesis of a hardware-efficient collision-free QoS-aware MAC protocol for wireless Network-on-Chip, IEEE Transactions on Computers 57 (9) (2008) 1230–1245.

[9] M. Crepaldi, D. Demarchi, A 130-nm CMOS 0.007 mm2 ring-oscillator-based self-calibrating IR-UWB transmitter using an asynchronous logic duty-cycled PLL, IEEE Transactions on Circuits and Systems II 60 (5) (2013) 237–241.

[10] F. Clermidy, N. Cassiau, N. Coste, D. Dutoit, M. Fantini, D. Ktenas, R. Lemaire, L. Stefanizzi, Reconfiguration of a 3GPP-LTE telecommunication application on a 22-core NoC-based system-on-chip, in: IEEE/ACM International Symposium on Networks on Chip, 2011, pp. 261–262.

[11] K. Chen, S. Lin, H. Hung, A. Wu, Topology-aware adaptive routing for nonstationary irregular mesh in throttled 3D NoC systems, IEEE Transactions on Parallel and Distributed Systems 24 (10) (2013) 2109–2120.

[12] K. S. M. Li, CusNoC: Fast full-chip custom NoC generation, IEEE Transactions on VLSI systems 21 (4) (2013) 692–705.

[13] S. Murali, D. Atienza, P. Meloni, S. Carta, L. Benini, G. D. Micheli, L. Raffo, Synthesis of predictable networks-on-chip-based interconnect architectures for chip multiprocessors, IEEE Transactions on VLSI systems 15 (8) (2007) 869–880.

[14] S. Rodrigo, J. Flich, A. Roca, S. Medardoni, D. Bertozzi, J. Camacho, F. Silla, J. Duato, Cost-efficient on-chip routing implementations for CMP and MPSoC systems, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 30 (4) (2011) 534–547.

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

[16] 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.

[17] F. Clermidy, C. Bernard, R. Lemaire, J. Martin, I. Miro-Panades, Y. Thonnart, P. Vivet, N. Wehn, A 477mw NoC-based digital baseband for MIMO 4G SDR, in: IEEE International Solid-State Circuits Conference, 2010, pp. 278–279.

[18] C. Condo, M. Martina, G. Masera, VLSI implementation of a multi-mode turbo/LDPC decoder architecture, IEEE Transactions on Circuits and Systems - I 60 (6) (2013) 1441–1454.

[19] M. Martina, G. Masera, Turbo NOC: A framework for the design of network-on-chip-based turbo decoder architectures, IEEE Transactions on Circuits and Systems - I 57 (10) (2010) 2776–2789.

[20] H. Moussa, A. Baghdadi, M. Jezequel, Binary de Bruijn on-chip network for a flexible multiprocessor LDPC decoder, in: Design Automation Conference, 2008, pp. 429–434.

[21] N. G. de Bruijn, A combinatorial problem, Koninklijke Nederlandse Akademie v. Wetenschappen 49 (1946) 758–764.

[22] W. H. Kautz, Design of optimal interconnection networks for multiprocessors, Architecture and Design of Digital Computers (1969) 249–272.

[23] R. Sabbaghi-Nadooshan, M. Modarresi, H. Sarbazi-Azad, The 2D digraph-based NoCs: attractive alternatives to the 2D mesh NoCs, Journal of Supercomputing 59 (1) (2012) 1–21.

[24] M. Hosseinabady, M. R. Kakoee, J. Mathew, D. K. Pradhan, Low latency and energy efficient scalable architecture for massive NoCs using generalized de Bruijn graph, IEEE Transactions on VLSI systems 19 (8) (2011) 1469–1480.

[25] Y. Chen, J. Hu, X. Ling, T. Huang, A novel 3D NoC architecture based on De Bruijn graph, Elsevier Computers and Electrical Engineering 38 (3) (2012) 801–810.

[26] C. Y. Tsai, Y. J. Lee, C. T. Chen, L. G. Chen, A 1.0TOPS/W 36-core neocortical computing processor with 2.3tb/s Kautz NoC for universal visual recognition, in: IEEE International Solid-State Circuits Conference, 2012, pp. 480–482.

[27] F. Stas, A. K. Lusala, J. D. Legat, D. Bol, Investigation of the routing algorithm in a De Bruijn-based NoC for low-power applications, in: IEEE Faible Tension Faible Consommation, 2013, pp. 1–4.

[28] M. R. Samatham, D. K. Pradhan, The De Bruijn multiprocessor network: A versatile parallel processing and sorting network for VLSI, IEEE Transactions on Computers 38 (4) (1989) 567–581.

[29] D. Z. Du, F. K. Hwang, Generalized de Bruijn digraphs, Networks 18 (1988) 27–38.

[30] G. Liu, K. Y. Lee, Optimal routing algorithms for generalized de Bruijn digraphs, in: International Conference on Parallel Processing, 1993, pp. 167–174.

[31] J. Flich, J. Duato, Logic-based distributed routing for NoCs, IEEE Computer Architecture Letters 7 (1) (2008) 13–16.

[32] S. Rodrigo, S. Medardoni, J. Flich, D. Bertozzi, J. Duato, Efficient implementation of distributed routing algorithms for NoCs, IET Computers and Digital Techniques 3 (5) (2009) 460–475.

[33] N. Choudhary, C. M. Samota, A survey of logic based distributed routing for on-chip interconnection networks, International Journal of Soft Computing and Engineering 3 (2) (2013) 233–237.

[34] C. Condo, M. Martina, M. R. Roch, G. Masera, Rediscovering logarithmic diameter topologies for low latency Network-on-Chip-based applications, in: Euromicro International Conference on Parallel, Distributed and Network-Based Processing, 2014, pp. 418–423.

[35] T. Vogt, N. Wehn, Reconfigurable ASIP for convolutional and turbo decoding in an SDR environment, IEEE Transactions on VLSI 16 (10) (2008) 1309–1320.

[36] O. Muller, A. Baghdadi, M. Jezequel, From parallelism levels to a multi-ASIP architecture for turbo decoding, IEEE Transactions on VLSI 17 (1) (2009) 92–102.

[37] A. Jalabert, S. Murali, L. Benini, G. D. Micheli, xpipesCompiler: a tool for instantiating application specific Networks on Chip, in: Design, Automation and Test in Europe Conference and Exhibition, 2004, pp. 884–889.

[38] S. Saponara, M. Martina, M. Casula, L. Fanucci, G. Masera, Motion estimation and CABAC VLSI co-processors for real-time high-quality H.264/AVC video coding, Elsevier Microprocessors and Microsystems 34 (7-8) (2010) 316–328.

[39] W. Liu, J. Xu, X. Wu, Y. Ye, X. Wang, W. Zhang, M. Nikdast, Z. Wang, A noc traffic suite based on real applications, in: IEEE Computer Society Annual Symposium on VLSI, 2011, pp. 66–71.

[40] S. M. Reddy, D. K. Pradhan, J. G. Kuhl, Direct graphs with minimal and maximal connectivity, Tech. rep., School of Engineering, Oakland University (1980).

[41] M. Imase, M. Itoh, Design to minimize diameter on building-block network, IEEE Transactions on Computers 30 (6) (1981) 439–442.

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

[43] W. G. Bridges, On the impossibility of directed moore graphs, Journal of Combinatorial Theory 29 (3) (1980) 339–341.

[44] 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.

[45] M. Martina, Turbo NOC: Network On Chip based turbo decoder architectures, downloadable at http://personal.delen.polito.it/maurizio.martina/turbo.html (2012).

[46] M. Martina, G. Masera, H. Moussa, A. Baghdadi, On chip interconnects for multiprocessor turbo decoding architectures, Elsevier Microprocessors and Microsystems 35 (2) (2011) 167–181.

[47] A. Pulimeno, M. Graziano, G. Piccinini, UDSM trends comparison: From technology roadmap to UltraSparc Niagara2, IEEE Trans. on VLSI 20 (7) (10.1109/TVLSI.2011.2148183) 1341–1346.