Chapter 1

BUILDING PREDICTABLE SYSTEMS ON CHIP: AN ANALYSIS OF GUARANTEED COMMUNICATION IN THE AETHEREAL NETWORK ON CHIP

Om Prakash Gangwal, Andrei Rădulescu, Kees Goossens,
Santiago González Pestana and Edwin Rijpkema
Philips Research Laboratories, Eindhoven, The Netherlands
{O.P.Gangwal, Andrei.Radulescu, Kees.Goossens,
Santiago.Gonzalez.Pestana, Edwin.Rijpkema}@philips.com

Abstract: As the complexity of Systems-on-Chip (SoC) is growing, meeting real-time requirements is becoming increasingly difficult. Predictability for computation, memory and communication components is needed to build real-time SoC. We focus on a predictable communication infrastructure called the Æthereal Network-on-Chip (NoC). The Æthereal NoC is a scalable communication infrastructure based on routers and network interfaces (NI). It provides two services: guaranteed throughput and latency (GT), and best effort (BE). Using the GT service, one can derive guaranteed bounds on latency and throughput. To achieve guaranteed throughput, buffers in NI must be dimensioned to hide round-trip latency and rate difference between computation and communication IPs (Intellectual Property). With the BE service, throughput and latency bounds cannot be derived with guarantees. In this chapter, we describe an analytical method to compute latency, throughput and buffering requirements for the Æthereal NoC. We show the usefulness of the method by applying it on an MPEG-2 (Moving Picture Experts Group) codec example.

Keywords: Networks-on-chip, Systems-on-chip, Time division multiplexing, Real-time systems, Predictable systems, Guaranteed throughput and latency connections, Best effort connections, Analysis and Verification of Networks-on-chip.

1. INTRODUCTION

As systems on a chip (SoC) grow in size and complexity, the current ways of system interconnect, such as buses and switches, cannot be used anymore,

because of, e.g., scalability and layout problems. For such complex systems, networks on chip (NoCs) have emerged as an interconnect solution. Some examples of NoC are SPIN (Adriahantenaina, Charlery, Greiner, Mortiez and Zeferino, 2003; Guerrier and Greiner, 2000), Æthereal (Goossens, van Meerbergen, Peeters and Wielage, 2002; Rădulescu, Dielissen, González Pestana, Gangwal, Rijpkema, Wielage and Goossens, 2005; Rijpkema, Goossens, Rădulescu, Dielissen, van Meerbergen, Wielage and Waterlander, 2003), Nostrum (Millberg, Nilsson, Thid and Jantsch, 2004), SoCBUS (Wiklund and Liu, 2003), QNoC (Bolotin, Cidon, Ginosar and Kolodny, 2004), aSOC (Liang, Swaminathan and Tessier, 2000), and others (Benini and De Micheli, 2001; Benini and De Micheli, 2002; Dally and Towles, 2001; Karim, Nguyen and Dey, 2002).

Most of the current interconnects, as well as NoCs have been built to offer best-effort (BE) communication services. BE communication infrastructures are not analyzable. Therefore, they require simulations to verify if the specified requirements are fulfilled. Because for complex chips, the interconnect is a central component in the system (Goossens, Gangwal, Röver and Niranjan, 2004), complete system simulations are required for system verification. Covering worst-cases for all configurations is not possible through simulations, because they are based on sample (demanding) inputs, which are never guaranteed to cover worst-case and corner cases. Problems that may appear during simulations are resolved by adjusting parameters in one or several of the many arbiters. If any change, system has to be resimulated again. There are three main problems with such systems: 1) long simulation times at each change, 2) numerous changes because of interdependences which lead to change side effects, and 3) worst-case behavior is not necessarily covered.

To solve these problems, we advocate the use of throughput and latency guarantees (Goossens et al., 2004; Goossens et al., 2002; Rijpkema et al., 2003). Each IP (Intellectual Property) module (i.e., computation and memories modules) can then be designed in isolation, because the interconnect requirements are made explicit. As the communication has a guaranteed behavior, the composed system will function according to the specifications provided all IP modules meet their specifications (correct by construction system) (Goossens et al., 2004). If IP modules have predictable behavior, the system behavior can be formally verified, without the need of simulations. If IP modules do not have predictable behavior, providing guarantees in the interconnect is still useful, because of the system compositionality resulted from offering guarantees: the system does not need to be simulated as a whole, but simulating only IP modules is enough. Moreover, there are no interdependencies, and, therefore, modifying parts of the system does not affect other parts of the system.

In this chapter, we focus on verifiable systems without a need of simulations. We define a model to characterize traffic of streaming IP modules, which

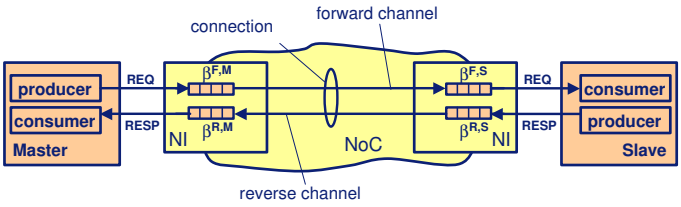


Figure 1-1. Connection example.

are characterized by the fact that they produce/consume data periodically. Using this and the Æthereal NoC's guaranteed-throughput service, we show how to compute the latency and throughput for the worst-case. We also compute the lower-bound sizes of the buffers between the NoC and the IP modules that guarantee the latency and throughput requirements are met. All these computations have been implemented in a verification tool, which is used by the Æthereal design flow (Goossens, Dielissen, Gangwal, González Pestana, Rădulescu and Rijpkema, 2005; Goossens, González Pestana, Dielissen, Gangwal, van Meerbergen, Rădulescu, Rijpkema and Wielage, 2005) to dimension and configure the NoC to satisfy the application requirements. We illustrate the use of the verification tool by applying it to an MPEG-2 (Moving Pictures Expert Group) codec example.

The chapter is organized as follows. In the next section, we describe the basics of the Æthereal NoC, focusing on the guaranteed-throughput and -latency communication services. In this section, we also define a communication model for the IP modules, and introduce some notation used in the chapter. In Section 3, we use these models to derive the throughput resulting from a given NoC for which the slots have been allocated. In Section 4, we compute the lower-bound sizes of the buffers between NoC and the IP modules. Further, in Section 5, we derive the latency that results from a given system consisting of a NoC and its attached IP modules. Our, throughput, buffer size, and latency formalizations and their implementation in a verification tool are shown in use by means of an MPEG-2 codec example in Section 6. We present our conclusions in Section 7. To ease reading, we also include in Section 8 a list of symbols used throughout the paper.

2. AN ANALYTICALLY VERIFIABLE SoC MODEL

In this section, we first describe the Æthereal NoC, focusing on the aspects that impact NoC analysis. Then, we list the conditions that IP modules need to satisfy to enable (sub)system analytical verification.

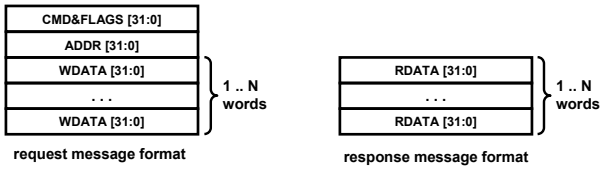


Figure 1-2. Example request and response message formats.

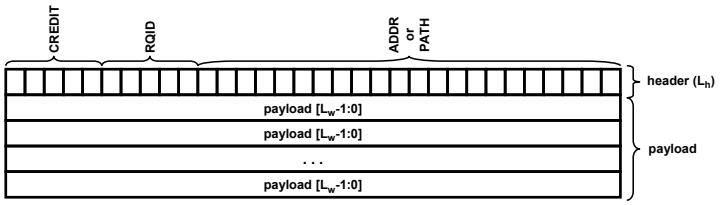


Figure 1-3. Æthereal packet format.

2.1 The Æthereal NoC Model

The Æthereal NoC (Rădulescu and Goossens, 2004; Rijpkema et al., 2003; Goossens et al., 2002) provides communication services on *connections* (Rădulescu and Goossens, 2004). As shown in Figure 1-1, connections are between two IP modules: one master, which is the module initiating the communication, and one slave which is the target module responding in the communication[1]. On each connection, we follow existing on-chip communication protocols, such as AXI, Advanced eXtensible Interface, (ARM Ltd., 2003), OCP, Open Core Protocol, (OCP International Partnership, 2003), or DTL, Device Transaction Level protocol, (Philips Semiconductors, 2002), and implement a *transaction*-based communication. That is, the masters issue *request messages*, consisting of a command (e.g., read/write), flags (e.g., burst length, mask, etc), address, and possibly write data (see Figure 1-2). Requests are transported via the NoC to the slave, which interprets and executes them, possibly issuing a *response message*, consisting of read data or acknowledgments/error flags (see Figure 1-2). In the current analysis, we use a simplified model where no acknowledgments/error flags are included.

From a NoC point of view, the request and response messages are just data which is packetized and transported over the NoC. The packet header (see Fig-

[1]More complex connections are possible, e.g., between one master and multiple slaves, but this is outside the scope of this chapter. For further information on the types of connections, please refer Rădulescu and Goossens, 2004

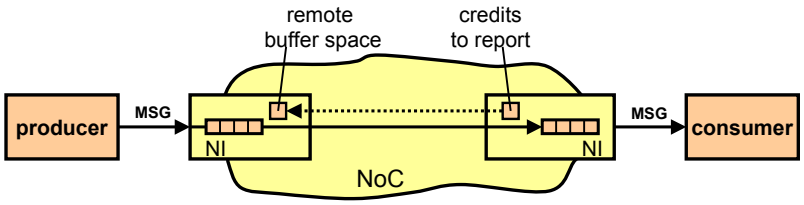


Figure 1-4. Credit-based flow control.

ure 1-3) added by the NoC has L_h words (in the current Æthereal implementation, $L_h = 1$), where the *word* is the unit of application data that is transferred on a single clock edge, and is measured in L_w number of bits (currently, in Æthereal a word consists of $L_w = 32$ bits)[2].

A connection consists of two *channels*: one *forward* channel, on which request messages are transferred, and one *reverse* channel, on which response messages are transferred. For each channel, there are two buffers that decouple IP modules from the NoC: one buffer in the network interface (NI) accepting messages in the NoC, and one buffer at the NI delivering messages to the destination IP module (see Figure 1-1). For a connection c_i, these buffers are denoted $\beta^{F,M}$, $\beta^{F,S}$, $\beta^{R,S}$ and $\beta^{R,M}$ for the buffers associated to the forward channel at master and slave sides, and those associated to the reverse channel at the slave and master sides, respectively. As shown in Figure 1-1, for each channel there is a producer (the master for a forward channel, and the slave for a reverse channel), and a consumer (the slave for a forward channel, and the master for a reverse channel).

The NoC provides *credit-based end-to-end flow control* (Tanenbaum, 1996) for every channel in the NoC. This means that at the producer's NI, there is a counter ("remote buffer space" in Figure 1-4) tracking the available space in the buffer at the consumer NI. Initially, this counter is set to the size of the consumer NI's buffer. Whenever the producer NI sends a word, the counter is decremented, and, if it reaches zero, no data is allowed to be sent to prevent buffer overflow at the consumer NI. When the consumer's NI delivers messages to the consumer IP module, another counter ("credits to report" in Figure 1-4) is incremented. This credit value needs to be sent to the producer's NI, to let

[2] A glossary of symbols is provided at the end of this chapter.

the "remote buffer space" counter correctly follow the consumer's NI buffer empty space[3].

This implies that besides application messages, credit information is also transported for every channel in the network. In our implementation, and, hence, also in our model, the credit information is transported in the packet header (see Figure 1-3). If data is transported on the same connection in the same direction in which credits must be sent, the credit is piggybacked on the created packets in the header (see Figure 1-3). If there is no data to be sent, empty packets are sent (i.e., consisting of only headers with credit information). Because of implementation reasons (fixed number of bits in the header), there is a maximum amount of credits that can be sent with one packet: M_{FC}. Consequently, the total amount of credits that can be transported on the NoC is a function of the number of packet headers that are sent.

Throughput and latency guarantees are provided using *time-division multiplexed circuits* (Rijpkema et al., 2003). Communication streams are mapped to connections, for which time *slots* in a slot table are reserved. For each slot, a circuit is set up between the producer and the consumer that communicate with each other. These circuits are dedicated to only the producer and the consumer involved, and, therefore, any interference between different connections is prevented. By using time-division multiplexing, the circuits are changed at each time slot. This allows link bandwidth to be shared between multiple connections.

To improve link utilization even further, we implement *pipelined* circuits. That is, basic units of data (i.e., the amount of data that fits in a slot) are transferred across consecutive links in consecutive slots. For example, in Figure 1-5 we show links L1, L2, L3, L4 and L5, for which slots X1, X2 and Y are reserved. Each of the slots on consecutive links are allocated consecutively (e.g., X1 has reservations in slots 1, 2, 3 and 4 for L1, L2, L3 and L4, respectively). In the bottom part of the figure, we show how data is transferred in the network following the slot reservations.

All of the NoC components (routers and NIs) run at the same frequency f_{noc} (500 MHz for Æthereal), corresponding to a clock period of $T_{noc} = 1/f_{noc}$ (2 ns for Æthereal). As already mentioned, all links have the same link width L_w. This results in a raw link bandwidth of $B_L = L_w \times f_{noc}$ (16 Gbit/s, or 500 Mwords/s).

[3] An alternative way of preventing overflow at the consumer's NI buffers is to rely on the link-level flow control. This would be possible for best-effort communication, however, data could wait in the NoC if a consumer does not consume data fast enough, and NoC congestion and/or deadlock may also occur, disturbing NoC functionality. For guaranteed communication, there is no link-level flow control, and, hence, credit-based flow control is the only way to prevent buffer overflow in the case consumer's behavior is not fully known.

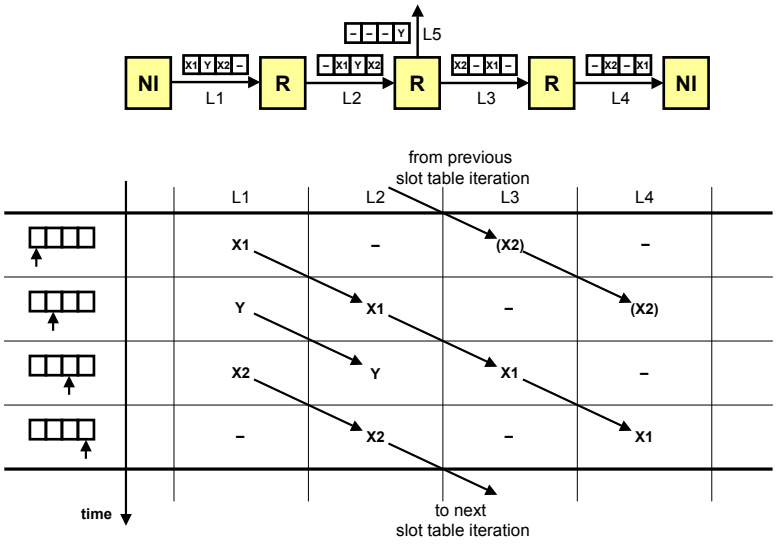


Figure 1-5. NoC with pipelined circuits using slots.

All NoC links have associated slot tables of equal sizes. We denote the slot sequence in a slot table with $\mathcal{S}$, and slot table size with $|\mathcal{S}|$. All the slots have an equal size: L_s (a slot has L_s words of L_w bits, which are transferred in $T_s = L_s \times T_{noc}$ seconds). For the Æthereal NoC, $L_s = 3$ words, and $T_s = 6$ ns. Note that a slot should be large enough to at least accommodate a complete packet header, because the packet header contains the routing information, and this information is needed to forward the slot to the correct destination ($L_s > L_h$).

Given a slot table size, we define $B_s = B_L/|\mathcal{S}|$ to be the bandwidth associated to a reserved slot, and $B_w = B_s/L_s$ to be the bandwidth associated to a reserved word.

The slot allocation and assignment of each connection c_i are stored in the network interfaces. For a connection c_i, there are two slot allocations: $\mathcal{S}_i^F \in \mathcal{S}$ and $\mathcal{S}_i^R \in \mathcal{S}$, for the forward and reverse channels, respectively.

The slot allocation for each link in the NoC is correct when (1) the number of allocated slots does not exceed the slot table size, (2) every slot of a link is allocated to at most one channel, and (3) when a channel traverses several links, the slots allocated for those links are consecutive.

2.2 The IP Module Model

On a connection c_i, IP modules are assumed to produce and/or consume data in bursts, distributed uniformly in time. That is, application bursts always come within a fixed-length periodic time interval T. As shown in Figure 1-6, we consider two cases:

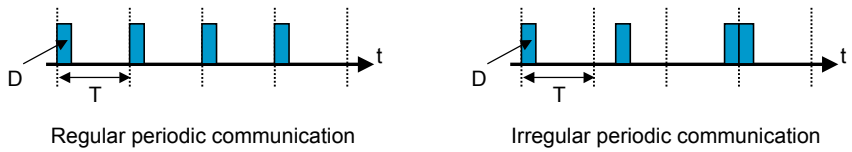


Figure 1-6. Periodic communication model.

regular, when the time between any consecutive data transfer is exactly T,

irregular, when data can be transferred at any time within a period T.

The IP module behavior is modeled by the data rate and burst size. The data rate for a connection c_i, measured in link-width words per second, is denoted for writes and reads by $R^{Wr}_{IP,i}$ and $R^{Rd}_{IP,i}$, respectively. This includes the application data (i.e., read or write data), but not the bandwidth required by the command, command flags, and address. The reason to specify only the application data is ease of specifications, as it allows to focus only on the way the application communicates, without being linked to any particular protocol or protocol implementation.

Burst sizes, denoted by $L^{Rd}_{DATA,i}$ and $L^{Wr}_{DATA,i}$ for a connection c_i, represent the amount of data that is transferred in a single read or write transaction (i.e., using a single read or write command), respectively. In our current model, a read transaction consists of a request containing a read command on the forward channel, and a response with read data on the reverse channel. A write transaction consists of a request containing a write command and write data on the forward channel, and no response on the reverse channel.

Together with the number of words needed to encode the command and its address $L^{Rd}_{CMD,i}$ and $L^{Wr}_{CMD,i}$ (2 words for the Æthereal NoC for both read and write commands), we can fully characterize the messages. For convenience, we use the command to data ratio, $\gamma^{Rd}_i = L^{Rd}_{CMD,i}/L^{Rd}_{DATA,i}$ and $\gamma^{Wr}_i = L^{Wr}_{CMD,i}/L^{Wr}_{DATA,i}$. Knowing γ^{Rd}_i and γ^{Wr}_i, we can compute the command and address rate as $R^{Rd}_{CMD,i} = R^{Rd}_{IP,i} \times \gamma^{Rd}_i$ and $R^{Wr}_{CMD,i} = R^{Wr}_{IP,i} \times \gamma^{Wr}_i$.

As an example, let us consider a connection c_i with $R^{Rd}_{IP,i} = 12$ Mwords/s, and $L^{Rd}_{DATA,i} = 16$ words. Then $\gamma^{Rd}_i = L^{Rd}_{CMD,i}/L^{Rd}_{DATA,i} = 2/16 = 0.125$. The resulting command and address rate is $R^{Rd}_{CMD,i} = R^{Rd}_{IP,i} \times \gamma^{Rd}_i = 12 \times 0.125 = 1.5$ Mwords/s.

From IP rate and burst sizes, we derive the period with which the IP module produces and/or consumes data. For a connection c_i, we assume these periods are identical for the master and slave IP modules attached to c_i. To capture the possible difference in read and write patterns on a connection, the IP periods are defined separately for reads and writes as $T^{Rd}_{IP,i} = L^{Rd}_{DATA,i}/R^{Rd}_{IP,i}$ and $T^{Wr}_{IP,i} = L^{Wr}_{DATA,i}/R^{Wr}_{IP,i}$, respectively.

Flow control is sent in the opposite direction compared to data (be it commands, addresses or application data). The credit stream must be enough to compensate for the data stream.

2.3 Notation

In this section, we further define operations and notation to help in our analysis. We use the channel type *ch* (i.e., F=forward and R=reverse) and computation type *comp* (i.e., P=producer, C=consumer, M=master, S=slave) as superscripts, and attributes to specialize the symbol *attr* (e.g. CMD=command, DATA=data, I=input, O=output) and connection index *conn_id* as subscripts for a given symbol:

$$Symbol^{ch,comp}_{attr,conn\_id} \tag{1-1}$$

We introduce two new operators: $\oplus$ and $\ominus$ for addition and subtraction modulo (denoted as %) $|\mathcal{S}|$, respectively, as follows:

$$s \oplus s' = (s + s')\%|\mathcal{S}| \tag{1-2}$$

$$s \ominus s' = (|\mathcal{S}| + s - s')\%|\mathcal{S}| \tag{1-3}$$

For each connection c_i, we define for both forward and reverse channels a set $\mathcal{F}_i^{ch}$ containing the blocks of contiguous slots allocated for that connection. A block from slots s to s' (including s and s' inclusive) is described by a tuple containing the first slot s and the block length:

$$\begin{aligned}\mathcal{F}_i^{ch} = \{\langle s, ((s' \ominus s) + 1)\rangle \quad | \quad & s, s' \in \mathcal{S}_i^{ch} \wedge \\ & (s \ominus 1) \notin \mathcal{S}_i^{ch} \wedge \\ & (s' \oplus 1) \notin \mathcal{S}_i^{ch} \wedge \\ & \forall s'',\ (s' \ominus s) \geq (s'' \ominus s), s'' \in \mathcal{S}_i^{ch}\}\end{aligned} \tag{1-4}$$

We define an "empty" set $\mathcal{E}_i^{ch}$ containing the contiguous blocks of slots not allocated to each channel of a connection c_i:

$$\begin{aligned}\mathcal{E}_i^{ch} = \{\langle s, ((s' \ominus s) + 1)\rangle \quad | \quad & s, s' \notin \mathcal{S}_i^{ch} \wedge \\ & (s \ominus 1) \in \mathcal{S}_i^{ch} \wedge \\ & (s' \oplus 1) \in \mathcal{S}_i^{ch} \wedge \\ & \forall s''\ (s' \ominus s) \geq (s'' \ominus s), s'' \notin \mathcal{S}_i^{ch}\}\end{aligned} \tag{1-5}$$

Using $\mathcal{F}_i^{ch}$, we can also define $\mathcal{H}_i^{ch}$ as the set of slots which contain headers in the case there is data sent at full rate. As in Æthereal packets correspond to blocks of slots (Rădulescu et al., 2005), at each block of slots a header is introduced:

$$\mathcal{H}_i^{ch} = \{s_h \in \mathcal{S}_i^{ch} \quad | \quad \langle s_h, \lambda\rangle \in \mathcal{F}_i^{ch}\} \tag{1-6}$$

3. THROUGHPUT ANALYSIS

The available throughput for a connection depends on the slot allocation for sending data and flow control information and the size of buffers in NIs. For this analysis, we assume that slot allocation for a connection is given. To fully utilize the available bandwidth, buffers must be dimensioned based on the analysis of Section 4 and there must be enough bandwidth available for sending flow control information.

To derive available throughput, for a given connection c_i, network specific overheads (e.g., packet header) and transaction specific overheads (e.g., command and address) need to be subtracted from the raw bandwidth based on the slot allocation, $\mathcal{S}_i$.

We first consider the case of a producer connected to a consumer through a channel *ch* with a slot allocation of $\mathcal{S}_i^{ch}$. We call $N_{h,i}^{ch}$ the number of headers introduced in one slot table iteration of a channel *ch*.

$$N_{h,i}^{ch} = |\mathcal{H}_i^{ch}| \tag{1-7}$$

Where $\mathcal{H}_i^{ch}$ denotes a set of allocated slots, where a header will be sent, for the channel *ch*.

We define $W_{r,i}^{ch}$ as the total number of words reserved for the channel in a slot table iteration. These words are divided in two categories, the first one is used to carry headers $W_{h,i}^{ch}$, and the second one is used to carry payload data[4] $W_{p,i}^{ch}$ (see Figure 1-3).

$$W_{r,i}^{ch} = |\mathcal{S}_i^{ch}| \times L_s \tag{1-8}$$

$$W_{h,i}^{ch} = N_{h,i}^{ch} \times L_h \tag{1-9}$$

$$W_{p,i}^{ch} = W_{r,i}^{ch} - W_{h,i}^{ch} \tag{1-10}$$

Based on the bandwidth associated to a reserved word B_w, we define raw bandwidth $B_{r,i}^{ch}$, header bandwidth $B_{h,i}^{ch}$, and payload bandwidth $B_{p,i}^{ch}$ for a given channel *ch*.

$$B_{r,i}^{ch} = W_{r,i}^{ch} \times B_w \tag{1-11}$$

$$B_{h,i}^{ch} = W_{h,i}^{ch} \times B_w \tag{1-12}$$

$$B_{p,i}^{ch} = W_{p,i}^{ch} \times B_w \tag{1-13}$$

The maximum flow control value that can be sent in one packet header is denoted by M_{FC} and the rate to send flow control words by $\Theta_{FC,i}^{ch} = N_{h,i}^{ch} \times B_w$.

[4]Recall that in the payload data, we include command, address, and application data (see Section 2.1 for explanation).

For a correct operation, the amount of credits sent (using flow control headers) on opposite channel (e.g. R) must be greater than or equal to the amount of data consumed by the consumer on a channel *ch* (e.g. F).

$$\Theta^{F}_{FC,i} \times M_{FC} \geq R^{R,M}_{i} \quad (1\text{-}14)$$

$$\Theta^{R}_{FC,i} \times M_{FC} \geq R^{F,S}_{i} \quad (1\text{-}15)$$

where $R^{R,M}_{i}$ and $R^{F,S}_{i}$ are the data rate of the consumer on the reverse channel and forward channel, respectively.

In the following sections, we derive exact formulas for throughput of write-only, read-only and read-write connections. Furthermore, we also provide formulas to check whether the given slot allocation meets the flow control requirements or not.

3.1 Throughput for write-only connections

For a write connection c_i, all data (including command, address, write data) are sent on the forward channel and reverse channel is used only for sending flow control data. The available data throughput for write data is derived from Equation (1-13):

$$B^{F}_{p,i} = W^{F}_{p,i} \times B_w \quad (1\text{-}16)$$

The available data $R^{Wr}_{DATA,i}$ and command $R^{Wr}_{CMD,i}$ throughput for write data (excluding packet overhead), for a given command to data ratio γ^{Wr}_{i}, is:

$$R^{Wr}_{DATA,i} = \frac{B^{F}_{p,i}}{(1+\gamma^{Wr}_{i})} \quad (1\text{-}17)$$

$$R^{Wr}_{CMD,i} = \gamma^{Wr}_{i} \times R^{Wr}_{DATA,i} \quad (1\text{-}18)$$

The specified data rates $R^{Wr}_{IP,i}$ are met when $R^{Wr}_{DATA,i} \geq R^{Wr}_{IP,i}$.

On the reverse channel, no data is sent for write transactions but flow control information (i.e., amount of data removed from buffer of the NI of consumer) needs to be sent. For a correct operation, the amount of credits sent (using a flow control header in a packet) must be greater than or equal to the amount of data consumed by the consumer. By substituting values for channel type (i.e., R) and the specified data rates in Equation (1-15), the condition is:

$$\Theta^{R}_{FC,i} \times M_{FC} \geq (1+\gamma^{Wr}_{i}) \times R^{Wr}_{IP,i} \quad (1\text{-}19)$$

3.2 Throughput for read-only connections

For a read connection c_i, commands and flow control information for read data is sent on the forward channel and on the reverse channel read data and flow control information for commands is sent.

The available data throughput for sending commands (excluding packet header overhead) is derived from Equation (1-13). As we only send commands through the forward channel, the available data throughput $B^F_{p,i}$ is fully used for commands $R^{Rd}_{CMD,i}$

$$R^{Rd}_{CMD,i} = B^F_{p,i} = W^F_{p,i} \times B_w \tag{1-20}$$

The available data throughput for sending read data (excluding packet header overhead) is derived from Equation (1-13). As we only send data through reverse channel, the available data throughput $B^R_{p,i}$ is fully used for sending data $R^{Rd}_{DATA,i}$

$$R^{Rd}_{DATA,i} = B^R_{p,i} = W^R_{p,i} \times B_w \tag{1-21}$$

For a command to data ratio of γ^{Rd}_i, the conditions when the specified data $R^{Rd}_{IP,i}$ and command rates are met are:

$$R^{Wr}_{DATA,i} \geq R^{Rd}_{IP,i} \tag{1-22}$$

$$R^{Wr}_{CMD,i} \geq \gamma^{Rd}_i \times R^{Rd}_{IP,i} \tag{1-23}$$

For a correct operation, the amount of credits sent (using flow control headers) for the forward (reverse) channel must be greater than or equal to the amount of data (command) sent in the reverse (forward) channel. By substituting the values for the data rates in Equations (1-14) and (1-15), the conditions are:

$$\Theta^F_{FC,i} \times M_{FC} \geq R^{Rd}_{IP,i} \tag{1-24}$$

$$\Theta^R_{FC,i} \times M_{FC} \geq \gamma^{Rd}_i \times R^{Rd}_{IP,i} \tag{1-25}$$

3.3 Throughput for read-write connections

For the forward path, which is used for read commands, write commands, write data, and end-to-end flow control for read data, the data rate is defined as:

$$R^{F,RdWr}_i = R^{Rd}_{CMD,i} + R^{Wr}_{CMD,i} + R^{Wr}_{DATA,i} \tag{1-26}$$

For the reverse path, read data and flow control for forward data are sent. The reverse data rate is defined as:

$$R^{R,RdWr}_i = R^{Rd}_{DATA,i} \tag{1-27}$$

For a read-write connection, the conditions when the specified data rates are met are:

$$R^{F,RdWr}_i \geq (1 + \gamma^{Wr}_i) \times R^{Wr}_{IP,i} + \gamma^{Rd}_i \times R^{Rd}_{IP,i} \tag{1-28}$$

$$R^{R,RdWr}_i \geq R^{Rd}_{IP,i} \tag{1-29}$$

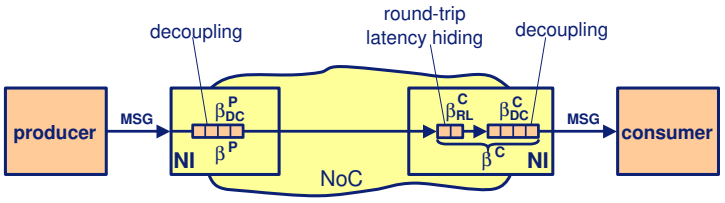


Figure 1-7. Buffers are logically split in (1) decoupling buffers β^P_{DC} and β^C_{DC}, and (2) flow-control round-trip latency hiding buffers β^C_{RL}.

For a correct operation, the amount of credits sent (using flow control headers) for the forward (reverse) channel must be greater than or equal to the amount of data (command and/or data) sent in the reverse (forward) channel. By substituting the values for the data rates in Equations (1-14) and (1-15) the conditions are:

$$\Theta^F_{FC,i} \times M_{FC} \geq R^{Rd}_{IP,i} \quad (1\text{-}30)$$

$$\Theta^R_{FC,i} \times M_{FC} \geq (1 + \gamma^{Wr}_i) \times R^{Wr}_{IP,i} + \gamma^{Rd}_i \times R^{Rd}_{IP,i} \quad (1\text{-}31)$$

4. BUFFER SIZE ANALYSIS

As shown in Figure 1-1, an Æthereal connection consists of two channels: one forward channel, and one reverse channel. Each channel has one buffer at the producer side (forward buffer at the master side, and reverse buffer at the slave side), and one buffer at the consumer side (forward buffer at the slave side, and reverse buffer at the master side). Both buffers at the producer and consumer side are used to decouple the IP blocks from the NIs, namely to hide the differences in operating frequency and communication pattern of the IP blocks and NI. Moreover, the consumer-side buffer is also used to hide the round-trip latency of reporting the flow-control credits.

For analysis purposes, we split the buffer at the consumer in two: one part for flow-control round-trip latency hiding (β^C_{RL}), and the other for decoupling (β^C_{DC}) (see Figure 1-7). In an actual implementation, for efficiency reasons, these two parts should be merged into a single buffer $\beta^C = \beta^C_{RL} + \beta^C_{DC}$. In the following two sections we describe how to compute the worst-case size for these two kinds of buffers.

4.1 Decoupling Buffers

The decoupling-buffer size computation relies on the fact that modules exchanging data exhibit a particular behavior. In our context, consisting of real-time audio/video applications, it is safe to assume that modules transfer data periodically, with an upper bound on the amount of data transferred per period.

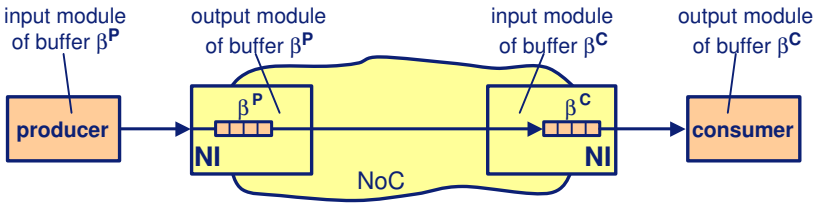


Figure 1-8. For each buffer, there are an input module (filling the buffer) and an output module (emptying the buffer).

This is, the traffic generated or consumed by a module is characterized by the following parameters (see Section 2.2 and Figure 1-6):

Period (T) is the minimum period in which a constant amount of data is sent. For a connection c_i of an IP module , T corresponds to $T^{Rd}_{IP,i}$, and/or $T^{Wr}_{IP,i}$ for read and write transactions, respectively. For an Æthereal NoC with arbitrary slot allocation on a connection c_i's channel, the period T is equal to the duration of a complete slot table rotation $|\mathcal{S}| \times T_s$. When slots are allocated equidistantly in blocks of k contiguous slots, T is taken $(|\mathcal{S}| \times T_s \times k)/|\mathcal{S}^{ch}_i|$. As shown further in this section and Section 5, a smaller period T implies smaller buffers and shorter worst-case latencies.

Data amount (D) is the upper bound on the transferred data in the given period. For the IP modules, D corresponds to the messages, and is equal to $L^{Wr}_{CMD,i} + L^{Wr}_{DATA,i}$ for write requests, $L^{Rd}_{CMD,i}$ and $L^{Rd}_{DATA,i}$ for read requests and read responses sent on a connection c_i, respectively. For a NoC with arbitrary slot allocation on a connection c_i's channel, D is equal to the number of payload words $W^{ch}_{p,i}$ transferred in a complete slot rotation. In case of equidistantly allocated blocks of k contiguous slots, $D = W^{ch}_{p,i} \times k/|\mathcal{S}^{ch}_i|$.

Regular or irregular to specify if an IP module transfers data in the same or in an arbitrary position within the interval T, respectively. IP modules can be either regular or irregular. An Æthereal NoC, however, always transfers data in the reserved slots, which do not change from a slot table rotation to another. For this reason, the NoC is always periodic over a complete slot table rotation period $|\mathcal{S}| \times T_s$. For connection c_i's channel with equidistantly allocated blocks of slots, there may be jitter in the slot allocation, and, if there is jitter, NoC will be periodic irregular over $|\mathcal{S}| \times T_s \times k/|\mathcal{S}^{ch}_i|$.

We use the same method of computing the buffer size for all the decoupling buffers, at producer and consumer sides, and for all connections. To simplify

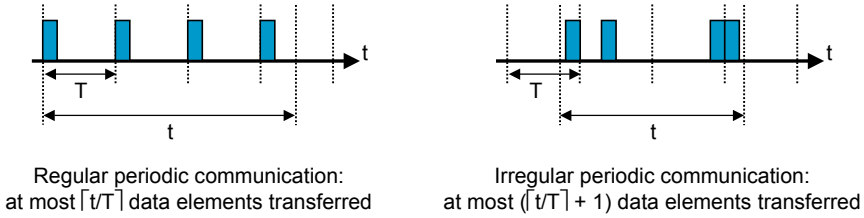


Figure 1-9. Upper bounds for periodic data transfer.

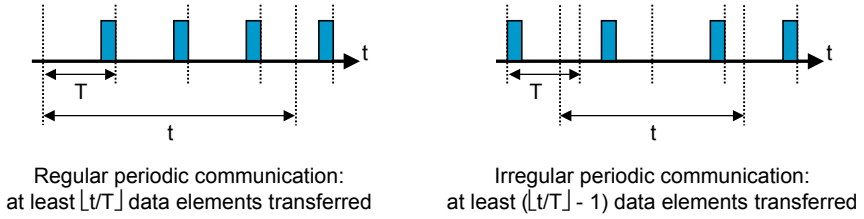


Figure 1-10. Lower bounds for periodic data transfer.

the discussion, we call the module attached to the input of the buffer the input module (producer, and NI at the consumer side), and the module attached to the output of the buffer the output module (NI at the producer side, and consumer), as shown in Figure 1-8.

Given an input module and an output module, let their periods be T_I and T_O, and the data amount per period be D_I and D_O, respectively. The maximum buffer size required between an input module M_I and an output module M_O is given by the maximum difference between the data produced by M_I and the data consumed by M_O over any time interval.

To compute this maximum difference for an arbitrary time interval of duration t, we must consider the worst-case for data production and consumption, respectively. We consider the two cases presented in Section 2.2: regular and irregular.

The worst-case data to be buffered is when the amount of produced data over an arbitrary time interval t is maximized. As shown in Figure 1-9, for the regular case, the amount of produced data is bounded by the minimum number of periods T_I that covers the time interval considered t:

$$\phi_I^R(t) \leq \left\lceil \frac{t}{T_I} \right\rceil \times D_I \tag{1-32}$$

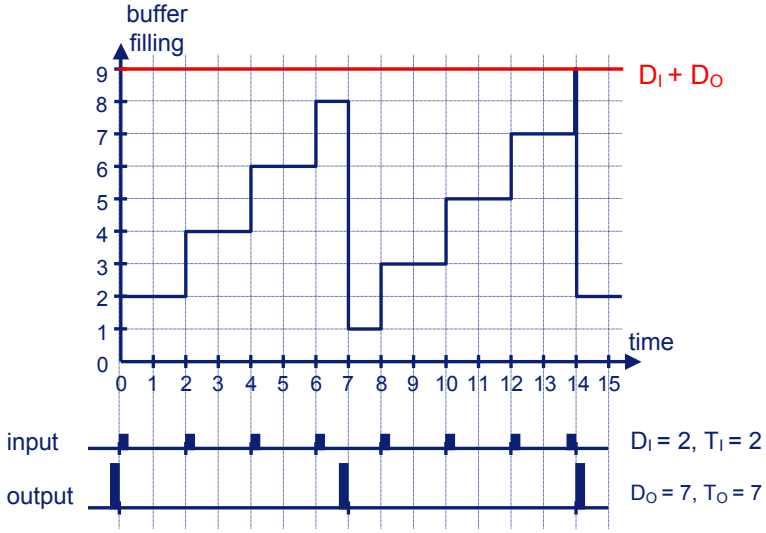


Figure 1-11. Example buffer filling for almost regular production and consumption of data.

For the irregular case, one more data item needs to be added because of the arbitrary position that the data may take inside the period:

$$\phi_I^I(t) \leq \left(\left\lceil \frac{t}{T_I} \right\rceil + 1 \right) \times D_I \tag{1-33}$$

Similarly, the worst-case data consumption occurs when the amount of consumed data over a time interval t is minimized. As shown in Figure 1-10, for the regular case, the amount of data is minimized by the number of periods T that fit in the time interval t:

$$\phi_O^R(t) \geq \left\lfloor \frac{t}{T_O} \right\rfloor \times D_O \tag{1-34}$$

For the irregular case, one more data item needs to be subtracted because of the arbitrary position that the data may take inside the period:

$$\phi_O^I(t) \geq \left(\left\lfloor \frac{t}{T_O} \right\rfloor - 1 \right) \times D_O \tag{1-35}$$

The buffer filling is a function of time t, and is less than or equal to the difference of the amount of data sent by M_I (i.e., $\phi_I(t)$), and the amount of data that can be received by M_O (i.e., $\phi_O(t)$) (see Figure 1-11 for an example). In the case M_O can consume more data than the M_I can produce, the buffer filling

is zero:

$$\phi(t) \leq \max\{\phi_I(t) - \phi_O(t), 0\} \tag{1-36}$$

For both regular production and regular consumption of data, the worst-case buffer size requirements are:

$$\phi^R R(t) \leq \max\{\phi_I^R(t) - \phi_O^R(t), 0\} \tag{1-37}$$

From Equations (1-32), (1-33), (1-34) and (1-35), and assuming the input rate is at most the output rate of the considered buffer ($D_I/T_I \leq D_O/T_O$):

$$\begin{aligned} \phi_I^R(t) - \phi_O^R(t) &\leq \left\lceil \frac{t}{T_I} \right\rceil \times D_I - \left\lfloor \frac{t}{T_O} \right\rfloor \times D_O \\ &\leq \left(\frac{t}{T_I} + 1 \right) \times D_I - \left(\frac{t}{T_O} - 1 \right) \times D_O \\ &\leq D_I + D_O + t \times \left(\frac{D_I}{T_I} - \frac{D_O}{T_O} \right) \\ &\leq D_I + D_O \end{aligned} \tag{1-38}$$

From (1-37) and (1-38):

$$\phi^{RR}(t) \leq D_I + D_O \tag{1-39}$$

In other words, the maximum buffer filling for regular periodic input and output modules is equal to the sum of the data produced D_I and consumed D_O in the periods T_I and T_O, respectively.

In the case of a NoC-based system, the buffers always reside in between an IP module (master or slave) and the NoC. The Æthereal NoC provides guaranteed-bandwidth data transfers using slot reservations in slot tables. Consequently, the NoC behavior is always regular and periodic.

In this chapter, we address the cases in which the IP module behavior is periodic and can be either regular or irregular. As the NoC is always regular and periodic, Equation (1-39) covers the case in which the IP module is periodic. For an irregular consumer, the worst-case buffer size requirements are given by:

$$\phi^{RI}(t) \leq D_I + 2 \times D_O \tag{1-40}$$

and, for an irregular IP module producer, the worst-case buffer size requirements are given by:

$$\phi^{IR}(t) \leq 2 \times D_I + D_O \tag{1-41}$$

These two equations are derived similarly to Equation (1-39).

For a connection c_i for which the throughput is guaranteed, there are a number of slots reserved in the slot table for the forward and reverse channels. As explained in Section 3, the bandwidth reserved with these slots is split in bandwidth for headers ($W_{h,i}$) and bandwidth for payload ($W_{p,i}$). In the interval given by the slot table period ($|\mathcal{S}| \times L_s/B_L$), the data is produced/consumed regularly. The NoC acts as an output module at the master side for the forward channel, and at the slave side for the reverse channel, and as an input module at the master side for the reverse channel and at the slave side for the forward channel.

Let us first consider the case when the master and slaves produce and consume data regularly. For a non-acknowledged write-only connection c_i, the decoupling buffer sizes (measured in words of L_w bits) for forward master and slave, and reverse slave and master are given by:

$$\beta^{F,M}_{DC,i} = (L^{Wr}_{DATA,i} + L^{Wr}_{CMD,i}) + W^{F}_{p,i} \quad (1\text{-}42)$$

$$\beta^{F,S}_{DC,i} = W^{F}_{p,i} + (L^{Wr}_{DATA,i} + L^{Wr}_{CMD,i}) \quad (1\text{-}43)$$

$$\beta^{R,S}_{DC,i} = 0 \quad (1\text{-}44)$$

$$\beta^{R,M}_{DC,i} = 0 \quad (1\text{-}45)$$

respectively[5].

For a read-only connection, buffer sizes are given by:

$$\beta^{F,M}_{DC,i} = L^{Rd}_{CMD,i} + W^{F}_{p,i} \quad (1\text{-}46)$$

$$\beta^{F,S}_{DC,i} = W^{F}_{p,i} + L^{Rd}_{CMD,i} \quad (1\text{-}47)$$

$$\beta^{R,S}_{DC,i} = L^{Rd}_{DATA,i} + W^{R}_{p,i} \quad (1\text{-}48)$$

$$\beta^{R,M}_{DC,i} = W^{R}_{p,i} + L^{Rd}_{DATA,i} \quad (1\text{-}49)$$

For a read-write connection, buffer sizes are a sum of the buffers for the write-only and read-only cases:

$$\beta^{F,M}_{DC,i} = (L^{Wr}_{DATA,i} + L^{Wr}_{CMD,i} + L^{Rd}_{CMD,i}) + W^{F}_{p,i} \quad (1\text{-}50)$$

$$\beta^{F,S}_{DC,i} = W^{F}_{p,i} + (L^{Wr}_{DATA,i} + L^{Wr}_{CMD,i} + L^{Rd}_{CMD,i}) \quad (1\text{-}51)$$

$$\beta^{R,S}_{DC,i} = L^{Rd}_{DATA,i} + W^{R}_{p,i} \quad (1\text{-}52)$$

$$\beta^{R,M}_{DC,i} = W^{R}_{p,i} + L^{Rd}_{DATA,i} \quad (1\text{-}53)$$

For the case the master and/or slave produce or consume data irregularly in their periods, the buffer requirements at the master/slave side double (see

[5]There is no buffer needed for the reverse channel, because, in the write-only case, there is no data being sent in the reverse channel. Slots must still be reserved for the transportation of credits, however, they are not buffered in the NI, but processed directly.

Equations (1-40) and (1-41). Let us consider, for example, the case in which both master and slave produce and consume data irregularly. For a write-only connection, buffer sizes are given by:

$$\beta^{F,M}_{DC,i} = 2 \times (L^{Wr}_{DATA,i} + L^{Wr}_{CMD,i}) + W^{F}_{p,i} \quad (1\text{-}54)$$

$$\beta^{F,S}_{DC,i} = W^{F}_{p,i} + 2 \times (L^{Wr}_{DATA,i} + L^{Wr}_{CMD,i}) \quad (1\text{-}55)$$

$$\beta^{R,S}_{DC,i} = 0 \quad (1\text{-}56)$$

$$\beta^{R,M}_{DC,i} = 0 \quad (1\text{-}57)$$

For a read-only connection, buffer sizes are given by:

$$\beta^{F,M}_{DC,i} = 2 \times L^{Rd}_{CMD,i} + W^{F}_{p,i} \quad (1\text{-}58)$$

$$\beta^{F,S}_{DC,i} = W^{F}_{p,i} + 2 \times L^{Rd}_{CMD,i} \quad (1\text{-}59)$$

$$\beta^{R,S}_{DC,i} = 2 \times L^{Rd}_{DATA,i} + W^{R}_{p,i} \quad (1\text{-}60)$$

$$\beta^{R,M}_{DC,i} = W^{R}_{p,i} + 2 \times L^{Rd}_{DATA,i} \quad (1\text{-}61)$$

For a read-write connection, buffer sizes are again a sum up of the buffers for the write-only and read-only cases:

$$\beta^{F,M}_{DC,i} = 2 \times (L^{Wr}_{DATA,i} + L^{Wr}_{DATA,i} + L^{Rd}_{CMD,i}) + W^{F}_{p,i} \quad (1\text{-}62)$$

$$\beta^{F,S}_{DC,i} = W^{F}_{p,i} + 2 \times (L^{Wr}_{DATA,i} + L^{Wr}_{CMD,i} + L^{Rd}_{CMD,i}) \quad (1\text{-}63)$$

$$\beta^{R,S}_{DC,i} = 2 \times L^{Rd}_{DATA,i} + W^{R}_{p,i} \quad (1\text{-}64)$$

$$\beta^{R,M}_{DC,i} = W^{R}_{p,i} + 2 \times L^{Rd}_{IP,i} \quad (1\text{-}65)$$

4.2 Round-Trip Latency-Hiding Buffers

The round-trip latency-hiding buffer is located in the NI at the consumer side (see Figure 1-7). It is needed to compensate for the time from which a producer NI reduces its credits when sending packets until it receives back the credits from the consumer NI. If this buffer is too small, the producer NI can run out of credits and temporarily stall its transmission of packets, and, therefore, does not meet its bandwidth requirements.

To compute the minimum size of round-trip latency-hiding buffer, we need to compute the worst-case latency necessary for the credits to be reported back to the NI from where the data is sent. The buffer size must be larger than or equal to the maximum amount of credits that can be consumed when sending data without being reported back to the sender NI.

For a connection c_i, the round-trip latencies $T^{M}_{RL,i}$ and $T^{S}_{RL,i}$ from when the data is sent by the producer NI (at master and slave, respectively) until the first

credits reach back the producing NI is given by:

$$T^{M}_{RL,i} = T^{F,T}_{L,i} + T^{R,T}_{L,i} + \max_{\langle s,\lambda\rangle \in \mathcal{E}^{R}_{i}} \lambda \tag{1-66}$$

$$T^{S}_{RL,i} = T^{R,T}_{L,i} + T^{F,T}_{L,i} + \max_{\langle s,\lambda\rangle \in \mathcal{E}^{F}_{i}} \lambda \tag{1-67}$$

where $T^{F,T}_{L,i}$ and $T^{R,T}_{L,i}$ refer to the amount of time to transport data (given by the number of hops, i.e., the number of links traversed by the packets of a channel from the producer to the consumer) on the forward and reverse channels, respectively, and $\mathcal{E}^{F}_{i}$ and $\mathcal{E}^{R}_{i}$ refer to the slots not reserved (to send flow control) for the forward and reverse channels, respectively. The first two terms represent the network latency in both directions, and the third term represents the time flow control has to wait in the NI until it can be sent. This formula represents the latency until the first credits are returned. Recall that the maximum amount of credits that can be transported in a packet is bounded to M_{FC}.

To address the case in which the amount of data transferred in the $T^{F}_{RL,i}$ and $T^{R}_{RL,i}$ intervals is larger than M_{FC}, we compute the amount of flow control that can accumulate in any time interval spanning over $0 < \delta \leq |\mathcal{S}|$ slots[6] at the destination NI without being reported back with credits (in case it is consumed)[7]:

$$\begin{aligned} \phi^{F,acc}_{i}(\delta) \;=\; & L_s \times \max_{\langle s_0,\lambda\rangle \in \mathcal{F}^{F}_{i}} \left\{ \begin{array}{l} |\,\{s \in \mathcal{S}^{F}_{i} \mid 0 < s \ominus s_0 \leq \delta\}\,| - \\ |\,\{s \in \mathcal{H}^{R}_{i} \mid 0 < s_0 \ominus s \leq \delta\}\,| \end{array} \right\} - \\ & M_{FC} \times \min_{s_0 \in \mathcal{H}^{R}_{i}} |\,\{s \in \mathcal{H}^{R}_{i} \mid 0 < s_0 \ominus s \leq \delta\}\,| \end{aligned} \tag{1-68}$$

$$\begin{aligned} \phi^{R,acc}_{i}(\delta) \;=\; & L_s \times \max_{\langle s_0,\lambda\rangle \in \mathcal{F}^{R}_{i}} \left\{ \begin{array}{l} |\,\{s \in \mathcal{S}^{R}_{i} \mid 0 < s \ominus s_0 \leq \delta\}\,| - \\ |\,\{s \in \mathcal{H}^{F}_{i} \mid 0 < s_0 \ominus s \leq \delta\}\,| \end{array} \right\} - \\ & M_{FC} \times \min_{s_0 \in \mathcal{H}^{F}_{i}} |\,\{s \in \mathcal{H}^{F}_{i} \mid 0 < s_0 \ominus s \leq \delta\}\,| \end{aligned} \tag{1-69}$$

where the first terms represent the maximum amount of data that can be transferred in the interval δ (excluding headers), and the second terms represent the minimum amount of credits that can be transported back to the sender in the same interval δ.

[6] This interval is practically a sliding window of δ slots: $[t, t + \delta \times L_s/B_L]$, where t is any time aligned to the slot boundary.

[7] For a correct slot allocation, for each channel, the maximum amount of credits that can be sent by the consumer in a slot table rotation ($M_{FC} \times |\mathcal{H}^{R}_{i}|$ and $M_{FC} \times |\mathcal{H}^{F}_{i}|$) must be larger than or equal to the amount of data produced by the producer NI in a slot table rotation ($W^{F}_{p,i}$ and $W^{R}_{p,i}$, respectively).

Let $\delta_i^{F,acc}$ and $\delta_i^{R,acc}$ be the largest interval less than a slot table rotation (i.e., $0 \leq \delta_i^{F,acc} \leq |\mathcal{S}_i^F|$ and $0 \leq \delta_i^{R,acc} \leq |\mathcal{S}_i^R|$) for which $\beta_i^{F,acc}$ and $\beta_i^{R,acc}$ are maximized, respectively. Then

$$T_{RL,i}^{M} = T_{L,i}^{F,T} + T_{L,i}^{R,T} + \delta_i^{F,acc} \tag{1-70}$$

$$T_{RL,i}^{S} = T_{L,i}^{R,T} + T_{L,i}^{F,T} + \delta_i^{R,acc} \tag{1-71}$$

represent the minimum time intervals in which the worst-case amount of data (the maximum amount of data) matches with the worst-case amount of credits (the minimum amount of credits).

The buffering to hide the round-trip latency is then:

$$\beta_{RL,i}^{F,S} = \left\lfloor \frac{T_{RL,i}^{M}}{|\mathcal{S}_i^F|} \right\rfloor \times W_{p,i}^{F} + \beta_i^{F,acc}(\delta_i^{F,acc}) \tag{1-72}$$

$$\beta_{RL,i}^{R,M} = \left\lfloor \frac{T_{RL,i}^{S}}{|\mathcal{S}_i^R|} \right\rfloor \times W_{p,i}^{R} + \beta_i^{R,acc}(\delta_i^{R,acc}) \tag{1-73}$$

The first terms represent the buffering needed to accommodate the data sent in complete table rotations (when latency is larger then a complete slot rotation), and the second terms represent the buffering needed to accommodate the maximum amount of data that can be sent in the fraction of the latency overlapping with a partial slot table rotation.

4.3 Total Buffer Sizes

As mentioned earlier, the buffer sizes in the network interfaces are given by several components: decoupling buffers and credit round-trip latency-hiding buffers. At the producer sides ($\beta_i^{F,M}$ and $\beta_i^{R,S}$), the buffers are for decoupling buffer only, while at the consumer sides ($\beta_i^{F,S}$ and $\beta_i^{R,M}$), the buffers are used for both decoupling and credit round-trip latency-hiding. As a result, the buffer sizes are given by:

$$\beta_i^{F,M} = \beta_{DC,i}^{F,M} \tag{1-74}$$

$$\beta_i^{F,S} = \beta_{DC,i}^{F,S} + \beta_{RL,i}^{F,S} \tag{1-75}$$

$$\beta_i^{R,S} = \beta_{DC,i}^{R,S} \tag{1-76}$$

$$\beta_i^{R,M} = \beta_{DC,i}^{R,M} + \beta_{RL,i}^{R,M} \tag{1-77}$$

5. LATENCY ANALYSIS

The latency of a connection c_i, $T_{L,i}$ is composed of the latency of forward channel $T_{L,i}^{F}$, reverse channel $T_{L,i}^{R}$ and IP latency $T_{L,i}^{IP}$ (see Figure 1-12). For the

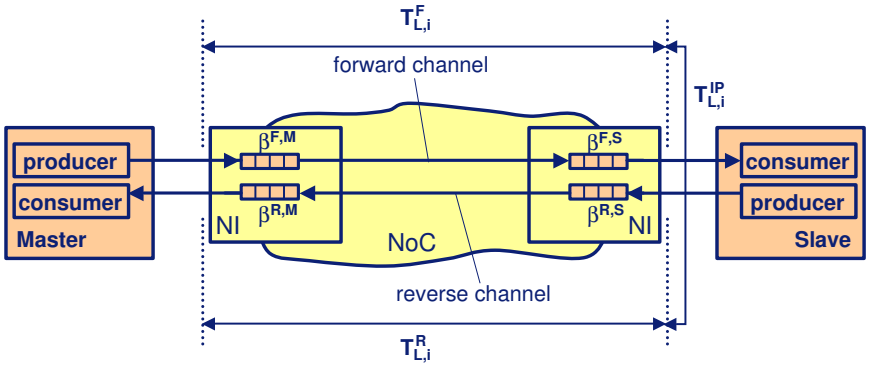


Figure 1-12. Latency of a connection.

sake of simplicity, we calculate latency in terms of slot cycles, so, in all the formulas that we derive for latency a multiplication factor of T_s must be used when converting them in terms of seconds.

We first address the general case with one producer and one consumer connected through a channel (see Figure 1-13). The latency for a channel, $T^{ch}_{L,i}$, is measured from the time a word data is accepted by the NI at producer side until the same word is accepted by the consumer. Note that the latency of a channel depends on the behavior of the consumer. We consider two cases for consumer:

Unoccupied consumer, when a consumer is ready to consume data as soon as data are offered by the NI at the consumer side.

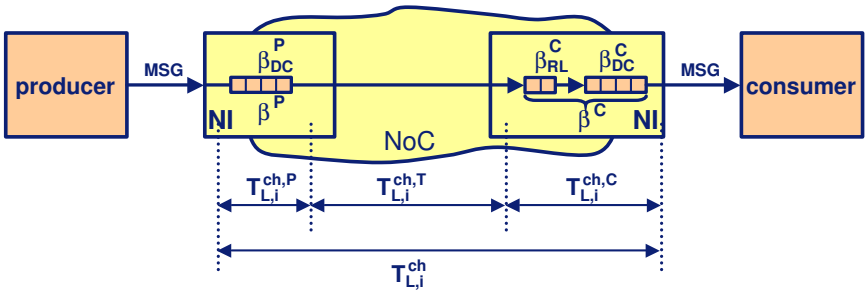


Figure 1-13. Latency of a channel.

Occupied consumer, when a consumer delays consumption of data due to, for example, sharing of the same port by many connections or when the consumer cannot, for some reason, accept the data.

In the previous section buffering values for a channel between a producer and a consumer are derived such that full utilization of bandwidth can be achieved, i.e., the producer never stalls because of lack of credits. Buffers are introduced at the producer NI (called producer buffer) as well as at consumer NI (called consumer buffer), these buffers contribute to the latency (see Figure 1-13). We split total latency for a channel $T^{ch}_{L,i}$ in three components, $T^{ch,P}_{L,i}$ the time required for the data in front of the current word to leave the producer NI, $T^{ch,T}_{L,i}$ the latency to transport a word from producer NI to the consumer NI (given by the number of hops), and $T^{ch,C}_{L,i}$ the time required for the data in front of the current word to leave the consumer NI.

$$T^{ch}_{L,i} = T^{ch,P}_{L,i} + T^{ch,T}_{L,i} + T^{ch,C}_{L,i} \tag{1-78}$$

The latency $T^{ch,P}_{L,i}$ depends on the slot allocation. For a given slot allocation the amount of data that can be removed from the producer buffer during one iteration of slot table is denoted by $W^{ch}_{p,i}$ (see Equation (1-10) on page 10). In the worst-case, the buffer is full, meaning that as many words as the buffer size $\beta^{ch,P}_i$ must be sent. We divide the data of the buffer into two parts, first part is an integer multiple n of $W^{ch}_{p,i}$ and the second is remainder r.

$$\beta^{ch,P}_i = n \times W^{ch}_{p,i} + r \tag{1-79}$$

The time $T^{ch,P}_{L_I,i}$ to send the first part of the buffer is n iterations of the slot table. However, to derive time $T^{ch,P}_{L_R,i}$ to send the remainder data r, first a function $W^{ch}_{p_{min},i}(d)$ is defined to calculate the minimum number of payload words that can be sent for a given window size d. The payload words are calculated by subtracting the number of words used for sending headers from the total number of words that can be sent for the allocated slots for the channel in the given window.

$$W^{ch}_{p_{min},i}(\delta) = \min_{s \in \mathcal{S}} \left\{ \begin{array}{l} L_s \times |\{s' \in \mathcal{S}^{ch}_i \mid s' \ominus s < \delta\}| - \\ |\{s' \in \mathcal{H}^{ch}_i \mid s' \ominus s < \delta\}| \end{array} \right\} \tag{1-80}$$

Figure 1-14 shows the number of payload words sent for the given values of δ in an example slot allocation. For a window size of 4 (i.e., $\delta = 4$) two extreme possibilities for payload p=2 and p=7 and the minimum value for payload is 2.

The set $\{\delta \in \mathbb{N} \mid W^{ch}_{p_{min},i}(\delta) \geq r\}$ defines the values of window in which at least the remaining data r can be sent. The minimum value in this set is not the

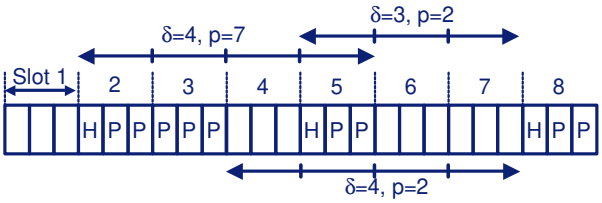


Figure 1-14. The amount of payload data p sent for a given window d for an example slot table size of 8.

worst-case delay value as it does not include all empty slots around allocated slots. For example, to send a payload of 2 (i.e., $p = 2$) the set of window is $\{3, 4, 5, 6, 7, 8\}$ (derived from the condition above) and the minimum of the set is 3 that is not the worst-case value rather the worst-case value is 4 (see Figure 1-14). So, we define an upper bound ($W^{ch}_{p_{min},i}(\delta) < (r + L_s)$) for the set such that it only allows sending the minimum payload data of remainder plus the number of words in a slot. As the minimum payload words are discrete values with the maximum step size of L_s, this upper bound also ensures that the set of window is not empty. The maximum value of the bounded set gives the worst-case delay to transfer the remaining data.

$$T^{ch,P}_{L_R,i} = \max\left\{\delta \in \mathbb{N} \mid r \leq W^{ch}_{p_{min},i}(\delta) < (r + L_s)\right\} \tag{1-81}$$

The total latency $T^{ch,P}_{L,i}$ is the sum of the integer part $T^{ch,P}_{L_I,i}$ and the remainder part $T^{ch,P}_{L_R,i}$.

$$\begin{aligned} T^{ch,P}_{L,i} &= T^{ch,P}_{L_I,i} + T^{ch,P}_{L_R,i} && (1\text{-}82)\\ &= n \times |\mathcal{S}| + \max\left\{\delta \in \mathbb{N} \mid r \leq W^{ch}_{p_{min},i}(\delta) < (r + L_s)\right\} && (1\text{-}83) \end{aligned}$$

where

$$\begin{aligned} n &= \left\lfloor \frac{\beta^{ch,P}_i}{W^{ch}_{p,i}} \right\rfloor \\ r &= \beta^{ch,P}_i \% W^{ch}_{p,i} \end{aligned}$$

$$T^{ch,T}_{L,i} = \text{Number of hops between producer NI and consumer NI.} \tag{1-84}$$

The latency for the consumer buffer, $T^{ch,C}_{L,i}$, depends on the consumption pattern of the consumer. We consider the case of an unoccupied consumer which is ready to consume data as soon as it is offered by the consumer NI. In this case, the consumer buffer would always remain empty as consumer is aggressively removing data. Hence, latency caused by the consumer buffer does not have any contributions to the total latency, (i.e., $T^{ch,C}_{L,i} = 0$).

In the case of an occupied consumer, we assume that in the worst-case a consumer consumes ($R_{IP,i} \times T^C_{IP,i}$) data words in a period $T^C_{IP,i}$ (see Section 2). $T^{ch,C}_{L,i}$ is given by the number of consumer periods needed to empty a full buffer. We convert it into the number of slot periods by dividing it by the time required to traverse a slot T_s.

$$T^{ch,C}_{L,i} = \left\lceil \frac{\beta^{ch,C}_i}{R^C_{IP,i} \times T^C_{IP,i}} \right\rceil \times \frac{T^C_{IP,i}}{T_s} \tag{1-85}$$

We derive the latencies for a given connection (read and/or write) from the latencies for a given channel in the following sections.

5.1 Latency for write-only connections

The latency $T^{Wr}_{L,i}$ for a write-only connection c_i depends only on the forward channel as data are only sent in forward direction. The latency for the *occupied consumer* case is the same as the latency of the forward channel (see Equation (1-78)).

$$T^{Wr}_{L,i} = T^F_{L,i} = T^{F,M}_{L,i} + T^{F,T}_{L,i} + T^{F,S}_{L,i} \tag{1-86}$$

For the latency of the *unoccupied consumer* case the latency introduced due to the buffer in the forward channel at the slave side $T^{F,S}_{L,i}$ is always zero as consumer keeps the buffer empty.

The specialized formulas are derived by substituting the values of channel, component type and and data rates in Equations (1-83), (1-84) and (1-85).

$$\begin{aligned}
T^{F,M}_{L,i} &= \left\lfloor \frac{\beta^{F,M}_i}{W^F_{p,i}} \right\rfloor \times |\mathcal{S}| + \\
&\quad \max\left\{ \delta \in \mathbb{N} \mid (\beta^{F,M}_i \% W^F_{p,i}) \leq W^F_{p_{min},i}(\delta) < ((\beta^{F,M}_i \% W^F_{p,i}) + L_s) \right\} \\
T^{F,T}_{L,i} &= \text{Number of hops in forward direction.} \\
T^{F,S}_{L,i} &= \left\lceil \frac{\beta^{F,S}_i}{(1+\gamma^{Wr}_i) \times R^{Wr}_{IP,i} \times T^{Wr}_{IP,i}} \right\rceil \times \frac{T^{Wr}_{IP,i}}{T_s}
\end{aligned}$$

5.2 Latency for read-only connections

The latency $T^{Rd}_{L,i}$ for a read-only connection c_i, depends on the both forward and reverse channel as read commands are sent in the forward direction and read data are sent in the reverse direction. We further add the latency of IP to provide responses T^{IP}_L.

$$T^{Rd}_{L,i} = T^F_{L,i} + T^{IP}_L + T^R_{L,i} \tag{1-87}$$

By substitution Equation (1-78) for each channel, $T^{Rd}_{L,i}$ is given as:

$$T^{Rd}_{L,i} = T^{F,M}_{L,i} + T^{F,T}_{L,i} + T^{F,S}_{L,i} + T^{IP}_{L} + T^{R,S}_{L,i} + T^{R,T}_{L,i} + T^{R,M}_{L,i} \tag{1-88}$$

For the *unoccupied consumer* case, there are no latency contributions from slave buffer in the forward channel (i.e., $T^{F,S}_{L,i} = 0$) and master buffer in the reverse channel (i.e., $T^{R,M}_{L,i} = 0$) as both sides can accept data as soon as it is offered by the respective NIs.

The specialized formulas are derived by substituting the values of channel, component type and data rates in Equations (1-83), (1-84) and (1-85).

$$\begin{aligned}
T^{F,M}_{L,i} &= \left\lfloor \frac{\beta^{F,M}_{i}}{W^{F}_{p,i}} \right\rfloor \times |\mathcal{S}| + \\
&\quad \max \left\{ \delta \in \mathbb{N} \mid (\beta^{F,M}_{i} \% W^{F}_{p,i}) \leq W^{F}_{p_{min},i}(\delta) < ((\beta^{F,M}_{i} \% W^{F}_{p,i}) + L_s) \right\} \\
T^{F,T}_{L,i} &= \text{Number of hops in forward direction.} \\
T^{F,S}_{L,i} &= \left\lceil \frac{\beta^{F,S}_{i}}{\gamma^{Rd}_{i} \times R^{Rd}_{IP,i} \times T^{Rd}_{IP,i}} \right\rceil \times \frac{T^{Rd}_{IP,i}}{T_s} \\
T^{IP}_{L,i} &= \text{Latency of IP to provide responses after receiving requests.} \\
T^{R,S}_{L,i} &= \left\lfloor \frac{\beta^{R,S}_{i}}{W^{R}_{p,i}} \right\rfloor \times |\mathcal{S}| + \\
&\quad \max \left\{ \delta \in \mathbb{N} \mid (\beta^{R,S}_{i} \% W^{R}_{p,i}) \leq W^{R}_{p_{min},i}(\delta) < ((\beta^{R,S}_{i} \% W^{R}_{p,i}) + L_s) \right\} \\
T^{F,T}_{L,i} &= \text{Number of hops in reverse direction.} \\
T^{R,M}_{L,i} &= \left\lceil \frac{\beta^{R,M}_{i}}{R^{Rd}_{IP,i} \times T^{Rd}_{IP,i}} \right\rceil \times \frac{T^{Rd}_{IP,i}}{T_s}
\end{aligned}$$

5.3 Latency for read-write connections

By substituting the buffer sizes derived for read-write case from Section 4.3 in equations defined in Section 5.1 and Section 5.2, latency for write and read transactions can be calculated.

6. VERIFICATION TOOL AND RESULTS

We have developed a tool to verify the performance of a SoC against its specifications using the analytical method described in the previous sections. This verification tool takes as input NoC attributes (e.g., slot table size, slot size, word width, and frequency), a NoC configuration per connection (e.g.,

connection id, slot table allocation for both channels, number of hops for each channel, transaction type), and the specified values for throughput, latency and buffer sizes per connection. The tool derives the required buffering, the worst-case latency, and the minimum throughput for each connection. We have developed this tool in Matlab^{TM} and the tool produces results in XML (Extensible Markup Language) format which can be converted in HTML (Hypertext Markup Language) format.

The tool provides bounds for buffering for each connection and available slack (e.g., additional amount of buffer) from the specified values. This information allows to build correct NoCs with properly dimensioned buffers. The available slack information can be used to reduce the cost of a NoC by stripping additional buffers because the cost of a NoC is dominated by the cost of buffering (Rădulescu et al., 2005). In our tool flow (Goossens, Dielissen, Gangwal, González Pestana, Rădulescu and Rijpkema, 2005), for GT connections, we can automatically adjust buffer size to the derived bounds for each connection.

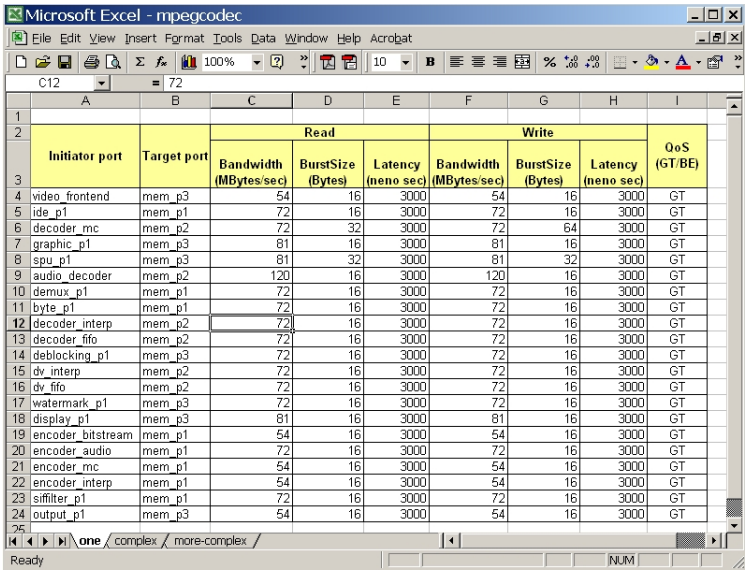
This tool derives exact values for throughput, assuming buffers are dimensioned using the derived bounds, for a given slot allocation. Furthermore, it checks whether the given slot allocation meets the specified data rates and the required flow control rates in the forward and the reverse direction. When these requirements are not met, the tool provides detailed feedback about what requirements are not met with exact numbers. Note that these verification equations have been incorporated in our slot allocation tool to build our NoCs in a correct-by-construction manner.

The tool also calculates the worst-case latency for both unoccupied consumer case and occupied consumer case, per connection basis. It also provides feedback whether we meet the specified latency requirements or not.

The tool processes one connection at a time, so, execution time of the tool is linear in number of connections in a SoC. The execution time was approximately a minute for a complex SoC with as many as 200 connections. We demonstrate the usefulness of the analysis method and the tool through an MPEG-2 codec example.

6.1 Example

The example MPEG-2 codec SoC has 16 IPs and 3 memories and 21 guaranteed throughput read-write connections (see Figure 1-15). A connection is specified between an initiator port and target port with read and/or write bandwidth requirements, burst size and latency requirements. These connections have bandwidth requirements varying from 54 to 120 Mbytes/sec and burst sizes varying from 16 to 64 bytes.



Microsoft Excel - mpegcodec

File Edit View Insert Format Tools Data Window Help Acrobat

C12 = 72

	Initiator port	Target port	Read			Write			QoS (GT/BE)
			Bandwidth (MBytes/sec)	BurstSize (Bytes)	Latency (neno sec)	Bandwidth (MBytes/sec)	BurstSize (Bytes)	Latency (neno sec)	
4	video_frontend	mem_p3	54	16	3000	54	16	3000	GT
5	ide_p1	mem_p1	72	16	3000	72	16	3000	GT
6	decoder_mc	mem_p2	72	32	3000	72	64	3000	GT
7	graphic_p1	mem_p3	81	16	3000	81	16	3000	GT
8	spu_p1	mem_p3	81	32	3000	81	32	3000	GT
9	audio_decoder	mem_p2	120	16	3000	120	16	3000	GT
10	demux_p1	mem_p1	72	16	3000	72	16	3000	GT
11	byte_p1	mem_p1	72	16	3000	72	16	3000	GT
12	decoder_interp	mem_p2	72	16	3000	72	16	3000	GT
13	decoder_fifo	mem_p2	72	16	3000	72	16	3000	GT
14	deblocking_p1	mem_p3	72	16	3000	72	16	3000	GT
15	dv_interp	mem_p2	72	16	3000	72	16	3000	GT
16	dv_fifo	mem_p2	72	16	3000	72	16	3000	GT
17	watermark_p1	mem_p3	72	16	3000	72	16	3000	GT
18	display_p1	mem_p3	81	16	3000	81	16	3000	GT
19	encoder_bitstream	mem_p1	54	16	3000	54	16	3000	GT
20	encoder_audio	mem_p1	72	16	3000	72	16	3000	GT
21	encoder_mc	mem_p1	54	16	3000	54	16	3000	GT
22	encoder_interp	mem_p1	54	16	3000	54	16	3000	GT
23	siffilter_p1	mem_p1	72	16	3000	72	16	3000	GT
24	output_p1	mem_p3	54	16	3000	54	16	3000	GT

one complex more-complex

Ready NUM

Figure 1-15. Description of connections for the example MPEG-2 codec.

6.2 Analysis Results and Observations

We build two NoCs for the example MPEG-2 codec. The first is an automatically generated (using our tool flow (Goossens, Dielissen, Gangwal, González Pestana, Rădulescu and Rijpkema, 2005; Goossens, González Pestana, Dielissen, Gangwal, van Meerbergen, Rădulescu, Rijpkema and Wielage, 2005) minimum mesh topology of size 2x3 using a slot table size of 64 (called *ex64*) and the second is manually mapped and dimensioned 1x3 mesh topology using a slot table size of 8 (called *ex8*). The results for both examples are shown in Figure 1-16 and 1-17, respectively. First we describe what is shown in the result tables then we compare results for both NoCs. The key points to observe in this comparison are the effects of the size of a NoC, the size of a slot table, and the burst size on the buffer sizes, the latency, and the throughput of a connection.

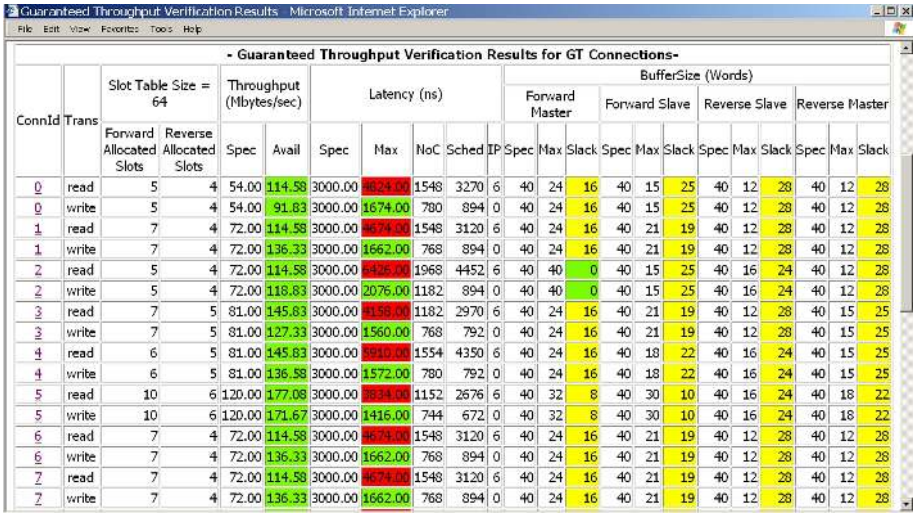
The first column of the table shows the unique connection identifiers of the connections, the second column shows the type of transactions allowed on the connection (i.e., read and/or write). The third and fourth columns provide information about slot table size and allocated number of slots for both forward and reverse channel. The fifth and sixth columns show the specified throughput values and available throughput for the given slot allocation in Mbytes per second. Notice that the available throughput is not exactly equal to the spec-

ified value rather it is usually more than the specified value because the slot allocation results in terms of integer number of slots, the available bandwidth may exceed the specified bandwidth. Note that the bandwidth per slot depends on the slot table size (i.e., the larger the slot table the smaller is the bandwidth per slot). A larger slot table provides more options to match closely with the specified throughput. For example, the available throughput for read connection 1 matches more closely to the specified value (i.e., 72 MB/s) for the ex64 architecture with 64 slots (i.e., 114.58 MB/s) than for the ex8 architecture with only 8 slots (i.e., 166.67 MB/s).

The next four columns are for the latency represented in nanoseconds. It shows the specified latency and worst-case latency including contributions from the NoC, contributions of an occupied consumer port, (i.e., *Sched* column), and the given latency of the consumer to send responses, (i.e., *IP* column) (see Figure 1-12). Note that, for low bandwidth read connections scheduling latencies dominate the unoccupied consumer latency. A connection with low bandwidth requirements gets its turn to be served by the consumer after long time (as the port of consumer is occupied with other connections of high bandwidth requirements) leading to high latency. By deriving unoccupied consumer latency and latency due to an occupied consumer port separately, one can understand which part of the latency is due to what reason. When comparing the results for the ex64 architecture with the ex8 architecture, the latency for the ex64 architecture is always larger than for the ex8 architecture due to larger buffer sizes and larger mesh for the ex64 architecture.

The rest of the columns show the specified and the computed buffer sizes, and the slack for all four buffers (forward master, forward slave, reverse slave and reverse master), respectively. The slack information tells where additional buffering is required (when it is a negative number), and where the specified buffer size is higher than needed (when it is a positive number). The buffer sizes for all buffers is larger for the ex64 architecture than the ex8 architecture due to larger NoC (mesh) and more number of allocated slots. Recalling the equations for buffering, we observe that buffer sizes are proportional to the number of allocated slots per channel. We can also back annotate these buffer sizes per connection automatically and run the analysis again. The results after running the analysis are shown in Figure 1-18.

When comparing the results for the two different topologies (ex64 and ex8) for the given example, we observe that larger slot table sizes allow a good match for the specified throughput requirements but they result in larger buffer sizes and latency. The net effect of all these is an increase in area due to the larger slot tables itself and larger buffer sizes. Figure 1-19 provides the overview of area numbers (for 0.13μ CMOS technology) for both topologies with derived buffering (i.e., buf-opt suffix) and with a fixed buffer size of 40 (i.e., buf-no-opt suffix). As expected router cost is higher for ex64 topology as



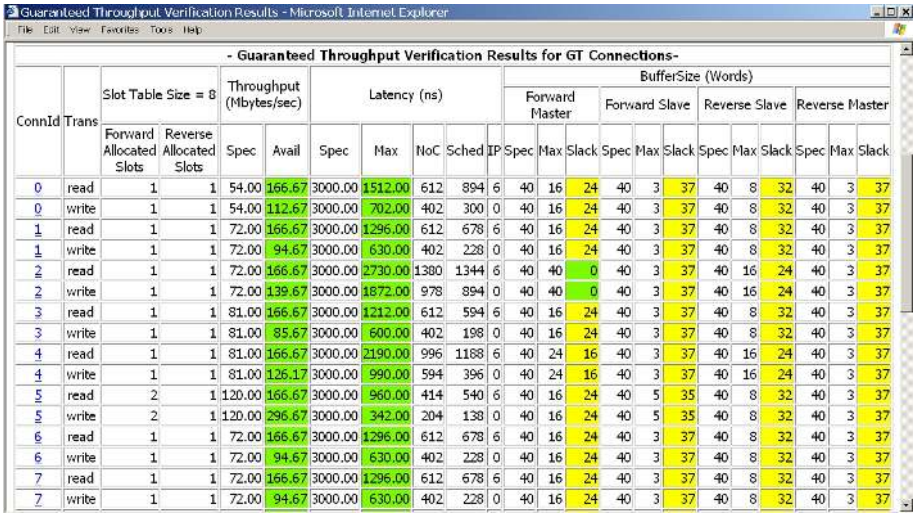
Guaranteed Throughput Verification Results - Microsoft Internet Explorer

File Edit View Favorites Tools Help

- Guaranteed Throughput Verification Results for GT Connections-

ConnId	Trans	Slot Table Size = 64		Throughput (Mbytes/sec)		Latency (ns)					BufferSize (Words)											
											Forward Master			Forward Slave			Reverse Slave			Reverse Master		
		Forward Allocated Slots	Reverse Allocated Slots	Spec	Avail	Spec	Max	NoC	Sched	IP	Spec	Max	Slack	Spec	Max	Slack	Spec	Max	Slack	Spec	Max	Slack
0	read	5	4	54.00	114.58	3000.00	4624.00	1548	3270	6	40	24	16	40	15	25	40	12	28	40	12	28
0	write	5	4	54.00	91.83	3000.00	1674.00	780	894	0	40	24	16	40	15	25	40	12	28	40	12	28
1	read	7	4	72.00	114.58	3000.00	4674.00	1548	3120	6	40	24	16	40	21	19	40	12	28	40	12	28
1	write	7	4	72.00	136.33	3000.00	1662.00	768	894	0	40	24	16	40	21	19	40	12	28	40	12	28
2	read	5	4	72.00	114.58	3000.00	6426.00	1968	4452	6	40	40	0	40	15	25	40	16	24	40	12	28
2	write	5	4	72.00	118.83	3000.00	2076.00	1182	894	0	40	40	0	40	15	25	40	16	24	40	12	28
3	read	7	5	81.00	145.83	3000.00	4158.00	1182	2970	6	40	24	16	40	21	19	40	12	28	40	15	25
3	write	7	5	81.00	127.33	3000.00	1560.00	768	792	0	40	24	16	40	21	19	40	12	28	40	15	25
4	read	6	5	81.00	145.83	3000.00	5910.00	1554	4350	6	40	24	16	40	18	22	40	16	24	40	15	25
4	write	6	5	81.00	136.58	3000.00	1572.00	780	792	0	40	24	16	40	18	22	40	16	24	40	15	25
5	read	10	6	120.00	177.08	3000.00	3834.00	1152	2676	6	40	32	8	40	30	10	40	16	24	40	18	22
5	write	10	6	120.00	171.67	3000.00	1416.00	744	672	0	40	32	8	40	30	10	40	16	24	40	18	22
6	read	7	4	72.00	114.58	3000.00	4674.00	1548	3120	6	40	24	16	40	21	19	40	12	28	40	12	28
6	write	7	4	72.00	136.33	3000.00	1662.00	768	894	0	40	24	16	40	21	19	40	12	28	40	12	28
7	read	7	4	72.00	114.58	3000.00	4674.00	1548	3120	6	40	24	16	40	21	19	40	12	28	40	12	28
7	write	7	4	72.00	136.33	3000.00	1662.00	768	894	0	40	24	16	40	21	19	40	12	28	40	12	28

Figure 1-16. Results of GT verification for the ex64 architecture.



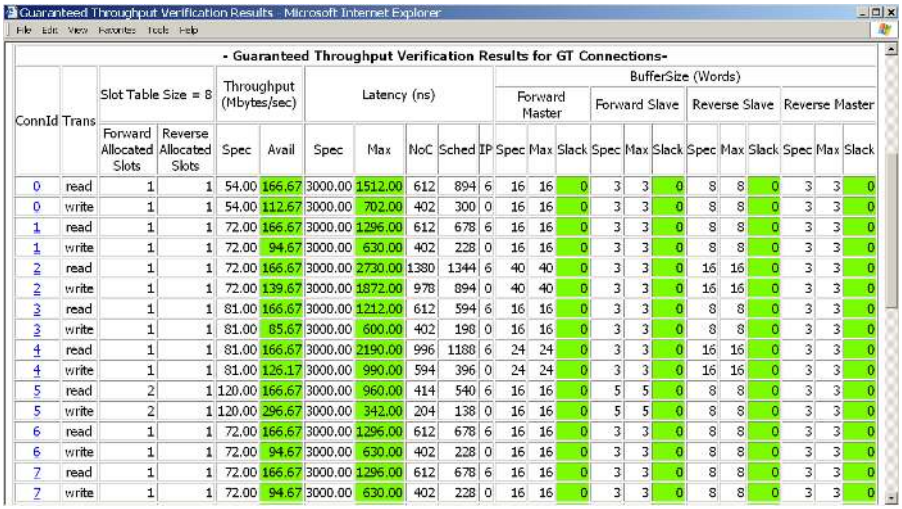
Guaranteed Throughput Verification Results - Microsoft Internet Explorer

File Edit View Favorites Tools Help

- Guaranteed Throughput Verification Results for GT Connections-

ConnId	Trans	Slot Table Size = 8		Throughput (Mbytes/sec)		Latency (ns)					BufferSize (Words)											
											Forward Master			Forward Slave			Reverse Slave			Reverse Master		
		Forward Allocated Slots	Reverse Allocated Slots	Spec	Avail	Spec	Max	NoC	Sched	IP	Spec	Max	Slack	Spec	Max	Slack	Spec	Max	Slack	Spec	Max	Slack
0	read	1	1	54.00	166.67	3000.00	1512.00	612	894	6	40	16	24	40	3	37	40	8	32	40	3	37
0	write	1	1	54.00	112.67	3000.00	702.00	402	300	0	40	16	24	40	3	37	40	8	32	40	3	37
1	read	1	1	72.00	166.67	3000.00	1296.00	612	678	6	40	16	24	40	3	37	40	8	32	40	3	37
1	write	1	1	72.00	94.67	3000.00	630.00	402	228	0	40	16	24	40	3	37	40	8	32	40	3	37
2	read	1	1	72.00	166.67	3000.00	2730.00	1380	1344	6	40	40	0	40	3	37	40	16	24	40	3	37
2	write	1	1	72.00	139.67	3000.00	1872.00	978	894	0	40	40	0	40	3	37	40	16	24	40	3	37
3	read	1	1	81.00	166.67	3000.00	1212.00	612	594	6	40	16	24	40	3	37	40	8	32	40	3	37
3	write	1	1	81.00	85.67	3000.00	600.00	402	198	0	40	16	24	40	3	37	40	8	32	40	3	37
4	read	1	1	81.00	166.67	3000.00	2190.00	996	1188	6	40	24	16	40	3	37	40	16	24	40	3	37
4	write	1	1	81.00	126.17	3000.00	990.00	594	396	0	40	24	16	40	3	37	40	16	24	40	3	37
5	read	2	1	120.00	166.67	3000.00	960.00	414	540	6	40	16	24	40	5	35	40	8	32	40	3	37
5	write	2	1	120.00	296.67	3000.00	342.00	204	138	0	40	16	24	40	5	35	40	8	32	40	3	37
6	read	1	1	72.00	166.67	3000.00	1296.00	612	678	6	40	16	24	40	3	37	40	8	32	40	3	37
6	write	1	1	72.00	94.67	3000.00	630.00	402	228	0	40	16	24	40	3	37	40	8	32	40	3	37
7	read	1	1	72.00	166.67	3000.00	1296.00	612	678	6	40	16	24	40	3	37	40	8	32	40	3	37
7	write	1	1	72.00	94.67	3000.00	630.00	402	228	0	40	16	24	40	3	37	40	8	32	40	3	37

Figure 1-17. Results of GT verification for the ex8 architecture.



Guaranteed Throughput Verification Results - Microsoft Internet Explorer

File Edit View Favorites Tools Help

- Guaranteed Throughput Verification Results for GT Connections-

ConnId	Trans	Slot Table Size = 8		Throughput (Mbytes/sec)		Latency (ns)					BufferSize (Words) Forward Master			Forward Slave			Reverse Slave			Reverse Master		
		Forward Allocated Slots	Reverse Allocated Slots	Spec	Avail	Spec	Max	NoC	Sched	IP	Spec	Max	Slack	Spec	Max	Slack	Spec	Max	Slack	Spec	Max	Slack
0	read	1	1	54.00	166.67	3000.00	1512.00	612	894	6	16	16	0	3	3	0	8	8	0	3	3	0
0	write	1	1	54.00	112.67	3000.00	702.00	402	300	0	16	16	0	3	3	0	8	8	0	3	3	0
1	read	1	1	72.00	166.67	3000.00	1296.00	612	678	6	16	16	0	3	3	0	8	8	0	3	3	0
1	write	1	1	72.00	94.67	3000.00	630.00	402	228	0	16	16	0	3	3	0	8	8	0	3	3	0
2	read	1	1	72.00	166.67	3000.00	2730.00	1380	1344	6	40	40	0	3	3	0	16	16	0	3	3	0
2	write	1	1	72.00	139.67	3000.00	1872.00	978	894	0	40	40	0	3	3	0	16	16	0	3	3	0
3	read	1	1	81.00	166.67	3000.00	1212.00	612	594	6	16	16	0	3	3	0	8	8	0	3	3	0
3	write	1	1	81.00	85.67	3000.00	600.00	402	198	0	16	16	0	3	3	0	8	8	0	3	3	0
4	read	1	1	81.00	166.67	3000.00	2190.00	996	1188	6	24	24	0	3	3	0	16	16	0	3	3	0
4	write	1	1	81.00	126.17	3000.00	990.00	594	396	0	24	24	0	3	3	0	16	16	0	3	3	0
5	read	2	1	120.00	166.67	3000.00	960.00	414	540	6	16	16	0	5	5	0	8	8	0	3	3	0
5	write	2	1	120.00	296.67	3000.00	342.00	204	138	0	16	16	0	5	5	0	8	8	0	3	3	0
6	read	1	1	72.00	166.67	3000.00	1296.00	612	678	6	16	16	0	3	3	0	8	8	0	3	3	0
6	write	1	1	72.00	94.67	3000.00	630.00	402	228	0	16	16	0	3	3	0	8	8	0	3	3	0
7	read	1	1	72.00	166.67	3000.00	1296.00	612	678	6	16	16	0	3	3	0	8	8	0	3	3	0
7	write	1	1	72.00	94.67	3000.00	630.00	402	228	0	16	16	0	3	3	0	8	8	0	3	3	0

Figure 1-18. Results of GT verification with derived buffer sizes for the ex8 architecture.

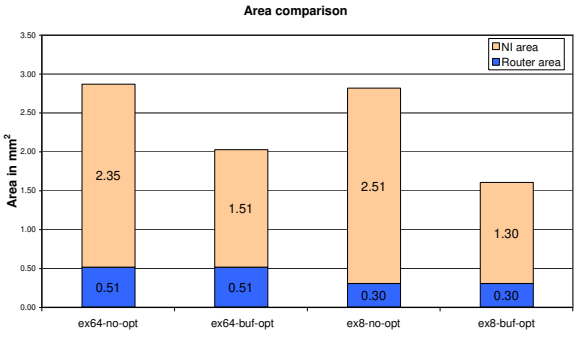


Figure 1-19. Comparison of various topologies and buffering.

compared to ex8. It is clear from Figure 1-19 that cost of NI dominates the cost of network. The cost of the network can thus be reduced by carefully choosing the network parameters (e.g., slot table size and number of routers and their connections to IPs).

Our technique allows analysis of each GT connection independently leading to a composable design. The complete design is analyzable without the use of any simulation techniques.

7. CONCLUSIONS

To build predictable systems all components of the system must be predictable. This includes computation components, memory components and communication components. Our focus is on predictable communication components (i.e., Æthereal NoC). We explain that guaranteed services are required to do worst-case analysis of a NoC without performing time consuming simulations that may not cover the worst-case, anyways. An analyzable communication infrastructure is a must to build correct-by-construction predictable systems. We derived bounds for worst-case values for buffer sizes, latency, and throughput for the Æthereal NoC. We show how these techniques can be applied using an MPEG-2 codec example. Furthermore, we show that the cost of the communication infrastructure can be reduced with derived values for buffer sizes and quick exploration of various topologies through our analysis without a need of simulation.

8. GLOSSARY

Symbol	**Brief description (subscript *i* denotes a connection identifier)**	**Page**
$\beta_{DC,i}^{F,M}$	Decoupling buffer size for forward channel at master side, in words	18
$\beta_{DC,i}^{F,S}$	Decoupling buffer size for forward channel at slave side, in words	18
$\beta_{DC,i}^{R,M}$	Decoupling buffer size for reverse channel at master side, in words	18
$\beta_{DC,i}^{R,S}$	Decoupling buffer size for reverse channel at slave side, in words	18
$\beta_{RL,i}^{F,S}$	Buffer size needed to hide the round-trip latency delay of the flow control for the forward channel at the slave side, in words	21
	Continued on next page ...	

Symbol	Brief description	Page
$\beta_{RL,i}^{R,M}$	Buffer size needed to hide the round-trip latency delay of the flow control for the reverse channel at the master side, in words	21
$\beta_i^{F,M}$	Total buffer size at the master side of the forward channel, in words	21
$\beta_i^{F,S}$	Total buffer size at the slave side of the forward channel, in words	21
$\beta_i^{R,M}$	Total buffer size at the master side of the reverse channel, in words	21
$\beta_i^{R,S}$	Total buffer size at the slave side of the reverse channel, in words	21
γ_i^{Wr}	Command to data ratio for a write connection	8
γ_i^{Rd}	Command to data ratio for a for a read connection	8
$\Theta_{FC,i}^{ch}$	Flow control symbol rate for channel *ch*, in Symbol/sec	10
B_L	Raw link bandwidth, in words/sec	6
$B_{h,i}^{ch}$	Header bandwidth for channel *ch*, in words/sec	10
$B_{p,i}^{ch}$	Payload bandwidth for channel *ch*, in words/sec	10
$B_{r,i}^{ch}$	Total raw bandwidth for channel *ch*, in words/sec	10
B_s	The bandwidth associated to a reserved slot, in words/sec	7
B_w	The bandwidth associated to a reserved word, in words/sec	7
$\mathcal{E}_i^{ch}$	Blocks of contiguous slots not allocated to channel *ch*	9
$\mathcal{F}_i^{ch}$	Blocks of contiguous slots allocated to channel *ch*	9
f_{noc}	Frequency of a NoC, in Hz	6
$\mathcal{H}_i^{ch}$	Slots containing headers for channel *ch*	9
L_{CMD}	Number of words used to encode the command and address in a message	8
L_{DATA}	The amount of words that is transferred in a single transaction	8
L_h	Packet header length, in words	5
L_s	Slot size, in words	7
L_w	Word length, in bits	5

Continued on next page ...

Symbol	Brief description	Page
M_{FC}	The maximum flow control value that can be sent in one header (or symbol)	6
$R^{Rd}_{CMD,i}$	The available data throughput for read commands, in words/sec	12
$R^{Wr}_{CMD,i}$	The available data throughput for write commands, in words/sec	11
$R^{Rd}_{DATA,i}$	The available data throughput for read data, in words/sec	12
$R^{Wr}_{DATA,i}$	The available data throughput for write data, in words/sec	11
$R^{Rd}_{IP,i}$	The specified data throughput for a read connection, in words/sec	8
$R^{Wr}_{IP,i}$	The specified data throughput for a write connection, in words/sec	8
$\mathcal{S}$	Sequence of slots in a slot table	7
$\|\mathcal{S}\|$	Slot table size	7
$\mathcal{S}^{ch}_i$	Sequence of slots allocated to channel *ch*	7
$T^{Rd}_{IP,i}$	The period with which an IP module issues/processes read commands, in seconds	8
$T^{Wr}_{IP,i}$	The period with which an IP module issues/processes write commands, in seconds	8
T^{IP}_{L}	IP latency to provide responses after a request is made, in seconds	25
$T_{L,i}$	Latency of a connection c_i, in seconds	21
$T^{F,T}_{L,i}$	Latency to transport data in the forward channel, in seconds (given by the number of hops)	20
$T^{R,T}_{L,i}$	Latency to transport data in the reverse channel, in seconds (given by the number of hops)	20
$T^{ch}_{L,i}$	Latency of a channel *ch* in seconds	22
$T^{Rd}_{L,i}$	Latency of a read-only connection, in seconds	26
$T^{Wr}_{L,i}$	Latency of a write-only connection, in seconds	25
T_{noc}	Clock period of a NoC, in seconds	6
T_s	Time required to traverse a slot, in seconds	7
$W^{ch}_{h,i}$	Number of header words sent in one iteration of	10

Continued on next page ...

Symbol	Brief description	Page
	the slot table for channel *ch*	
$W^{ch}_{p,i}$	Number of payload words sent in one iteration of the slot table for channel *ch*	10
$W^{ch}_{r,i}$	Total number of words sent in one iteration of the slot table for channel *ch*	10

References

Adriahantenaina, A., Charlery, H., Greiner, A., Mortiez, L. and Zeferino, C. A., 2003, SPIN: A scalable, packet switched, on-chip micro-network, *Proc. Design, Automation and Test in Europe Conference and Exhibition (DATE)*.

ARM Ltd., 2003, *AMBA AXI Protocol Specification*.

Benini, L. and De Micheli, G., 2001, Powering networks on chips, *Int'l Symposium on System Synthesis (ISSS)*, pp. 33–38.

Benini, L. and De Micheli, G., 2002, Networks on chips: A new SoC paradigm, *IEEE Computer* **35**(1), 70–80.

Bolotin, E., Cidon, I., Ginosar, R. and Kolodny, A., 2004, QNoC: QoS architecture and design process for network on chip, *Journal of Systems Architecture* **50**(2–3), 105–128. Special issue on Networks on Chip.

Dally, W. J. and Towles, B., 2001, Route packets, not wires: on-chip interconnection networks, *Proc. Design Automation Conference (DAC)*, pp. 684–689.

Goossens, K., Dielissen, J., Gangwal, O. P., González Pestana, S., Rădulescu, A. and Rijpkema, E., 2005, A design flow for application-specific networks on chip with guaranteed performance to accelerate SOC design and verification, *Proc. Design, Automation and Test in Europe Conference and Exhibition (DATE)*.

Goossens, K., Gangwal, O. P., Röver, J. and Niranjan, A. P., 2004, Interconnect and memory organization in SOCs for advanced set-top boxes and TV — Evolution, analysis, and trends, *in* J. Nurmi, H. Tenhunen, J. Isoaho and A. Jantsch (eds), *Interconnect-Centric Design for Advanced SoC and NoC*, Kluwer, chapter 15, pp. 399–423.

Goossens, K., González Pestana, S., Dielissen, J., Gangwal, O. P., van Meerbergen, J., Rădulescu, A., Rijpkema, E. and Wielage, P., 2005, Service-based design of systems on chip and networks on chip, *in* P. van der

Stok (ed.), *Dynamic and Robust Streaming in and Between Connected Consumer-Electronic Devices*, Kluwer.

Goossens, K., van Meerbergen, J., Peeters, A. and Wielage, P., 2002, Networks on silicon: Combining best-effort and guaranteed services, *Proc. Design, Automation and Test in Europe Conference and Exhibition (DATE)*, pp. 423–425.

Guerrier, P. and Greiner, A., 2000, A generic architecture for on-chip packet-switched interconnections, *Proc. Design, Automation and Test in Europe Conference and Exhibition (DATE)*, pp. 250–256.

Karim, F., Nguyen, A. and Dey, S., 2002, An interconnect architecture for networking systems on chips, *IEEE Micro* **22**(5), 36–45.

Liang, J., Swaminathan, S. and Tessier, R., 2000, aSOC: A scalable, single-chip communications architecture, *Proc. Int'l Conference on Parallel Architectures and Compilation Techniques (PACT)*.

Millberg, M., Nilsson, E., Thid, R. and Jantsch, A., 2004, Guaranteed bandwidth using looped containers in temporally disjoint networks within the Nostrum network on chip, *Proc. Design, Automation and Test in Europe Conference and Exhibition (DATE)*.

OCP International Partnership, 2003, *Open Core Protocol Specification. 2.0 Release Candidate*.

Rădulescu, A., Dielissen, J., González Pestana, S., Gangwal, O. P., Rijpkema, E., Wielage, P. and Goossens, K., 2005, An efficient on-chip network interface offering guaranteed services, shared-memory abstraction, and flexible network programming, *IEEE Transactions on CAD of Integrated Circuits and Systems* **24**(1).

Rădulescu, A. and Goossens, K., 2004, Communication services for networks on chip, *in* S. S. Bhattacharyya, E. F. Deprettere and J. Teich (eds), *Domain-Specific Processors: Systems, Architectures, Modeling, and Simulation*, Marcel Dekker, pp. 193–213.

Rijpkema, E., Goossens, K. G. W., Rădulescu, A., Dielissen, J., van Meerbergen, J., Wielage, P. and Waterlander, E., 2003, Trade offs in the design of a router with both guaranteed and best-effort services for networks on chip, *Proc. Design, Automation and Test in Europe Conference and Exhibition (DATE)*, pp. 350–355.

Philips Semiconductors, 2002, *Device Transaction Level (DTL) Protocol Specification. Version 2.2*.

Tanenbaum, A. S., 1996, *Computer Networks*, Prentice-Hall.

Wiklund, D. and Liu, D., 2003, Socbus: switched network on chip for hard real time embedded systems, *Proc. Int'l Parallel and Distributed Processing Symposium (IPDPS)*.