On All-to-All Broadcast in Dense Gaussian Network On-Chip

Abderezak Touzene, Member, IEEE

Abstract—Gaussian networks are gaining popularity as good candidates Network On-Chip (NoC) for interconnecting Multiprocessor System-on-Chips (MPSoCs). They showed better topological properties compared to the 2D torus networks with the same number of nodes \( N \) and the same degree 4. All-to-all broadcast is a collective communication algorithm used frequently in many parallel applications. Recently, Z. Zhang et al. [1] have proposed an all-to-all broadcast algorithm for Gaussian on-chip networks that achieves the minimum delay time but requires 4\( k \) extra buffers per router, where \( k \) is the network diameter. In this paper, we propose a new all-to-all broadcast algorithm for dense Gaussian on-chip networks that achieves the minimum delay time without requiring any extra buffers per router. Along with low latency, reducing the amount of buffer space and power consumption are very important issues in NoC architectures.

Index Terms—Network-on-Chip (NoC), Gaussian networks, all-to-all broadcasting, spanning trees.

1 INTRODUCTION

The scale down of transistor technology allows microelectronics manufacturers to embed more sophisticated systems on a single micro-chip (SoC) [2], [3], [4], [5], [6], [7]. On chip integration is becoming popular not only in the Chip Multi Processor (CMP) research area but also in the wider range of nowadays devices such as cell-phones, smart-phones, smart-houses and vehicle embedded systems, etc.

As the number of cores increase in CMP, the system performance begins to be affected by on-chip interconnects. The widely used bus structures (shared bus, hierarchal bus, and bus matrix) are becoming a limiting factor for performance, space and energy consumption [5]. In order to overcome the disadvantages of bus structures, they are being replaced by Network-on-Chip (NoC). NoC architectures are an attempt to scale down the concepts of large-scale networks, and apply them in the embedded system-on-chip (SoC) domain [8], [9], [10], [11], [12], [13]. A NoC is an on-chip communication network used to route data from a source Processing Element (PE) to a destination PE via a network fabric that consists of switches (routers) and interconnection links (wires) [8]. NoC interconnects address scalability issues and provide a low latency communication layer, higher bandwidth, and communication parallelism.

Among the most popular NoC topologies developed for CMPs such as n-dimensional mesh, torus, folded torus, hypercube, and octagon, low-dimensional networks, like 2D-mesh and 2D-torus, offer better performance in terms of higher throughput and lower latency than higher-dimensional networks (high dimensional k-aryn-cubes and meshes) [13], [14]. Recently, two other wrap-around networks have been proposed as suitable alternatives to the 2D torus network: the degree 4 Gaussian networks [15], [16], [17] and the degree 6 Eisenstein-Jacobi (EJ) networks [17], [31], [32]. These families of 2D-toroidal networks exhibit remarkable topological improvement over a 2D torus networks with the same number of nodes \( N \) and the same degree. Recently, AlBader et al. [37] presented collective communication for off-chip degree 6 EJ networks. In the context of NoC architectures, our focus is on the degree 4 Gaussian networks because they are less complex in their design and less costly compared to degree 6 EJ networks (more links). Gaussian networks have a 2D grid-like structure with a regular topology and low degree 4. The network diameter of the Gaussian network is about \( \frac{1}{\sqrt{2}} \) the diameter of its torus counterpart. The average hop-to-hop distance in the Gaussian network is considerably reduced compared to its torus counterpart [15]. For the mentioned topological characteristics Gaussian networks are more suitable topology candidate for NoC architectures. With the development of diverse applications in CMPs, collective communications such as one-to-many (multicast), one-to-all (broadcast) and all-to-all (gossip) are becoming more common. Despite the fact that NoC architectures can be seen as a special case of parallel computing architectures, these systems are characterized by tight constraints in terms of resources such as memory storage. NoC architectures requires the development of new communication algorithms and protocols taking into consideration these constraints. It should be possible to implement routing algorithms as embedded circuits into the NoC architectures. For this reason, routing and collective communication algorithms have to be
simple, fast and their buffer requirements have to be reduced to the minimum. In [35] a study of finding the best topology that gives the best performance for different collective communication in NoC architectures. There are few works about multicast and broadcast in NoC architectures [20], [21], [22], [41], [42] all of them applied to 2D-mesh. In [41], [42] a Logic-Based Distributed multicast and broadcast algorithms are presented which support faulty nodes in 2D-mesh. These algorithms perform the one-to-all broadcast using a tree implemented at the logic level to improve both latency and power consumption. However, the all-to-all operation is not addressed. Recently, Walter et al. [43] explore the benefits of adding a low-latency, customized shared bus as an integral part of the NoC architectures. This bus is used for certain transactions such as broadcast of queries, fast delivery of control signals, and quick exchange of small data items (one-to-all operation).

With the increase of interest to Gaussian networks, the authors in [16] proposed a one-to-all algorithm for off-chip Gaussian networks which has been adapted later in [1] for on-chip Gaussian networks. Among the collective communication algorithms we focus on developing an efficient all-to-all broadcast algorithm for Gaussian networks where each node exchanges its information with all the other nodes in the network. All-to-all broadcast algorithm is also called multi-node broadcast as it consists of simultaneous broadcasting from each node in the network. We assume that the messages broadcast from all nodes are of the same length. In many applications, such as matrix multiplication, LU-factorization, Householder transformations, and basic linear algebra operations, only all-to-all broadcast communication is needed [44], [45], [46], [47], and [1].

Recently, Z. Zhang et al. [1] have proposed an all-to-all broadcast algorithm for Gaussian on-chip networks that achieve the minimum delay time but require 4k extra buffers per router, where k is the network diameter. Adding extra-buffers in the router requires more space on-chip (higher cost) and also increases the power consumption [33]. Designing reduced buffers and bufferless communications algorithms is a new important trend in NoC architectures [33], [34].

In this paper we propose a new all-to-all broadcast algorithm for dense Gaussian network NoC architectures that achieve also the minimum delay time without requiring any extra-buffer space. Reducing the amount of buffer space is a very important issue in NoC architectures. With its reduced buffer requirements our proposed all-to-all algorithm is more appropriate for NoC architectures compared to the previous one.

Among the family of Gaussian networks we focus on the Dense Gaussian Networks (DGN). They are also called optimal Gaussian network because they achieve the largest network size for a given network diameter. In practice, dense Gaussian networks are the best candidate for interconnection networks among their Gaussian network family [15].

In this paper, we assume that every router of the on-chip Gaussian networks has full-duplex channels and all-port capability, meaning that all the channels allow simultaneous communications in both directions, and any node in the network can communicate with all its neighbors simultaneously. Each router has four first-in first-out (FIFO) input (output) buffers, one for each channel. Unlike the router requirements described in [1], our all-to-all algorithm guarantees no link contention, output buffers could be omitted to save buffer space in the router design.

Our proposed all-to-all broadcast algorithm in the dense Gaussian NoC architectures uses a special family of trees that ensure contention-free links and addresses the issue of reducing buffer space in the router such that no extra memory space is required. To the best of our knowledge, this is the first study which proposes performing all-to-all broadcast operation in the dense Gaussian network NoC architectures with reduced buffer requirements.

The remainder of the paper is organized as follows: section 2 presents some background on Gaussian networks and related work; section 3 shows how to construct the special broadcast trees in the dense Gaussian NoC architectures; section 4 outlines the proposed all-to-all broadcast algorithm; section 5 presents performance evaluation results; and section 6 concludes the paper.

2 Background and Related Work

In what follows we start by giving some definitions and terminology related to Gaussian network then we discuss the related work.

2.1 Gaussian Networks

Dense Gaussian networks are based on a special family of dense degree 4 Circulant graphs containing the maximum number of nodes for a given diameter [15]. A Circulant graph consists of set of N vertices labeled by integers from n=0, N-1 and a set of edges defined as follows: the vertex n is connected to the vertices n±j, mod N, where i=1, m. Circulant graphs are vertex-symmetric and regular graphs with degree 2m, denoted $C(j, j, \ldots, j, m)$. The regularity and the vertex-symmetry of a graph simplify considerably their study. As an example, developing a broadcast algorithm that starts at node x will work without any change as if it starts from any other node y.

For the degree 4 Circulant graphs $C(j, j, \ldots, j)$, it has been shown in [18], [19], [20], [21] that there are 4d different nodes at distance d from node 0. A dense Circulant graph $C(k, k+1)$ contains a maximum number of nodes N for a given diameter k such that $N = k^2 + (k+1)^2 = 2k^2 + 2k + 1$.

Gaussian networks are based on 4 degree Circulant graphs but could be seen as two-dimensional grid-like topology which facilitate their exploitation. Gaussian networks are modeled by Gaussian graphs where each node of the network is represented by a vertex in the graph and each link by an undirected edge. Throughout this paper in terms of Gaussian graph and Gaussian network we will use interchangeably vertex and node, or edge and link. We define Gaussian graphs by using
Gaussian integers. Gaussian integers is a subset of complex numbers $\mathbb{Z}[i] = \{x + y i \mid x, y \in \mathbb{Z}\}$. Where $\mathbb{Z}[i]$ is an Euclidean domain and the norm of a Gaussian integer $a + bi$, is defined as $N(a + bi) = a^2 + b^2$. There exist an Euclidean division for Gaussian integers: for every $a, \beta \in \mathbb{Z}[i]$ with $\alpha \neq 0$, there exist $q, r \in \mathbb{Z}[i]$ such that $\beta = qa + r$ with $N(r) < N(a)$. Typically, we denote the set of remainders of the integer Gaussian division $\mathbb{Z}[i]_\alpha$ the Gaussian integer modulo $\alpha$. A Gaussian integer can be represented as 2D complex plane.

Definition 1. Let $0 \neq \alpha = a + bi \in \mathbb{Z}[i]$. The Gaussian graph generated by $\alpha$, $G_\alpha = (V, E)$ such that: $V = \mathbb{Z}[i]_\alpha$ is the set of vertices, and $E = \{(y, \beta) \in V \times V \mid \beta - y \equiv \pm1, \pm i \pmod{\alpha}\}$ is the set of edges. The Gaussian graph generated by $\alpha$ has $N(a) = a^2 + b^2$ vertices, is a regular graph of degree 4, and is vertex-symmetric. We define dense Gaussian graphs as Gaussian graphs with the maximum number of vertices for a given diameter $k$ generated by $\alpha = k + (k + 1)i \in \mathbb{Z}[i]$, $G_{\alpha = k+(k+1)i}$. In what follows we focus only on dense Gaussian network because of their remarkable topological properties such as low diameter and low average distance compared to torus networks with the same size. Dense Gaussian networks are thus good candidate for NoC interconnection networks. It has been shown in [15], [16] the isomorphism between the dense Gaussian network $G_{\alpha = k+(k+1)i}$ and the Circulant $C(k, k+1)$. Fig. 1 (a) and (b) show the isomorphism between $G_{3+4i}$ and $C(3, 4)$. Next, we define Dense Gaussian Networks [15].

Definition 2. Let $k$ be a positive integer. The Dense Gaussian Network of diameter $k$, or $Q_k$, is defined as follows:

- The square $Q_k = \{(x, y) \in \mathbb{Z} \times \mathbb{Z} \mid |x| + |y| \leq k\}$ is the set of nodes and
- Every node $(x, y) \in Q_k$ is adjacent to the nodes $(x+1, y)$, $(x, y+1)$ and $(x, y-1)$ MOD $(k, k+1)$, where the equivalence relation MOD is defined as follows:
  
  \[
  (x, y) \equiv (x_0, y_0) \pmod{(k, k+1)}, \exists u, v \in \mathbb{Z} | (x = x_0 + uk - v(k+1)) \land (y = y_0 + u(k+1) + vk).
  \]

It can be seen from Fig. 1 (a) that the node $(0, 0)$ is located in the center of the network and the remaining nodes are located at coordinates $(x, y)$ pair of integer such that $|x| + |y| \leq k$. The node $(1, 2)$ is adjacent to nodes $(0, 2)$ and $(1, 1)$ inside the mesh. The wraparound links that connects the node $(1, 2)$ to its other two adjacent nodes are determined by the modulo reduction: $(1 + 1, 2) = (2, 2) \equiv (-1, -2) \pmod{(3, 4)}$ and taking $u = 1, v = 0$. And $(1, 2+1) = (1, 3) \equiv (-2, -1) \pmod{(3, 4)}$ and taking $u = 1, v = 0$. Note that this modulo function is only necessary for determining peripheral adjacency among nodes. Actually, this two-dimensional modulo function is the modulo reduction over the complex numbers with real and imaginary integer parts or Gaussian integers. From Fig. 1 (a), it is easy to remark that the $2k+1$ nodes located at the north boundary are connected to the $2k+1$ nodes at the south by wrap-around links are skewed $k$ positions. The same applies for the $2k+1$ nodes of the east boundary with the nodes of the west boundary. This remark simplifies the computation of the neighbors of the peripheral nodes by knowing their location without extra-cost of the modulo operation. Those nodes at the peripheral use wrap-around link. They will have a special treatment when it comes to our proposed all-to-all algorithm.

One of the problems with these special nodes is the length of their wrap-around links which is an issue to solve in the NoC architectures context. An efficient way to deal with the long link issue is proposed in [1], [22], and [23], where the long link is split into shorter segments connected by registers (See Fig. 2).

![Fig. 1 (a) Dense Gaussian Network $Q_3$ or $G_{3+4i}$](image1)

![Fig. 1 (b) Circulant Network $C_{25}(3,4)$](image2)
In the dense Gaussian NoC architectures $G_{(j+i+1)}$, the length of the wrap-around link is $k$ time the length of a normal link it can be implemented using $k-1$ registers. The register to register delay should not be longer than one clock cycle. In [1], the all-to-all broadcast algorithm the wrap-around links is pipelined using the $k-1$ register to reduce the distance effect. Another solution to the long link is proposed in [15], where the authors suggested folding the Gaussian networks.

### 2.2 Related Work

There have been several studies on all-to-all broadcast for different interconnection off-chip networks such as the torus interconnection networks [24], [25], [26], [27], [28], [29], and [30]. With the emergence of NoC architectures these studies are more focused now in developing collective communication primitives for NoC architectures. As we mentioned earlier dense Gaussian networks seem to be promising NoC architectures in the near future because of their very good topological properties.

Recently, Z. Zhang et al. [1] have proposed an all-to-all broadcast algorithm for Gaussian on-chip networks that achieve the minimum delay time but require $4k$ extra buffers per router, where $k$ is the network diameter. The size of the buffer is typically equal to the size of the message to be transmitted. This all-to-all algorithm uses broadcast trees of depth $k$ such as at step $d$ of the all-to-all algorithm; each node receives the messages from its $d$ perimeter (nodes at distance $d$). See Fig. 3.

![Fig. 2. Wrap-around long link in Dense Gaussian NoC](image)

![Fig. 3. Previous tree for Dense NoC $G_{(j+i)}$](image)

![Fig. 4. $4d$ Neighbors at distance $d$](image)

It has been shown as a property of the DGN in [1], [16] that the number of nodes at distance $0 < d \leq k$ is $4d$. We denote this property $p_1$. The broadcast tree presented in [1] is based on property $p_1$ by its construction (see Fig. 3).

In Fig. 3, the numbers on the links (in the four tree
branches) denotes the level of a node in the tree which is exactly the distance of the node from the root of the tree. These numbers correspond to communication steps in the all-to-all broadcast which consists of superposition of the same broadcast tree (the same structure) rooted at each node of the network. As consequence of property $p_1$ the structure of the proposed broadcast tree has the following property $p_2$: at each level $d$ of the broadcast tree structure there are $d$ North links, $d$ East links, $d$ South links, and $d$ West links. As an example in Fig. 3 at level 2 there are 2N, 2E, 2S, and 2W links.

As consequence of property $p_2$, we show here that the superposition of all the trees will not be link-free (some trees share the same communication link at a given communication step) when they are used in the context of the all-to-all. To see this fact, assume that a given node $x$ connect to a node $y$ by a North link (without loss of generality) in the tree rooted at node $(0, 0)$ denoted $T_{00}$. As a direct consequence of property $p_1$, each node $y$ will be at distance $d$ of $4d$ nodes which we denote $Nd_y$ (see Fig. 4).

Now let us focus on the $4d$ different trees rooted at the different nodes $Nd_y$ denoted $T_{Nd_y}$. The node $y$ is located at the level $d$ in all the different trees $T_{Nd_y}$ in $4d$ different branches (using different path from the root node $Nd_y$ to the node $y$). According to property $p_2$ the node $y$ is reached at level $d$ by $d$ North links, $d$ East links, $d$ South links, and $d$ West links ($4d$ different positions in the level $d$ in the $4d$ trees $T_{Nd_y}$). The North link connecting the node $x$ to the node $y$ at level $d$ is common to $d$ different trees. It is the same for the other links East, South, and West. It results that in the communication steps of all-to-all (except the first step) there will be link-conflict. Solving the link-conflict could by combining the different messages (coming from different trees) and send them on the same link. This will increase the size of the sent message and then larger buffers space is needed within the router. The peak for buffer requirements for the proposed all-to-all algorithm is in phase $d=k$, where $4k$ buffers of size the original message size are needed. Knowing the limited buffering resources in the NoC architectures, this all-to-all algorithm might not be suitable for NoC architectures.

In this paper we propose an all-to-all broadcast algorithm for dense Gaussian network suitable for NoC architectures from two aspects: first it achieves the lower bound of the all-to-all transmission delay and second it does not need any additional buffering in all the steps of the algorithm.

3 NEW BROADCAST TREE IN THE DENSE GAUSSIAN NETWORK

We propose a new family of broadcast trees NESW($0, 0$) rooted at the node $(0, 0)$ of the dense Gaussian NoC architectures. The same broadcast tree will be used to broadcast from any node $(x, y)$ because dense Gaussian network are node-symmetric. Broadcasting from node $(x, y)$, consists of forwarding data from the root of the tree NESW$(x, y)$ to each node along the broadcast tree. We will use a multi-node broadcast from each node to perform the all-to-all broadcast in the dense Gaussian NoC architectures. A multi-node broadcast operation is a simultaneous combination of broadcasts using all the trees NESW$(x, y)$ from all the nodes $(x, y)$ of the dense Gaussian network NoC architectures. One very important characteristic of this new family of trees NESW$(x, y)$ is what we call the NESW property. The NESW property ensures that during the multi-node broadcast steps there will be no link conflicts (no link used by more than one tree during a given broadcasting step).

Without loss of generality we show how to construct a tree NESW($0, 0$) rooted at node $(0, 0)$, we proceed in three phases. The first phase consists of partitioning the nodes of the dense NoC architectures in four distinct triangular areas plus the root node (at the center), namely NorthEast, SouthEast, SouthWest, and NorthWest. In the second phase we construct the NorthEast branch of the tree. In the third phase, we show how to derive the other branches of the tree using rotations. See Fig. 5. The four distinct triangular areas consist of the nodes $(x, y)$, such that $|x| + |y| < k$: NorthEast $x, y > 0$; SouthEast $x < 0, y < 0$; SouthWest $x < 0, y > 0$; NorthWest $x < 0, y > 0$.

3.1 Construction of the NorthEast branch

The NorthEast branch consists of $m = \left\lceil \frac{k-1}{2} \right\rceil$ sub-branches numbered from $b=1, m$. A sub-branch consists of a path which is a succession of links. In what follows we denote respectively, the North link, East link, South link, and West link, by respectively, $N, E, S, \text{and W}$. We also use the following notation: the path $N^0$ represents a path which consists of a succession of North link $j$ times.

<table>
<thead>
<tr>
<th>NorthEast Branch Construction Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N^2$,</td>
</tr>
<tr>
<td>For $b=1$ to $m$ {</td>
</tr>
<tr>
<td>$N^{(k-2b)}$, $E, S, S^{(k-2b)}$,</td>
</tr>
<tr>
<td>If (k is even) or (b&lt;m) <em>/change direction, next sub-branch</em>/</td>
</tr>
<tr>
<td>$E, N$,</td>
</tr>
<tr>
<td>}</td>
</tr>
</tbody>
</table>

As an example the NorthEast branch of the tree in Fig. 5 for the DGN NoC architectures $Q_4$ consists of $m = \left\lceil \frac{k-1}{2} \right\rceil = 2$ sub-branches numbered from $b=1, 2$. The sub-branch $b=1$ starts from the node $(0, 0)$, then follows $N^2$: an North link and then another $N$ (North link). Then follows the sequence of links $N^{(k-2b)}$, $E, S, S^{(k-2b)}$, the change direction $E$, $N$ to connect with the second sub-branch $b=2$. The second sub-branch $b=2$ consists of the following sequence of links $N^{(k-4-1)}$, $E, S, S^{(k-4-1)}$.

3.2 Derivation of the other branches by $\pi/2$ rotations

The SouthEast branch is obtained by applying a $\pi/2$ rotation (clock-wise) to the NorthEast branch. Applying $\pi/2$ rotation
maps the links N, E, S, and W to links respectively E, S, W, and N.

SouthEast Branch Construction Algorithm

\[
E^2,
\text{For } b=1 \text{ to } m \\
\{ E^{(k-2b)}, S, W, W^{(k-2b)} \\
\text{If } (k \text{ is even) or } (b< m) / \text{*change direction, next sub-branch*}/ S, E, \\
\}
\]

The SouthWest branch is obtained by applying a \( \frac{\pi}{2} \) rotation to the SouthEast branch. Finally, the NorthWest branch is obtained by applying a \( \frac{\pi}{2} \) rotation to the SouthWest branch (see Fig. 5).

SouthWest Branch Construction Algorithm

\[
S^2,
\text{For } b=1 \text{ to } m \\
\{ S^{(k-2b)}, W, N, N^{(k-2b)} \\
\text{If } (k \text{ is even) or } (b< m) / \text{*change direction, next sub-branch*}/ W, S, \\
\}
\]

NorthWest Branch Construction Algorithm

\[
W^2,
\text{For } b=1 \text{ to } m \\
\{ W^{(k-2b)}, N, E, E^{(k-2b)} \\
\text{If } (k \text{ is even) or } (b< m) / \text{*change direction, next sub-branch*}/ N, W, \\
\}
\]

In Fig. 5, the numbers on the links (NorthEast branch only for clarity sake) denotes the communication step numbers.

Note that the nodes \((k-1, 1), (1, -k+1), (-k+1, -1), \) and \((-1, k-1)\) belong respectively to the branches SouthWest, NorthWest, NorthEast, and SouthEast. Clearly, by using this process the constructed tree has 4 linear balanced-length branches. From the above branch construction algorithm we can compute the length of the branches in terms of number of link:

\[
\text{Branch}_{\text{Length}} = \left( \frac{k-1}{2} \right) (k + 3), \text{ if } k \text{ is odd or} \\
\text{Branch}_{\text{Length}} = \left| \frac{k-1}{2} \right| (k + 4) + 2, \text{ if } k \text{ is even.}
\]

Note that some special nodes such as node A and B belongs to both NorthEast and SouthWest branches. The same remark for the two nodes C and D, they belong to both SouthEast branch and NorthWest branch. Note also that these special nodes A, B and C, D are located at the same place relatively to their triangular area. Consequently, the nodes A (NorthEast branch), B (SouthWest branch), C (SouthEast branch), and D (NorthWest branch) are located at the same level in the broadcast tree NESW(0, 0). See Fig. 5 and 6. By construction of the tree, these special nodes occur once in each sub-branch except in the first sub-branch of every branch. Thus in every branch of the tree there will be exactly \( \left| \frac{k-1}{2} \right| - 1 \) of such nodes located at different levels in the tree.
We proceed by contradiction. Let \( d = s \). In a simultaneous correspondence to a level \( S \), all nodes send half size messages. There is still a problem, different special nodes per branch with the special nodes. Knowing that there are \( k \) steps involved during the steps involved with the special nodes. Clearly this will solve the redundant messages received by the special nodes, and reduce the communication cost to half during the steps involved with the special nodes. Knowing that there are \( \frac{k-1}{2} - 1 \) different special nodes per branch, there will be then \( \frac{k-1}{2} - 1 \) steps in the broadcast algorithm where the nodes send half size messages. There is still a problem, the one-to-all broadcast algorithm using NESW tree is far from the optimal which should performs the broadcast in \( k \) steps rather than \( \frac{k-1}{2} (k + 3) \) steps when \( k \) is odd. The NESW trees are not efficient to perform the one-to-all broadcast but we show later how they can be used to perform efficiently the all-to-all broadcast.

As we will see in the next section, our all-to-all broadcast algorithm performs simultaneous broadcasts using the NESW broadcast trees rooted at each of the nodes of the dense Gaussian NoC architectures. The important question is whether there are any link conflict, i.e. are there two or more broadcast trees sharing the same link at the same level? Lemma 1 gives a clear answer to this question.

**Lemma 1:** The family of NESW broadcast trees rooted at each node of the dense Gaussian NoC architectures is link conflict free at each level.

**Proof:** All the NESW broadcast trees rooted at each node of the dense Gaussian NoC architectures have the same structure. They are constructed from each other by applying a root translation. We proceed by contradiction. Let us consider a link \( d \) (North, East, West, or South) connecting nodes from a given level \( l \) to level \( l+1 \) in a tree rooted at node \((x_0, y_0)\) in branch \( br \). Assume that there exists another tree rooted at node \((x_1, y_1)\), where the same link \( d \) connects two nodes from level \( l \) to level \( l+1 \) in a same branch \( br \). Because the broadcast trees have the same structure then these two trees must be identical, i.e. \( (x_0, y_0) = (x_1, y_1) \). Now assume that the link \( d \) in the first tree belongs to another branch \( br' \neq br \) at the same level \( l \) in the second tree. That is not possible, because of the NESW property, the links at the same level of the tree are all of different direction (North, East, West, South), then we must have \( br' = br \).

The consequences of this lemma are first, in the all-to-all algorithm using NESW trees there are no link conflicts at all the steps. Second, at each step, the four messages arriving from different links will be routed to different links, i.e. each link will carry only one message. There is no need for extra buffers at the routers.

## 4 An All-to-all Broadcast Algorithm for the Dense Gaussian NoC Architectures

We propose an all-to-all broadcast algorithm for the dense Gaussian NoC architectures using a multi-node broadcast on the trees NESW\((x, y)\). The all-to-all broadcast algorithm is a collective communication that might be required by an application which concerns all the nodes in the NoC architectures. We assume that no other traffic can interfere in the network except the all-to-all traffic during the all-to-all operation (MPI-like paradigm at the application level).

We start by presenting a procedure called Communication as the basic operation in our all-to-all algorithm. The Communication procedure is executed by all the nodes of the network. As described in the following algorithm, we assume that the communication links are bi-directional using the all-port model, i.e. sending and receiving are performed in parallel on all four links of a node.

```
Procedure Communication \( l \), \( l_1 \), \( l_2 \), \( l_3 \) in \{North, East, South, West\}
Do in parallel
  Send the data received on the North link to \( l_1 \)
  Send the data received on the East link to \( l_2 \)
  Send the data received on the West link to \( l_3 \)
  Receive data on all links \( l_1 \), \( l_2 \), \( l_3 \), \( l_4 \)
```

As an example, Communication(South, West, North, East); performs the following action: the message received on the North link will be sent to the South link. The message received on the East link will be sent to the West

```
...}
```
link. The message received on the South link will be sent to the North link. The message received on the West link will be sent to the East link.

The following distributed all-to-all algorithm consists of BranchLength steps each of which is based on the basic communication procedure described above. The basic communication procedure is fairly simple and does not need any extra buffering at the routers. It consists of receiving data from one link and sending it on a given link. This simple operation can be easily implemented by the router hardware. Our all-to-all algorithm uses packet-switching which is the preferred transmission mode for NoC architectures [4].

```
Procedure All-to-All()
{
1. Do in parallel  //Initial step
{ Send the data on the four links
Receive the data on the four links }
2. Communication(South, West, North , East);
   m = (k-1)/2; //Number of Sub-branches
   For(b=1;b ≤ m; b++)
   {
     If (b==1) //Sub-branch 1 with no special nodes
   3.{ For(i=1; i ≤ k-2b; i++)
      {Communication(South, West, North, East);
      Communication(West, North, East, South);
      Communication(West, North, East, South); }
   } ElseIf (k is even)
   4.{ For(i=1; i ≤ k-2b; i++)
      {Communication(South, West, North, East); }
      Each node divides each of the four received messages to two halves L and R
      Communication(West(R), North(R), East(L), South(L));
      Communication(West(L), North(R), East(R), South(L));
      Each node recombines the four original messages L*+R
   } Else //k is odd
   5.{ For(i=1; i ≤ k-2b-1; i++)
      {Communication(South, West, North, East); }
      Each node divides each of the four received messages to two halves L and R.
      Communication(South(R), West(R), North(L), East(L));
      Communication(West(R), North(R), East(L), South(L));
      Each node recombines the four original messages L*+R
      Communication(West, North, East, South); }
   } For(i=1; i ≤ k-2b; i++)
   {Communication(South, West, North, East);}
   If (b < m)or (k is even)
   { //change direction to next sub-branch
      Communication(East, South, West, North);
      Communication(East, South, West, North);
   }
} //EndFor
} //EndProcedure
```

The presented all-to-all procedure is implemented and runs on every node of the dense Gaussian NoC architectures. It consists of \((k-1)/2 \times (k+3)\) synchronous steps, if k is odd or \((k-1)/2 \times (k+4) + 2\), if k is even. In step 1, each node sends its message on its four links (level one in the NESW trees). The second step consists of forwarding the received message from a given link to a specific link corresponding to the branch definition. As an example in step 2, Communication(South, West, North, East); performs the following action: the message received on the router North link will be sent to the South link (SouthWest branch). The message received the East link will be sent to the West link (NorthWest branch). The message received on the South link will be sent to the North link (NorthEast branch). The message received on the West link will be sent to the East link (SouthEast branch). See Fig. 5. The step 3, consists of forwarding the messages on the first sub-branch. In the case of the NorthEast branch, it consists of forwarding what is received from South link to the North link. Forwarding towards the North direction (first for loop). Then change the direction, and start forwarding towards the South (second for loop labeled 6). Note that in the sub-branch \(b = 1\) there is not special nodes. The communication procedure sends the whole message. In step 4 or 5 depending on \(k\) odd or even: Start by forwarding according to the sub-branch (normal steps, sending the whole message) then prepare for the special steps involving the special nodes, by dividing the message to two halves denoted \(L\) and \(R\) corresponding respectively to the Left and right part of the original message. As an example, the procedure Communication(West(R), North(R), East(L), South(L)) in step 4, performs the following actions: the Right part of the message received on the North link send it to the West link (SouthEast branch). The Right part of the message received on the East link send it to the North link (South-West branch). The Left part of the message received on the South link send it to the East link (NorthWest branch). The Left part of the message received on the West link send it to the South link (NorthEast branch).

The advantage of our all-to-all broadcast algorithm is its simplicity and easy implementation. Indeed, it is fully distributed, nodes do not need to know the coordinates of their neighbors, no extra-cost for neighborhood coordinates calculation as in the previous algorithm. Our all-to-all broadcast algorithm can be also implemented at the application layer of the network layered NoC architectures because the same procedure all-to-all runs in every node of the dense Gaussian NoC architectures.

## 5 Performance Evaluation

In this section we show the optimality of our all-to-all broadcast algorithm for the dense Gaussian NoC architectures in terms of transmission time.
5.1 Communication Model
Communication is assumed to be full-duplex and all-port. We suggest using the cycle as a unit of time measure to be generic and abstract from the router micro-architecture or frequency. For the purpose of comparison between both all-to-all algorithms, we use the same assumptions and model of communication as in [1]. It was assumed in [1] a Store-And-Forward (SAF) switching technique [47] and a linear communication model, i.e. the communication cost $T_{\text{com}}$ for sending a message of size $L$ bytes from a PE to a direct neighbor PE is: $T_{\text{com}} = t_s + t_w + L\tau$, where $t_s$, $t_w$, and $\frac{1}{\tau}$ are the startup time (message preparation time), switching time (arbitration time, and the switch traversal time) and the link traversal time (link bandwidth), respectively. In the following performance study of our proposed algorithm we use the lower bound analysis. Typical NoC architectures parameters are $t_s = 1$ cycle, $t_w = 2$ cycles, and $\tau = 1$ cycle/ link width [33], [36]. We assume that the link width is much larger than 1 byte and can easily reaches 16 bytes. To be conservative, we assume a link width of 8 bytes. The communication time is given by: $T_{\text{com}} = 3 + \frac{L}{8}$.

5.2 Analytical Performance
In this section, we analyze the communication delay of the newly proposed all-to-all broadcast algorithm for the dense Gaussian NoC architectures. Our all-to-all broadcast algorithm consists on simultaneous multi-node broadcasts using the constructed NEWS trees of depth $Branch_{\text{Length}}$ and rooted at every node in the network. It consists of $Branch_{\text{Length}}$ link conflict free steps among them $2\left(\frac{k-1}{2}\right)$ special steps, sending half of the message. In the following theorem 1, we show the communication cost of our all-to-all broadcast algorithm.

**Theorem 1.** The transmission delay for our all-to-all broadcast algorithm on the dense Gaussian NoC architectures is optimal:

$$TR_{\text{all-to-all}} = \left(\frac{k^2+k}{2}\right)L\tau.$$  

**Proof.** The optimal delay time for the all-to-all broadcast in any regular degree network topology (all the nodes have the same degree $r$) if the network size is $N$ nodes is given by: $\left(\frac{N-1}{r}\right)L\tau$.

Let us analyze the delay time for our all-to-all for the DGN NoC architectures with regular degree 4 and the number of node $N=2k^2+2k+1$.

If $k$ is odd, $T_{\text{all-to-all}} = \left[\left(\frac{k-1}{2}\right)(k+3) - 2\left(\frac{k-1}{2}\right)\right]L\tau + \left[2\left(\frac{k-3}{2}\right)\right]L\tau = \left(\frac{k^2+k}{4}\right)L\tau$.

If $k$ is even, $T_{\text{all-to-all}} = \left[\left(\frac{k-1}{2}\right)(k+4) + 2 - 2\left(\frac{k-3}{2}\right)\right]L\tau + \left(\frac{k^2+k}{2}\right)L\tau$. During the all-to-all algorithm, each node in the NoC architectures has to receive a total of $2k^2 + 2k$ different messages from all the other nodes. Because each node has 4 links that can receive in parallel, each link will receive a total of message $\left(\frac{k^2+k}{2}\right)$.

The transmission time of our all-to-all algorithm for dense Gaussian NoC architectures is then optimal in terms of transmission time as in the previous algorithm in [1].

As our all-to-all algorithm consists of $Branch_{\text{Length}}$ steps, where in each step each router receives four new messages from its four different input channels and then switch them to their four corresponding output channels. In what follows, we denote our all-to-all algorithm New and the one presented in [1] by PREV.

The total time for both all-to-all is:

- $Total_{\text{New}} = t_s + t_w \cdot Branch_{\text{Length}} + TR_{\text{all-to-all}}$.
- $Total_{\text{PREV}} = t_s + t_w \cdot 2 + TR_{\text{all-to-all}}$.

If SAF switching technique is used the total all-to-all time of PREV is less than New. In facts, PREV algorithm relies on buffering the messages (output buffers) at each router in order to overlap the switching time of a message with the transmission time of another message within the same router. In our New algorithm, when SAF is used, there will be no need for output buffers and consequently further reduction of buffer space is possible. If we assume cut-through technique for our algorithm, we will be able to overlap the switching time of an incoming message with the transmission of an outgoing message. In this case, $Total_{\text{PREV}}$ will be equal to $Total_{\text{New}}$ but there will be a need for an output buffer for each output channel. The required memory space for the output buffers for PREV is 4 times (4 output channels) the size of the message but New required memory space is 4k times the size of the message.

5.3 Simulation Results
In this section we present simulation results comparing our all-to-all algorithm New against PREV all-to-all algorithm. To compare the cost of the two different implementations, we used the same model as in ORION 2.0 [38] to estimate the space (area) occupied by the buffers (SRAM-based) on-chip.

Figure 7 shows a lower cost of our implementation in terms of buffer space on-chip. It shows also that the buffers area for New does not increase when the size of the network increases which is not the case for PREV.
We developed a simulation model using NS2 simulator to compare New and Prev algorithms in terms of latency (cycles) and the power dissipation (pf). We selected NS2 simulator for its flexibility and because most of the known NOC architectures simulators allow only specific network topologies such as ring, mesh, etc. We plugged in our NS2 simulation model the power model described in Nostrum NOC [39][40]. We considered only the power consumption in the buffers and the wires. We run the simulation model to measure the latency and the dynamic power consumptions by varying the size of the message (multiple of a 64 bits packet). We have also varied $k$ the diameter of the DGN($k$, $k+1$). For the power model, we used the parameters for 1GHz frequency and 0.18-micron CMOS as described in [40].

Fig. 8 shows that for small values of $k$ ($k=3$), the power dissipated by both algorithm is close to each other with a slight advantage for our New algorithm. When the size of the network increases, a clear advantage for our New algorithm is noticed. In fact the power consumption in the links is the same in both algorithms but buffers consumption in our algorithm is much less.

Fig. 9 shows that for small messages (from 1 to 2 packets) Prev algorithm has slightly less latency compared to New that is due to overlapping between switching time and transmission time in Prev. It shows also that for longer messages (3 and more packets) both Prev and New have a very close latency. That is because for large messages the switching time becomes negligible compared to the transmission time.

6 CONCLUSION

In this paper we have proposed a new efficient all-to-all broadcast algorithm for the dense Gaussian NoC architectures with reduced buffer requirements. Reduced buffer size and bufferless communication is a new research trend in NoC architectures. Our all-to-all algorithm uses special NESW trees. These trees guarantee at any step of the all-to-all broadcast algorithm all the communication links are contention free and hence no extra buffer space is required unlike previous all-to-all broadcast algorithm where the extra-buffer size scales with the size of the Gaussian network.

Our all-to-all broadcast algorithm is based on a simple communication procedure that can be easily implemented in the router hardware. It can be also implemented at the application layer of the NoC architectures network layered architecture. The simulation study showed that our all-to-all algorithm New has less implementation cost in terms of buffer area occupation on-chip, and less dynamic power consumption compared to the previous algorithm Prev [1]. The experiments showed that for large messages (more than two packets per message) both New and Prev have a very close latency. For its better performance in terms of the size occupied by the Gaussian network on-chip, the dynamic power consumption, our all-to-all algorithm is more suitable for dense Gaussian NoC architectures compared to the previously known algorithm.

ACKNOWLEDGMENT

This work was supported by Sultan Qaboos University Grant.

REFERENCES

Complexity in the Design of Parallel Numerical Algorithms, IEEE
[45] S.L. Johnsson, Communication Efficient Basic Linear Algebra Compu-
tations on Hypercube Architectures, J. Parallel Distributed Computing,

AUTHOR BIOGRAPHIES

Abderezak Touzene  received the BS degree in computer science from University Algiers in 1987, MS degree in computer science from Paris-Sud Orsay University (France) in 1988 and Phd degree in computer science from Institut polytechnique de Grenoble (France) in 1992. He is an Associate. Prof. in Sultan Qaboos University, Oman. Dr. Touzene’s research includes Interconnection Networks, Parallel and Distributed Systems, Network-On-chip, Wireless and Mobile Networks, Performance Evaluation, Cryptography.
Dr. Touzene is a member of the IEEE Computer Society.