Runtime connection-oriented guaranteed-bandwidth network-on-chip with extra multicast communication service

Faizal Arya Samman*

Universitas Hasanuddin, Fakultas Teknik, Jurusan Teknik Elektro, Jl. Perintis Kemerdekaan Km. 10, Makassar 90245, Indonesia

ARTICLE INFO

Keywords:
Network-on-chip
Wormhole Cut-Through Switching
Quality of service
Guaranteed-bandwidth service
Best-effort service
Connection-oriented protocol
Multicast communication

ABSTRACT

This paper presents a flexible runtime connection-oriented guaranteed-bandwidth Network on Chip (NoC). Comparing with a standard time-division multiplexing (TDM) method, our local ID-based method provides better flexibility to establish dynamic runtime connections. A specific pre-designed algorithm for finding a conflict-free scheduling, as commonly used in the TDM-based method, is not needed. The contention problem is solved with the hardware solution based on the locally organized message identity (ID), in which flits belonging to the same stream packet will have the same unique/local identity-tag (ID-tag) on each communication link. The ID-tags of each stream will vary locally over communication links and are updated. The updating is organized by ID-tag mapping management units. The routing is organized using runtime programmable routing reservation table. In addition, the proposed methodology supports also a deadlock-free multicast routing service.

© 2013 Elsevier B.V. All rights reserved.

1. Introduction

On-chip interconnection networks are an interesting alternative communication infrastructure to bring a new paradigm to design and develop a Multiprocessor System-on-Chip (MPSoC) and Chip-level Multiprocessor (CMP) systems. Bus systems, which is traditionally used as communication fabrics for SoC systems, tend to perform bottleneck, especially if they are loaded with a large number of high-speed processing elements. Instead of using the traditional bus systems as communication media among processing element (PE) units, on-chip interconnection networks is proposed, reflecting a concept of scalable shared communication media.

Several multimedia applications for MPSoCs consist of some communication edges that must be performed with a certain communication bandwidth. Video/audio streaming data transmission from a core to one or to many cores needs a constant transmission rate. Performance degradation at one the communication edges could reduce the overall application performance or even could break the multimedia applications. Hence, a specific service of the network is required to guarantee the data bandwidth of the video/audio streaming. This paper will present a methodology on how the bandwidths of unicast and multicast communication edges in an on-chip radio system application benchmark [9] can be well guaranteed. The methodology proposes a runtime virtual circuit configuration technique, where connections are established during application execution time. Moreover, the technique supports also a runtime multicast communication service.

Guaranteed-bandwidth (GB) service can be implemented by using end-to-end connection establishment technique, where a stream header reserves required bandwidth during connection set-up phase before sending the video/audio streaming. By further applying a policy where every network link cannot be consumed by considered traffic exceeding its maximum capacity, then (long-term) saturated network condition can be avoided (non-blocking traffic flow is guaranteed). Guaranteed-service can be implemented by allowing multiple packets to share the same link. The link sharing can be realized by using a data multiplexing technique. The shared link configuration in a network is also commonly called as switched virtual circuit (SVC) configuration. However, the guaranteed-bandwidth method is only suitable for NoC-based multicore processor systems, where processors intensively sending and receiving streaming data that require expected constant end-to-end communication rates.

This paper proposes one of the SVC method by using dynamic local identity (ID) assignment technique. By using this technique, a message is not split into packets, but it is split into flow control digits (flits) with extra bit fields for dynamic unique/local ID and flit-type control label per flit. The idea results in a novel routing paradigm, where the network routes flits instead of packets (“routes flits not packets”).

The rest of the paper is organized as follows. Section 2 will present the state-of-the-arts of the switched virtual circuit configuration methods. In this case, we compare our method with TDMA-based data multiplexing technique. Section 3 presents the
2. State-of-the-art of data multiplexing techniques for NoCs

2.1. NoCs with TDMA technique

A commonly used method to provide guaranteed-service for NoCs is a pipeline circuit switching based on the Time-Division Multiple Access (TDMA) method. Æthereal [12], Asynchronous clockless MANGO NoC [1], circuit switched PNoC [4], Sonics [19], DSPIN [11], Nostrum [10] and a NoC SMT Switch [8] are NoCs examples that use such methodology. Fig. 1(a) presents the conceptual view of the TDMA method. The link connecting the input and output port of the routers is shared by four packets, i.e. stream packet A, B, C and D. Each packet establishes a virtual circuit configuration based on time slots allocation on the outgoing port. In the figure, we assume that the link has 8 time slots. The more time slots are allocated for a packet, the more bandwidth (BW) it reserves. Thus, the packets A, D, B and C reserve 50%, 25%, 12.5% and 12.5% of the maximum link bandwidth capacity, respectively. A packet allocated at time slot S3 on a link must be allocated to time slot S1 on the next link. Based on Fig. 1(a) for instance, packet D allocated to time slots S1 and S2 on the link must be allocated to time slots S2 and S3 on the next link.

2.2. NoCs with IDMA technique

In this section, we introduce a concept based on local identity (ID) division multiple access (IDMA) technique. Fig. 1(b) presents the concept, in which local ID slots can be reserved by single data stream as its ID-tag. The local ID tag appears on every flit and is updated every time the data stream acquires the next link. Flits belonging to the same stream will always have the same local ID. In order to guarantee a correct routing function, an ID Management Unit must index every reserved ID slot by identifying the previous ID slot (IDN) number 1 (its new ID-tag), and is identified by the ID Slot Table as a packet from input port 5 having previous ID tag 0 in the router R1. The message D reserved also 25% of the maximum link bandwidth capacity (Bmax). In the next router R2, the stream/messages are routed based on their current/new ID-tags. Thus, the packet D flows from output Port 1 (East) of the router R1 to the input Port 3 (West) of the next router R2 with new ID-tag 1. The number of available ID slots reflects the maximum number of stream/messages allowed to form switched virtual circuit configurations on the link. The bandwidth can be guaranteed by further implementing a connection-oriented communication protocol, where the requested BW attached on a header flit bit fields is used to reserve the expected end-to-end communication bandwidth over the network links.

2.3. Comparisons of the multiple access methods

The TDMA-based switching requires a pre-design time-slot allocation algorithm to achieve a conflict-free routing and scheduling. UMARS + algorithms [3], TDM-based Virtual Circuit Configuration (VCC) Method [9], and a time-slot allocation algorithm made for μSpidergon NoC [2] are the NoC examples that use the time-slot allocation algorithm. The IDMA-based method does not need such time-slot allocation algorithms, because the local ID slot on each outgoing link is reserved and allocated autonomously by header flits of a streaming data during application execution time (flexible runtime autonomous switched virtual circuit reconfiguration). The same technique could be certainly applied to the TDMA-based method, but the probability in which the header flit fails to establish connection is very high especially in a very high traffic situation. The need for the time-slot allocation algorithm in the TDMA-based method is due to the conflict-free requirement.

A NoC design methodology that is based on three kernels, i.e. traffic classification, flit-based switching and path pre-assignment and link-BW setting has been introduced [7]. The traffic are classified into guaranteed-latency (GL), guaranteed-bandwidth (GB) and best-effort (BE) traffic. The GL traffic have stringent maximum delay requirement from data injection until data acceptance. The GB traffic requires constant end-to-end communication bandwidth, while the BE traffic does not have bandwidth requirement neither stringent data transfer latency. The link allocation (path assignments) for the GL and GB traffic is static or computed off-line at design time [7]. For a new application, the path assignment must be done again at design time. Therefore, the proposed methodology is not suitable for application mapping, where the applications are known after chip-manufacturing.

An extra bit field for dynamic local/unique ID-tag and an extra field to identify the flit type of each flit are attached on each flit. The extra bit fields are useful to guarantee that flits belonging to contributions of the paper. The basic infrastructure required to implement the proposed method is described in Section 4, i.e. the required standard packet format and the microarchitecture. Section 5 describes the connection-oriented multicast communication protocol. The ID-based routing mechanism and the ID Management Scheme are described in detail in Section 6. Section 7 shows an experimental result. In this section, we evaluate our data switched virtual circuit configuration method on a radio system application benchmark from Nokia [9]. The synthesis results of the NoCs are presented in Section 8. Brief discussions about the implementation issues of the multiple access methods are presented in Section 9. Section 10 concludes the work and presents the main advantageous features of the proposed method.
the same packet will have the same local ID-tag on every local communication channel. The following advantages can be then achieved.

- The concept of Wormhole Cut-Through Switching can be introduced, where wormhole packets can be interleaved flit-by-flit or can cut-through at flit-level, resulting in interesting and unique performance characteristics [16] compared to traditional wormhole switching method.
- The concept of Hold–Release Tagging Mechanism for Multicast Scheduling Policy can be introduced [13,14,17], which theoretically results in a unique solution for a runtime deadlock-free multicast routing method [15].
- The concept of Flexible Runtime Connection-Oriented Guaranteed-Bandwidth for Quality-of-Service can be introduced, which is explained in this paper, where the switched virtual circuit configuration can be made during application execution time.

There are also other multiple access methods for NoCs that have been introduced so far such as Spatial-Division Multiple Access (SDMA) [6] and Code-Division Multiple Access (CDMA) [18] methods. However, since both methods are minor in the NoC research area, we do not discussed these methods and compare them in this paper.

3. Contribution

The state-of-the-arts of the switched virtual circuit configuration methods or commonly called as multiple access methods are presented in this paper. The multiple access technique based on the locally organized message identity (ID) for multicast-enabled NoC has been introduced in our previous works [13,14]. However, the previous works implement the IDMA technique by using a best-effort communication protocols without guaranteed-bandwidth service. In the case of runout of available ID slots, the previous works must allow packet dropping mechanism to avoid packets stall. Hence, retransmission of data should be undertaken when such situation occurs. This mechanism will be time-consuming. Therefore, this paper introduces a concept of guaranteed-bandwidth multicast NoC with runtime connection-oriented communication protocol.

By using the connection-oriented protocol, the messages will not be sent to the NoC before a virtual circuit (connection), which guarantees the expected average end-to-end data throughput has been successfully established. The advantages of using such concept are (1) the packet dropping will never happen, (2) guaranteed-bandwidth service can be further implemented, and thus (3) network will not be saturated, since the considered traffic are not allowed to consume a link exceeding its maximum bandwidth capacity.

A simple data transfer technique by applying local addresses (labels) has been also presented [5]. However, the local labels presented in the work must be pre-computed at design time for each traffic in an application by using a path-based labeling algorithm. The routing paths for a target application is also statically computed by using a precount procedure to count the maximum number of the used communication pairs. We propose a runtime (dynamic) local message identity (ID) technique, which offers a flexibility without a pre-processing algorithm. Our runtime dynamic local labeling method will be suitable for post-manufacture MPSoC application mappings, where in general applications are preliminary unknown before SoC fabrications.

4. Basic Infrastructure for the IDMA method

4.1. Packet format

Fig. 2(a) shows a data stream (message), which consists of many flow control digits (flits). Each flit comprises of a 7-bit control field and data/header information in 32-bit data word (Note: Message in our NoC is not partitioned into several packets). Fig. 2(b) shows the detail packet format used in the NoC. Each flit can be identified through its 3-bit flit type field as stream header (head), a stream databody (dbod), a tail (tail) and response/status (resp) flit and its 4-bit local ID-tag. The source and target addresses of router nodes are attached respectively on the header flits. The “ReqBW” information is attached on the headers and is used for BW reservation (BW allocation). The “RespBW” information attached on the tail flit is used for BW deallocation.

A unicast message will have only one header flit, while a multicast message for N number of target/destination nodes contains N number of header flits. As shown in Fig. 2(b), the multicast message has N number of header flits for N number of multicast destination nodes, i.e. \((X_i, Y_i), k \in \{1, 2, \ldots, N\}\), where each header is for each target/destination node. The Ext field is extension bits that can be used for other requirements in the future. Each target node \(k\) will send back a response flit with ID-tag number “1111” to inform the status of the guaranteed-bandwidth connection made by a header flit for target node \((X_i, Y_i)\). Table 1 shows the binary encoding of the flit types.

**Definition 4.1.** Each flit coming from input port \(n\) can be formalized as \(F_n\) (tag, type), where type = \{head, dbod, tail, resp\} and tag = \{0, 1, 2, \ldots, N_{slot} − 1\}. \(N_{slot}\) is the amount of number of available ID slots per link. Because there are 4 bits for ID-tag field then we will have \(N_{slot} = 2^4 = 16\).

Beside routing information, i.e. source \((X_i, Y_i)\) and target \((X_i, Y_i)\) addresses, requested communication bandwidth (ReqBW) information is attached to each header flit. The 12 least-significant bits of the header and tail flits fields are used to identify the requested communication bandwidth (ReqBW) of the data stream/packet. The ReqBW information attached in a tail flit is used to remove the bandwidth reservations. Header and tail flits, which belong to the same multicast packet, will have equal ReqBW value. If a header is routed to a new outgoing link, then an amount of bandwidth (BW) (equal to ReqBW value) and an ID slot are reserved for the data stream/packet. The other header flits belonging to the same message with the previous header flit will not reserve again new BW space and new ID slot, when they enter the same outgoing link.

The response flit type as shown in Fig. 2(b) contains a connection status field. This field is used to indicate the status of the connection made by the header flit, whether the connection establishment is successful or unsuccessful. The response flit assigned with ID-tag “1111” is used to sent back a connection status from a target node \((X_i, Y_i)\) to a source node \((X_i, Y_i)\). If the connection establishment is successful, i.e. ID slot and a number of ReqBW bandwidth are successfully reserved on the routing paths from the receiver node \((X_i, Y_i)\) to the sender node \((X_i, Y_i)\), then payload/databody flits can be injected from the sender node with a constant injection rate of ReqBW as attached in the header flit. The tail flit will remove the BW and ID slot reservation after the end of the data injection.

4.2. Generic microarchitecture

The generic microarchitecture of our NoC router (switch) is presented in Fig. 3. The figure overviews the router’s components, the data paths and control paths connected to the router’s components. The descriptions of the microarchitecture are given in this subsection.

**Definition 4.2.** The set of input–output port is \(N_p = \{1, 2, \ldots, N\}\), where \(N\) is the total number of ports. The port number is defined as \(n \in N_p\).
In our current microarchitecture, we have $N = 5$ such that $n \in N_{IO} = \{1, 2, 3, 4, 5\}$. In general, the router consists of 4 main components located in every input and output port $n$. At every input port $n$, there are 2 components, i.e. a FIFO buffer ($F_{in}$) and a Routing Engine with Data Buffering (REB) ($R_{in}$). In the current microarchitecture, a static XY routing algorithm is implemented in the REB unit. At each output port $n$, there are also 2 components, i.e. a Multiplexer with ID Management (IDM) Unit ($M_{in}$) and an arbiter unit ($A_{in}$).

### Table 1

<table>
<thead>
<tr>
<th>Hex</th>
<th>Binary</th>
<th>Flit type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>&quot;000&quot;</td>
<td>Not data</td>
</tr>
<tr>
<td>4</td>
<td>&quot;100&quot;</td>
<td>Header for GB streams</td>
</tr>
<tr>
<td>5</td>
<td>&quot;101&quot;</td>
<td>Databody for GB streams</td>
</tr>
<tr>
<td>6</td>
<td>&quot;110&quot;</td>
<td>Tail flit for GB streams</td>
</tr>
<tr>
<td>7</td>
<td>&quot;111&quot;</td>
<td>Response/status flit</td>
</tr>
</tbody>
</table>

### Definition 4.3

Set of components at each input and output ports $n$ is defined as $\Psi_n = \{F_{in}, R_{in}, M_{in}, A_{in}\}$. $N_{IO} = 5$, the total set of components in a router is $\bigcup_{n=1}^{5} \Psi_n$.

#### 4.2.1. Buffer slots

The depth of the FIFO buffers in our current microarchitecture is set to 2 registers only. Beside implementing buffer slots in the FIFO buffer, we also implement a single buffer slot in the REB (Routing Engine with Buffer) unit. The purpose of the buffer slot insertion is to improve router performance. By inserting the buffer slot, each data is delayed for only one stage, i.e. output arbitration stage. Since the data is buffered soon after a routing direction has been computed, the data can be switched out to an output port next cycle after output arbitration stage. Without the buffer slot, the data must be delayed for two stages before making the output switch or link traversal stage, i.e. route compute stage and output arbitration stage.

#### 4.2.2. Routing engine

Beside the single buffer slot, a runtime-programmable routing reservation table and a routing state machine are implemented in the REB unit. In other words, the XHiNoC’s routing engine combines a routing look-up table and a routing machine. This combination allows us to establish connection at runtime, which is certainly also enabled by the attachment of the extra bit fields for dynamic local ID and flit type label on every flit of streaming data.

In our current microarchitecture, we use static/deterministic XY (X-First) routing algorithm, which is implemented in the routing state machine. Flits of stream packets will be firstly routed to X-direction (east or west), then routed to Y-direction (north or south).

#### 4.2.3. Routing and arbitration control signals

Beside datapaths, there are also two control paths, i.e. routing request signal and arbitration signal paths, which are intercon-
ected in the crossbar switch of the router. The paths are used to control and synchronize the data flows and data switching.

The formal definition of the routing request signals is explained as follows.

**Definition 4.4.** \( r(i,j) \) is single-bit routing request signal from \( yi \) to \( Aj \), where \( i,j \in N_o = \{ 1, 2, 3, 4, 5 \} \). \( r(i,j) \) is a digital (binary) signal, then \( r(i,j) \in \{ 0, 1 \} \).

According to Definition 4.4, we can further define that the routing request binary vector from input port \( i \) to the output port \( j \) is \( r(i) = [r(i, 1), r(i, 2), r(i, 3), r(i, 4), r(i, 5)] \). For example, when a flit from input port 2 requests a routing to output port 4, then we have \( r(2, 4) = 1 \), or the routing request vector is \( r(2) = [00010] \). When the request from the input port 2 is a multicast request to output ports 1, 3 and 4 for instance, then the routing request vector is \( r(2) = [10110] \).

Another control path used in the router is the routing arbitration path, which is generated by the arbiter unit and sent to the REB unit to grant a routing request. The formal definition of the routing arbitration signals is explained as follows.

**Definition 4.5.** \( a(i,j) \) is single-bit arbitration signal from \( Aj \) to \( yi \), where \( i,j \in N_o = \{ 1, 2, 3, 4, 5 \} \) as a feedback signal to grant the routing signal \( r(i,j) \). \( a(i,j) \) is a digital (binary) signal, then \( a(i,j) \in \{ 0, 1 \} \).

When the REB component at input port \( i \) sends routing request signal \( r(i,j) \) to all arbiter units, then it will receive a routing acknowledge signal \( a(i,j) \) from the arbiter units. If a data flit is buffered in the FIFO, it will then be routed and buffered again in the REB unit. The routing signal is sent to the arbiter unit located at the requested output port. If the REB unit receives a routing acknowledge signal from the arbiter unit, it will be switched out to the outgoing link in the next cycle.

According to Definition 4.5, we can further define a routing arbitration control vector from output port \( j \) as \( a(j) = [a(1,j), a(2,j), a(3,j), a(4,j), a(5,j)] \). For example, when a flit from input port 2 requests a routing to output port 4, then the arbiter unit will generate an arbitration control signal \( a(2,4) = 1 \), or the arbitration control vector is \( a(4) = [01000] \). When there are more than one request to the output port 4, for example 3 requests from input ports 1, 2 and 5, then the arbiter at the output port 4 will successively generate 3 arbitration control vector, i.e. \( a(4) = [00100] \) for request from input port 1, \( a(4) = [01000] \) for request from input port 2 and \( a(4) = [00010] \) for request from input port 5. The arbitration (input selection) will be rotated by the arbiter unit flit-by-flit, port-by-port. **Note:** Only one element of the arbitration vector can be set to 1. Otherwise the flits will conflict and the routed data flit will not valid.

Based on the microarchitecture and control vectors presented in this section, our XHiNoC router can switch 5 flits from 5 different input–output ports in parallel. In general, the formal description of the control path matrices for the routing and arbitration control signals is presented in Definition 4.6.

**Definition 4.6.** Based on Definitions 4.4 and 4.5, a time-dependent routing request matrix \( R \) and an arbitration matrix \( A \) can also be defined in the following:

\[
R = \begin{pmatrix}
    r(1,1) & r(1,2) & r(1,3) & r(1,4) & r(1,5) \\
    r(2,1) & r(2,2) & r(2,3) & r(2,4) & r(2,5) \\
    r(3,1) & r(3,2) & r(3,3) & r(3,4) & r(3,5) \\
    r(4,1) & r(4,2) & r(4,3) & r(4,4) & r(4,5) \\
    r(5,1) & r(5,2) & r(5,3) & r(5,4) & r(5,5)
\end{pmatrix}
\]

\[
A = \begin{pmatrix}
    a(1,1) & a(1,2) & a(1,3) & a(1,4) & a(1,5) \\
    a(2,1) & a(2,2) & a(2,3) & a(2,4) & a(2,5) \\
    a(3,1) & a(3,2) & a(3,3) & a(3,4) & a(3,5) \\
    a(4,1) & a(4,2) & a(4,3) & a(4,4) & a(4,5) \\
    a(5,1) & a(5,2) & a(5,3) & a(5,4) & a(5,5)
\end{pmatrix}
\]

5. Connection-oriented multicast protocol

The runtime connection-oriented guaranteed-bandwidth multicast routing protocol implemented in our NoC consists of four main phases, i.e. connection establishment, connection status response, data transmission and connection termination (see Fig. 4).

5.1. Connection establishment

The first phase is connection establishment, where the data producer node sends a request of a data transmission by injecting multicast header flits one-by-one to multiple destination nodes. If a data producer core in the NoC will send a multicast packet to \( n \) output ports, then \( n \) number of header flits are injected one-by-one by the data producer core, where each header brings information about the address of each data consumer core and the requested communication BW (ReqBW).

As presented in Fig. 4(a), a request for multicast connection setup. Each of them \( h_1, h_2 \) and \( h_3 \) is sent to destination cores at node (1,1), (2,2) and (3,3), respectively. As shown in the figure, for example \( h_1 \) header flows from core \( A \) to core \( F \) through the network and reserves ID slots 2, 3 and 1 in the links \( e_3, e_4 \) and \( e_0 \), respectively.

In our current NoC, the ID-tag field is 4 bits, resulting in 16 available local ID slots per link. In this case \( M \) is set to \( 11111111 \) or \( \text{‘}0\text{‘X}0\text{‘} \). Every destination node \( j \) can detect the ID-tag of the incoming header flit \( h_j \) to analyze if the connection setup is successful or not. If on a certain link, a header flit of the stream packet \( A \) fails to reserve an ID slot, then the header flit will be allocated to ID slot 5 (or use the ID-tag number \( M \) = \( \text{‘}11111111\text{‘} \)). Hence, ID tag 15 (\( 11111111 \) is reserved as an escaping tag, where the failed header flits will use this tag, and normal header flits should not be allocated to this ID slot number.

The BW reservation can be unsuccessful because of 2 reasons, i.e. because no more available (free) ID slot on the link \( e_j \) (case 1), or free BW space, which can meet the expected communication BW, is not sufficient (case 2). Once a header flit is allocated to ID slot \( M \), then it will always be routed afterwards with ID slot \( M \) starting from the link on which it fails until it reaches the destination node.

5.2. Connection status response

The second step is the response phase, where every destination node that receives a header flit \( h_j \) from the source node will analyze the header flit to know if the header has successfully established the multicast connection or failed. Afterwards, the processor or hardware core at destination node \( j \) sends back a response flit \( s_j \). The data consumer node informs the data producer node about the status of the connection establishment made previously by the header \( h_j \) (see Fig. 4(b)), by writing any information (special code) on the response flit. As shown in the figure, all status response flits flow through the network with ID-tag number 15 (\( \text{‘}11111111\text{‘} \) or \( \text{‘}0\text{‘X}0\text{‘} \)). The successful connection will implicate that the expected data rate or communication bandwidth (BW) of the multicast stream packet can be guaranteed.

As presented in Fig. 4(b), every target node, i.e. core \( D \) at node (1,1), core \( B \) at node (2,2) and core \( F \) at node (3,1) receives a header.
flit $h_1$, $h_2$ and $h_3$, respectively. Afterwards, each target node analyzes the accepted header flit and sends back a response flit to the source core $A$, where the response flits flow through the NoC links with ID-tag $M = 15$. The response flit indicates the successful of the connection establishment.

5.3. Data transmission

The third phase is the data transmission. If the producer has known that the multicast connection is set up successfully and the bandwidth on each reserved communication link is guaranteed, then it will send the multicast stream packet into the NoC through the same path set up previously by the header flits as presented in Fig. 4(c). The payload data flits are sent with the same ID-tag with the previously routed header, such that the data flits can track the paths configured by the headers.

If one of the multicast connections is not successfully established, then the producer node will firstly terminate the multicast connection by sending a tail flit. Afterwards, it will inject again new header flits to establish new connections. However, this runtime connection-oriented communications can potentially cause a livelock situation when a connection is never established.

In order to guarantee a livelock freedom from such situation, a new protocol must be implemented in the network interface or in the application layers. The protocol could for example limit the number of attempts to repeat the connection setup. For instance, In the application layer, the software protocol will only make 5 attempts to establish connection. Afterwards, a node that has made 5 attempts and always fails to established connection could broadcast a special header flits for all nodes. The special header flits could be indicated for example through the “ReqBW” field, which contains all-zero-bit.

After receiving the special header flits, the other nodes could for example renew and reduce their allocated bandwidth. Hence, the failed node could use the free BW space, although the free BW space is probably lower than its requirement. This decision should be wisely made in the software protocol layer. When the BWs of all communications are scaled down, the application performance will degrade, but probably the application could still run well enough.

In the case of ID slot run out, all nodes could also divide their streams into some partitions. Thus, the stream of the failed node

---

**Fig. 4.** Connection-oriented multicast routing protocol.
will also have a chance to use a free ID slot for some cycles, and re-
lease again or share the ID slot for other stream flows. This chal-
 lenging issue could also be solved with other solutions. Since this
issue is suitable implemented on application layers, we will not
deeply discuss it in this paper.

5.4. Connection termination for successful connection establishment

In the fourth phase as presented in Fig. 4(d), the tail flit is in-
jected to close or terminate the multicast connection from the
source to target node at the end of the data sending phase. The tail
flit will remove the bandwidth allocation and ID slot reservation of
the considered stream packet, which has been allocated and re-
erved previously by the considered header flit.

5.5. Connection termination for unsuccessful connection establishment

In the previous subsection, connection termination mechanism
for the case where the connection establishment is successful has
been explained. In the case of unsuccessful connection establish-
ment, a specific mechanism is also provided. When a header flit
fails to reserve a free ID-slot or an expected BW space in an in-
termediate node between the source and target node. The failed head-
er flit will always use ID-tag 15 starting from the intermediate
node until the target node without making a new ID-tag and rout-
ing table reservations.

After the failed header flit reaches the target node and its ID-tag field is indicated with ID-tag number 15 (it means that it fails to establish connection), the target node will send a response flit to inform the source node that the connection setup fails. This information can be obtained from the binary label attached on the con-
nection status field in the response flit. After the response flit reaches the source node, the source node will send a tail flit to tear
down the connection from the source node only until the interme-
diate node. Afterward, the tail flit will be dropped at the considered outgoing port in the intermediate node. A few cycles after sending the tail flit, the source node can start sending a new header flit to establish again a new connection.

6. ID-based routing and ID management

Every stream packet reserves one ID slot as its ID-tag on each
communication link, and flits belonging to the same message will have the same unique ID tag on the same link. In our NoC router,
there are 5 input–output ports, i.e. East(E), North(N), West(W),
South(S) and Local(L), which are assigned to port number 1, 2, 3,
4 and 5, respectively. The ID-based routing and ID management mecha-
nism can be divided into several procedures as follows.

1. ID-tag updating at the output port made by a header flit (Sec-
tion 6.1, Fig. 5).
2. Routing table reservation at the input port made by a header flit (Section 6.2, Fig. 5).
3. ID-tag indexing made by data body (payload) flits (probably by a header flit) at the output port (Section 6.3, Fig. 6).
4. Routing table indexing made by data body (payload) flits at the input port (Section 6.4, Fig. 6).
5. ID-tag and routing table terminations made by tail flits at the input and output ports (Section 6.5).

Routing table reservation (Section 6.2) and routing table index-
ing (Section 6.4) are parts of the ID-based routing mechanism. While ID-tag updating (Section 6.1) and ID-tag indexing (Sec-
tion 6.3) are parts of the ID management mechanism.

6.1. ID-tag updating

This mechanism is visually presented in Fig. 5. At the East out-
put port of the router node R1, a header flit with ID-tag 1 coming
from West(W) input port is being switched out to the East E output
port. The ID-tag updating procedure is explained in the following
items.

1. As presented in Fig. 5, the header flit with ID-tag 1 is coming from West (W) input port.
2. The Arbiter unit from East output port selects the flit by giving W signal arbitration to the multiplexer such that the header flit is switched out to the East output port.
3. When the ID-Management (IDM) unit in the multiplexer detects the type of the flit (header type in this case), then it looks for a free ID slot in the ID Slot Table and has found that ID slot number 2 is free ('F'). The nID column in the ID Slot Table represents the reservable ID slot that can be used by a flit as its new ID-tag. The Sta column is the status of the slot number that can be set to free ('F') or used ('U').
4. At the same cycle, the IDM unit writes the new ID-tag number, i.e., 1, and from which port the header flit comes, i.e., W input port, into the slot number 2 of the column ID and From of the Table, respectively.
5. Also at the same cycle, the UsedBW register is updated (Used-
BW = UsedBW + reqBW) by adding the current value of the used BW with the required BW of the packet attached on the header flit field.
6. The slot number 2 is then used by the header flit as its new ID-
tag to flow on the link and the status of the ID slot changes from free ('F') to used ('U') state.

Note: If another header flit coming from W input port with ID-
tag 1 is switched to the East output port in the next time periods (which means that it belongs to the same packet with the header flit shown in Fig. 5), then the header flit will not reserve a new ID-tag, but it will directly be allocated to ID-slot 2. This procedure will be explained in Section 6.3.

6.2. Routing table reservation

This mechanism is also visually presented in Fig. 5. At the West input port of the router node R2, now the header flit is buffered in the FIFO buffer at the West (W) input port. The routing reservation procedure is explained in the following items.

1. As presented in Fig. 5, the header flit with ID-tag 2 is now buf-
fered in the data buffer of the REB unit and is being routed by the
REB.
2. When the REB detects the type of the flit (header flit in this case), then the Routing State Machine (RSM) and Routing Reservation Table (RRT) are activated.
3. Routing slot number 2 in the RRT is selected in accordance with the ID-tag of the header flit.
4. When the header type is detected, then the routing information will be selected from the RSM unit.
5. At the same cycle, address information (Xt, Yt) attached in the
header flit is fed through to the RSM to make a routing decision. In this example, we see that the routing direction is S (South). Hence, the column S in the RRT is set such that the routing information ‘0 0 0 0 1 0’ is obtained as shown in the figure.

6.3. ID-tag indexing

This mechanism is visually presented in Fig. 6. At the East out-
put port of the router node R1, a payload flit with ID-tag 1 coming
from West (W) input port is being switched out to the East E output port. The ID-tag indexing procedure is explained in the following items.

1. As presented in Fig. 6, the databody flit with ID-tag 1 are coming from West (W) input port.
2. The Arbiter unit from East output port selects the flit by giving W signal arbitration to the multiplexor such that the databody flit is switched out to the East output port.
3. When the ID-Management (IDM) unit in the multiplexor detects the type of the flit (databody type in this case), then it looks for combination of (ID, From) in the ID and From columns of ID Slot Table. In this case, the databody flit gives combination (1, W).
4. The IDM unit finds the combination in the slot number 2. Hence, the slot number 2 is then used by the databody flit as its new ID-tag to flow on the link.

6.4. Routing table indexing

This mechanism is visually presented in Fig. 6. At the West input port of the router node R2, now the payload flit is buffered in the FIFO buffer at the West (W) input port. The routing indexing procedure is explained in the following items.

1. As presented in Fig. 6, the databody flit with ID-tag 2 is now buffered in the data buffer of the REB unit and is being routed by the REB.
2. When the REB detects the type of the flit (databody flit in this case), then only the Routing Reservation Table (RRT) is activated, and the routing information will be selected from the RRT unit.
3. Routing slot number 2 in the RRT is selected in accordance with the ID-tag of the databody flit. Hence, the routing decision is obtained by fetching the routing information from the slot number 2, i.e. ‘1 1 0 1 0’ as shown in the figure.

6.5. Bandwidth, ID-tag and routing terminations

When a tail flit of a packet is switched out to an output port, then the tail flit will make ID-tag indexing procedure as explained in Section 6.3. But at the last cycle, the bandwidth reservation, the ID and From column in the slot number according to the ID-tag number of the tail flit will be set free. The status of the ID-slot number will also be set from used (U) to free (F) state.

When a tail flit of a packet is routed from an input port, then the tail flit will make routing-indexing procedure as explained in Section 6.4. But at the last cycle, the Routing Information in the
routing slot number according to the ID-tag number of the tail flit will be set free.

7. Experimental results

In this section, the connection-oriented multicast method will be verified on a radio system application benchmark from Nokia [9]. The simulation is made at register-transfer level (RTL simulation) by using an HDL (hardware description language) simulator. The allocation of each task after application mapping on 2D 4 × 4 mesh topology is presented in Fig. 7. In general, the application consists of 11 communication edges. Two of them i.e. communication a and h are multicast data communication. Communication j is a broadcast (one-to-all) data communication, where the core at node1 broadcasts data to all other cores. Communication k is all-to-one data communication, where core at node1 receives data from all other cores.

The injection rate in our NoC is controlled by inserting jitters between two consecutive data flits. Jitter is a zero flit inserted in one cycle period. The more jitters inserted between two consecutive data flits, the lower the setpoint of the bandwidth (BW). If one data flits are injected with \( N_{jt} \) number of jitters in between, then the BW setpoint will be

\[
B_{\text{max}} = \frac{1}{N_{jt} + 1} \text{flit/cycle}
\]

In this simulation, the data frequency is set to 1 GHz, then 1 flit/cycle is equal to 1 GHz × 4 × 4 = 4000 MB/s (1 word consists of 4 bytes). However, the maximum BW capacity of our NoC link is only \( \frac{1}{4} \) flit/cycle. Hence, the maximum data BW is \( B_{\text{max}} = 4000 \times \frac{1}{4} = 2000 \text{ MB/s} \). Because our NoC router can perform 5 simultaneous IO connections, then the maximum BW capacity of the NoC router is \( 5 \times 2000 \text{ MB/s} = 10 \text{ GB/s} \). Thus, if we expect a BW of 512 MB/s, then we can set \( N_{jt} = 6 \), resulting in BW setpoint of
4000 \times \frac{1}{2} = 571.43 \text{ MB/s}. If we set \(N_{hit} = 7\), then the BW setpoint will be 4000 \times \frac{1}{7} = 500 \text{ MB/s}. Finer BW granularity can be controlled by a compute element, which actually produces data sent to network interface.

Fig. 8 shows the measurement of the expected, setpoint and actual measured BW for the on-chip radio system application. It looks that the minimum bandwidth requirements of all communication edge can be met. The expected bandwidth is obtained from Fig. 7. The setpoint BW is made by the source node based on the insertion of the number of jitters to set the BW requirement as explained before. The actual bandwidth is measured at the destination node.

For the sake of simplicity, the ID slot reservation results of the communication \(a\) – \(j\) and communication \(k\) is separated. Fig. 9(a) presents one of many possible local ID slot reservations made autonomously by the header flits for communication edge \(a\) until \(j\). If the communication \(j\) is performed firstly, the multicast header flits will be sent flit-by-flit to all other nodes. Thus, this broadcasting communication edge will reserve the first local ID slots on every communication media (i.e. ID slot 0). The other communication edges (communication \(a\)–\(i\)) will then reserve the rest ID slots that are not reserved by communication \(j\). Fig. 9(b) depicts also one of many possible combinations of the local ID slot reservations for communication edge \(k\), when this connection is set up after all other communication edges have established their connections. This communication edge will also reserve the local ID slots that have not reserved by the communication \(a\)–\(j\). The bottom part of the figure exhibits the ID slot reservation in the Local output port of the router node 1. The runtime local ID slot reservation is very flexible, because the header flits, which reserve the ID slots autonomously, will just check available (free) ID slots that can be utilized as their ID-tag from the ID Slot Table.

Reserved BW space on an output port is defined as the total accumulated BW reserved by some streams on the output port. When a stream with a certain BW flows into an output port, the stream’s BW will be allocated to the MIM unit at the output port in accordance with the requested BW space indicated on the header flit of the stream (please see again the packet format shown in Fig. 2 in Section 4.1). The more streams flow into an output port, the more reserved BW spaces allocated at the output port.

Fig. 10 presents the reserved BW spaces on every output port of all 16 router nodes. Fig. 10(a) shows the BW reservation for communication edge \(a\) until \(j\), while Fig. 10(b) presents the BW reservation for communication \(j\). The figures represent a congestion situation on each router node and hotspot locations in the network. In Fig. 10(a), we can see that the hotspots occur at node6, where total BW consumption of all output port in the node is about 4100 MB/s, or about 41% of total maximum BW capacity (10 GB/s) of the router. In Fig. 10(b), we can see clearly that the hotspots are located in the local output port of the router node 1, in the south output port of the router node 5 and of the router node 9, respectively.

8. Synthesis results

The synthesis results of the XHiNoC prototypes with multicast guaranteed-bandwidth (GB) service and the Æthereal NoC with the same technology size are presented in Table 2. The synthesis is made using 65-nm CMOS standard-cell technology library from Taiwan Semiconductor Manufacturing Company (TSMC). The NoC with GB service is synthesized with target frequency of 1.47 GHz (0.68 ns clock period).

For general overview, let us see the synthesis result of the Æthereal NoC [12]. The maximum frequency to transfer data in the Æthereal NoC, which combines the BE and GT services with 32-bit word size, is 500 MHz using a 130-nm standard-cell technology, resulting in an aggregate bandwidth of 5 \times 500 \text{ MHz} \times 32 \text{ bits} = 80 \text{ Gbit/s}. Hence, the bisection bandwidth for 32-bit data width of the Æthereal’s link is 2 \times 80 \text{ Gbit/s} = 160 \text{ Gbit/s}. The total logic area of the Æthereal NoC is 0.2600 mm².

For synthesis with 65 nm CMOS technology, the XHiNoC with guaranteed-bandwidth (GB) service has logic cell area of about 0.0457 mm² with 32-bit data width can be clocked until 1.47 GHz. With selected 1 GHz data frequency, the aggregate bandwidth of the XHiNoC router (static routing, 2-depth FIFO, 32-bit word size, 5 I/O ports) is 5 \times 1000 \text{ MHz} \times 32 \text{ bits} \times 1/2 = 80 \text{ Gbit/s}. Hence, the bisection bandwidth (for 32-bit data width and 1 GHz data frequency) of the XHiNoC’s link is 2 \times 80 \text{ Gbit/s} = 160 \text{ Gbit/s}. The aggregate bandwidth is divided by two because of two-cycle delay between every flit during data link traversal pipeline stage, or the maximum data rate per link in XHiNoC is 1/2 flit/cycle. Therefore, in order to gain the same bisection bandwidth, the XHiNoC should be clocked twice faster than the Æthereal NoC.

Table 3 shows the power estimation of the XHiNoC router with connection-oriented guaranteed-bandwidth service. The power analysis is made with target frequency 1.47 GHz with maximum switching activity, i.e. all input ports are actively switched with 1.47 GHz data frequency.

9. Discussions

The finest BW granularity presented by the IDMA-based multiple access method is independent from the available local ID slots. Based on our current implementation, we use 12 bits from header flit bit field for requested BW of every stream/message capable of providing \(2^{12} = 4096\) BW granularity. In our current microarchitecture, we set the maximum link BW of 2000 MB/s as \(O_{XTHP}\). By using a TDMA method, the BW granularity depends on the number of time slots on each link.

In the TDMA NoC version with pre-processing time-slot scheduling algorithm, a central reconfigurator unit is required to program and allocate a time slot for each message on every port of the routers to guarantee the conflict-free routing configuration. A global network view is required in this case. Thus, the pre-computed time slot allocation algorithm must be processed before running the real application. The XHiNoC concept does not require such pre-processing scheduling algorithm, because the connection can be established at runtime during application execution time. The central reconfigurator unit is also not required any more, and replaced by the runtime-programmable ID management table and runtime-programmable routing reservation table, which are hardcoded into the router hardware.

10. Conclusions

A guaranteed-bandwidth multicast-enabled NoC with the runtime connection establishment method has been presented in this paper. By implementing the connection-oriented data
communication, the total accepted bandwidth for each communication link in the NoC will not exceed its maximum bandwidth capacity. Each stream flow must register its bandwidth requirement on a BW management unit at each outgoing port of the NoC routers. A stream will not be allowed to flow on a link, which has no more free BW space. In this situation, the NoC will not be saturated, and communication rate of each stream flow can be guaranteed accordingly. An input port does not need to differentiate the incoming data streams with different bandwidth requirements. This is also the reason why the guaranteed BW communication can work with a simple 2-slot FIFO buffer.

**Fig. 9.** One of many possible runtime local ID slot reservation configurations for each communication.

**Fig. 10.** Number of bandwidth reservations at each outgoing port of all 16 network nodes.

**Table 2**

Comparison of the XHiNoC and Æthereal NoC.

<table>
<thead>
<tr>
<th></th>
<th>XHiNoC</th>
<th>Æthereal NoC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technology size</td>
<td>65 nm</td>
<td>130 nm</td>
</tr>
<tr>
<td>Technology vendor</td>
<td>TSMC</td>
<td>No info</td>
</tr>
<tr>
<td>Target frequency</td>
<td>1.47 GHz</td>
<td>500 MHz</td>
</tr>
<tr>
<td>Total logic cell area</td>
<td>0.0457 mm²</td>
<td>0.2600 mm²</td>
</tr>
<tr>
<td>Bisection BW (32-bit)</td>
<td>160 Gbit/s</td>
<td>160 Gbit/s</td>
</tr>
</tbody>
</table>

**Table 3**

Power estimation of the XHiNoC using 65-nm CMOS technology library.

<table>
<thead>
<tr>
<th></th>
<th>Synthesis result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target frequency</td>
<td>1.47 GHz</td>
</tr>
<tr>
<td>Est. net switch, power</td>
<td>1.9800 mW</td>
</tr>
<tr>
<td>Est. cell intern, power</td>
<td>39.3734 mW</td>
</tr>
<tr>
<td>Est. cell leakage power</td>
<td>2.2853 μW</td>
</tr>
</tbody>
</table>

NoC routers. A stream will not be allowed to flow on a link, which has no more free BW space. In this situation, the NoC will not be saturated, and communication rate of each stream flow can be guaranteed accordingly. An input port does not need to differentiate the incoming data streams with different bandwidth requirements. This is also the reason why the guaranteed BW communication can work with a simple 2-slot FIFO buffer.
The advantageous feature and characteristic of the NoC is the simplicity to design the guaranteed-bandwidth router. The unicast or multicast connection configuration and termination are flexible and made autonomously at runtime by header flits. The experimental result presented in this paper has shown that the IDMA method works well and can be an alternative solution to improve the quality-of-service for NoCs.

Acknowledgments

The author gratefully acknowledges the comments, helpful suggestions and positive critics made by the anonymous reviewers, and DAAD (Deutscher Akademischer Austausch-Dienst, German Academic Exchange Service) that has awarded the author with DAAD-Scholarship to obtain his doctor of engineering (Dr.-Ing.) degree at Technische Universität Darmstadt in Germany. The author would also like to thank LOEWE-Zentrum AdRIA in Fraunhofer Institute LBF Darmstadt for further cooperation within Project AdRIA (Adaptronik-Research, Innovation, Application) funded by Hessian Ministry of Science and Arts, and Prof. Thomas Holllstein at Tallinn University of Technology for valuable discussions about networks-on-chip topic.

References


Faizal Arya Samman was born in Makassar, Indonesia. He received his Bachelor of Engineering degree in Electrical Engineering from Gadjah Mada University at Yogyakarta, Indonesia in 1999. In 2002, he received his Master of Engineering degree with Scholarship Award from Indonesian Ministry of National Education in Control and Computer System Laboratory and in Inter-University Center for Microelectronics Research, at Bandung Institute of Technology in Indonesia. In 2002, he was appointed to be a research and teaching staff at Hasanuddin University in Makassar, Indonesia. From 2006 until 2010 he received Scholarship Award from DAAD (Deutscher Akademischer Austausch Dienst – German Academic Exchange Service) to pursue doctoral degree at Technische Universität Darmstadt in Germany. From 2010 until 2012, he was a postdoctoral fellow within the research cooperation framework between Darmstadt University of Technology and Fraunhofer Institute LBF in Darmstadt in LOEWE-Zentrum AdRIA (Adaptronik-Research, Innovation, Application). He is now a research and teaching staff at Department of Electrical Engineering, Universitas Hasanuddin in Makassar. His research interests include network-on-chip microarchitecture, adaptive multiprocessor systems, programming models for multiprocessor systems, design and implementation of analog and digital electronic circuits for adatronic and control system application as well as energy harvesting systems, wireless sensor nodes, wired/wireless distributed control systems.