Packetizing OCP Transactions in the MANGO Network-on-Chip

Bjerregaard, Tobias; Sparsø, Jens

Published in:
Proceedings of the 9th Euromicro Conference on Digital System Design, August

Link to article, DOI:
10.1109/DSD.2006.75

Publication date:
2006

Document Version
Publisher's PDF, also known as Version of record

Packetizing OCP Transactions in the MANGO Network-on-Chip

Tobias Bjerregaard and Jens Sparsø
Informatics and Mathematical Modelling
Technical University of Denmark (DTU)
2800 Lyngby, Denmark
{tob, jsp}@imm.dtu.dk

Abstract

The scaling of CMOS technology causes a widening gap between the performance of on-chip communication and computation. This calls for a communication-centric design flow. The MANGO network-on-chip architecture enables globally asynchronous locally synchronous (GALS) system-on-chip design, while facilitating IP reuse by standard socket access points. Two types of services are available: connection-less best-effort routing and connection-oriented guaranteed service (GS) routing. This paper presents the core-centric programming model for establishing and using GS connections in MANGO. We show how OCP transactions are packetized and transmitted across the shared network, and illustrate how this affects the end-to-end performance. A high predictability of the latency of communication on shared links is shown in a MANGO-based demonstrator system.

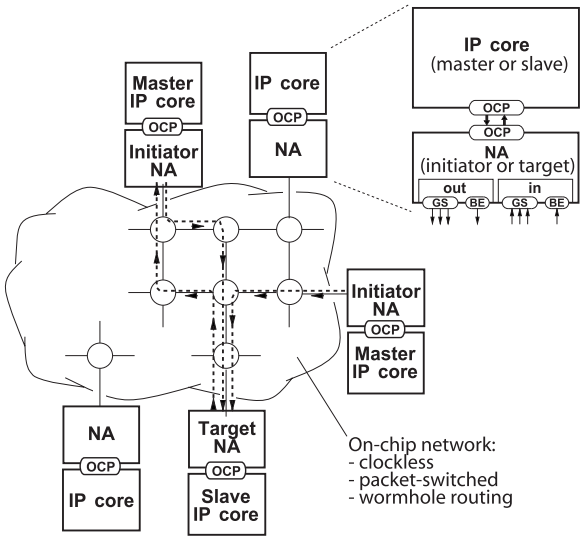

Figure 1. Core-centric view of a MANGO-based system.

1. Introduction

While transistor speeds improve with each new CMOS fabrication technology, wire speeds worsen. When scaling wires, the resistance per mm increases. The capacitance stays roughly constant, being mostly due to edge effects [12]. Hence the RC delay for a constant length wire increases. With a projected processor speed of 40 GHz in 2016, the latency cost of driving data 1 mm across a chip, on broad, top-level, *global* wires, will be 32 clock cycles [1]. While wire segmentation and pipelining can help, ultimately the result is a widening performance gap between communication and computation. Thus it has been evident for some time now, that data trafficking – not processing – will be the performance bottleneck in future single-chip systems. This calls for a communication-centric design flow.

Segmented interconnection networks, so called networks-on-chip (NoC), constitute a viable solution space to the communication challenges of future system-on-chip (SoC) designs [9][3][13]. While there are many approaches to increasing the throughput in such networks (e.g. pipelining), the fundamental drawback of scaling technologies concerns the latency of communication. In previous works, we have presented novel solutions relevant to the development of a clockless NoC MANGO (*Message-passing Asynchronous Network-on-chip providing Guaranteed services over OCP interfaces*) [4][5][7][6]. Of particular novelty is the ALG (*Asynchronous Latency Guarantees*) scheduling discipline employed on the network links. In contrast to time division multiplexing (TDM) scheduling schemes commonly used in NoCs [14][10], ALG provides latency and bandwidth guarantees which are not inversely dependent.

In this paper, we present the use of guaranteed service (GS) connections in MANGO from a system point-of-view. We demonstrate with simulations of a 0.13 μm prototype, showing end-to-end performance results. In Section 2 we describe the core-centric view of MANGO, and explain the

Proceedings of the 9th EUROMICRO Conference on Digital System Design (DSD'06)
0-7695-2609-8/06 $20.00 © 2006 IEEE

Table 1. OCP signal subset.

Signal Group	OCP Signal	Driver	Function
Basic	Clk	Master IP / Slave IP	OCP clock
	MAddr	Master IP / Target NA	transfer address
	MCmd	Master IP / Target NA	transfer command (write/read)
	MData	Master IP / Target NA	write data
	MDataValid	Master IP / Target NA	write data valid
	MRespAccept	Master IP / Target NA	master accepts response
	SCmdAccept	Slave IP / Initiator NA	initiator accepts transfer
	SData	Slave IP / Initiator NA	read data
	SDataAccept	Slave IP / Initiator NA	initiator accepts write data
	SResp	Slave IP / Initiator NA	transfer response
Burst extensions	MBurstLength	Master IP / Target NA	burst length
	MBurstPrecise	Master IP / Target NA	given burst length is precise
	MBurstSeq	Master IP / Target NA	address sequence of burst
	MBurstSingleReq	Master IP / Target NA	single request / multiple data
	MDataLast	Master IP / Target NA	last write data in burst
	MReqLast	Master IP / Target NA	last request in burst
	SRespLast	Slave IP / Initiator NA	last response in burst
Thread extensions	MConnID	Master IP	connection identifier
	MDataThreadID	Master IP / Target NA	write data thread identifier
	MThreadID	Master IP / Target NA	request thread identifier
	SThreadID	Slave IP / Initiator NA	reponse thread identifier
Sideband	SInterrupt	Slave IP / Initiator NA	slave interrupt

basics of its layered communication strategy. In Section 3 we describe the encapsulation of OCP (Open Core Protocol [2]) transactions in packets, and the two types of routing services available. Section 4 presents the service guarantees of the ALG scheduling scheme, and in Section 5 we put these in a system context, and relate to experimental results from the demonstrator system. Section 6 provides a conclusion.

2. Core-Centric View

This section introduces MANGO from the point of view of the communicating IP cores. As illustrated in Figure 1 a MANGO-based system is composed of master and slave IP cores, the packet switched network itself, and network adapters (NAs) through which the IP-cores connect to the network. A layered communication strategy is adapted in which read/write-style transactions are provided based on the message-passing primitives of the underlying shared network. Figure 2 illustrates the communication layers. The *transaction* layer is defined by the OCP specification. The *network* layer constitutes the encapsulation of OCP transactions into packets by the NAs. The network uses wormhole routing, packets being transmitted as sequences of *flits* (flow control units) across the network links. This defines the *link* layer. Such layering facilitates modularity in system design, and portability of IP cores (IP reuse).

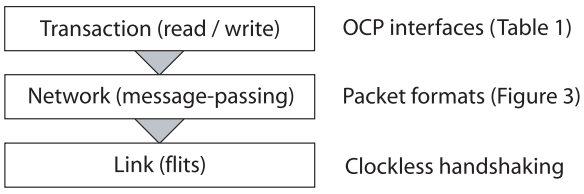

Figure 2. Layered communication stack.

The network is implemented using clockless circuit techniques, and each IP core may be clocked independently. The necessary synchronization to the local clock domain is performed in the NAs. This facilitates a globally asynchronous locally synchronous (GALS) system view [8][15], further helping to enable a modular system composition.

The IP/NA interface conforms to the OCP specification and provides read/write-style transactions into a distributed shared address space. In the presented prototype implementation the following OCP-transactions are supported: single reads and writes, burst reads and writes, and the use of threads, connections, and interrupts. Table 1 lists the subset of OCP signals implemented. Notice how signals are driven either by the master IP and the target NA (the target NA replicating the OCP transaction initiated by the master), or the slave IP and the initiator NA (the initiator NA replicating the response of the slave). The only exceptions are the connection identifier (*MConnID*) which is only used at the master IP, for specifying a connection through the network (further explained below), and the clock which is driven by the IP cores.

The network provides two types of routing services for establishing master to slave connectivity: best-effort (BE) services for which no performance guarantees are provided, and connection-oriented guaranteed services (GS) for which latency and bandwidth guarantees are made.

OCP-write transactions require only a forward path (request) from master IP to slave IP, while read transactions require also a return path (response). Any combination of BE and GS routing services is possible for the request and response.

As illustrated in Figure 1 each NA in the prototype has 4 unidirectional input ports and 4 unidirectional output ports connecting to the network. One ingoing and one outgoing port is dedicated for BE packets and the remaining ports are dedicated to GS connections. The choice of outgoing port is part of the OCP-transaction and is indicated using the connection identifier signal, *MConnID*.

BE packets are source routed, based on routing information in the packet header (further details in Section 3.2). To implement this each NA includes a small routing table which is indexed by the upper 8 bits of the OCP address, and which provides the routing information. Thus up to 256 slave units are supported. The entire system may contain more slave units but a maximum of 256 can be known to a given NA at any given time. In an actual implementation it will be considerably smaller, as it constitutes a major part of the NA area. The routing table may however be dynamically updated. It is implemented as a content addressable memory.

For establishing GS connections, the links of the network implement a number of independently buffered, logically separate virtual channels (VCs). GS packets are streamed from NA to NA along predefined virtual circuits, i.e., a reserved sequence of VCs through the network. Each router-node in the network allows static connections from input VCs to output VCs to be set up (input-to-output hops). When using GS connections the upper 8 bits of the OCP address are ignored, as the connection ID uniquely defines the destination NA.

The routing tables in the NAs and the "VC hop tables" in the individual router-nodes are mapped into the distributed shared address space. They can can be written to using BE transactions, by any master in the system, being addressable as IP slave cores. In a real application we envision a central network controller will be used to configure the network.

3. Message-Passing

The objective of a layered communication strategy is to enable modularity at the system level. The underlying hardware features are transparent to the system designer. In order to optimally utilize these underlying resources however, the system programmer should be aware of the basic mechanisms. In the following we will explain the routing features of MANGO, and by dissecting OCP transactions we will show how the use of GS routing affects end-to-end performance.

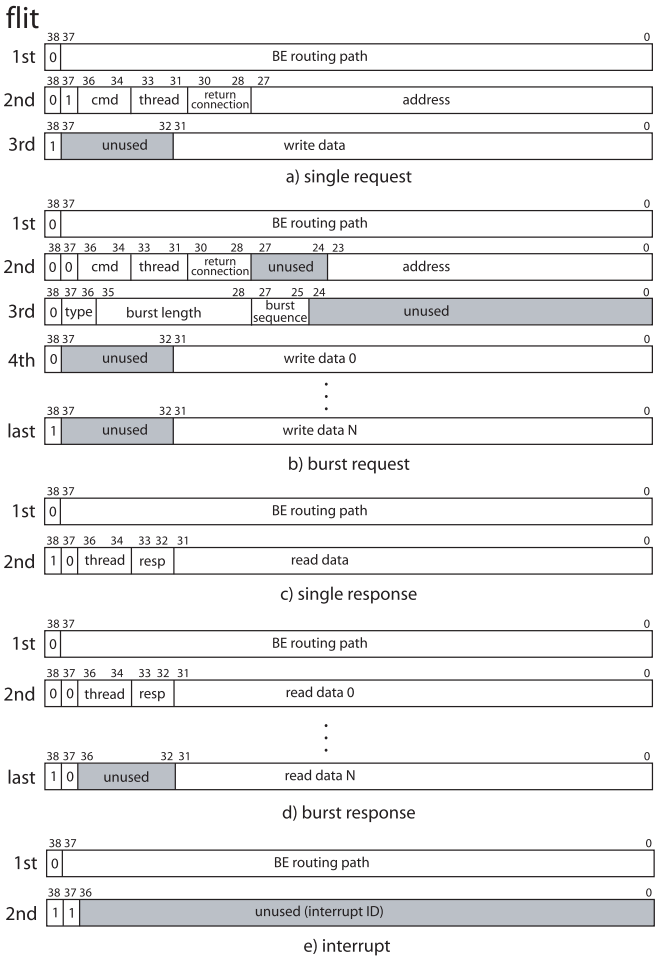

Figure 3. Packet formats for encapsulating OCP transactions for network transmission.

3.1. Packet Formats

Figure 3 details the different types of packets generated by the NAs. Requests are made by a master IP core while responses are made by the slave IP cores to these requests. Interrupts are generated by the slaves. The first bit in a flit is the end-of-packet bit, which is high only in the last flit of a packet. The first flit of BE packets is the header which holds the routing path. Transmitting on GS connections, there is no header flit as a connection uniquely defines the complete path from master to slave (request) or slave to master (response or interrupt).

3.2. Best-Effort Routing

No guarantees apart from completion and correctness are given for BE traffic. OCP commands are issued as BE transactions by addressing connection 0 (setting *MconnID = 0* on the OCP interface). Upon issuing an OCP transaction on connection 0, the upper 8 bits of the OCP address field are

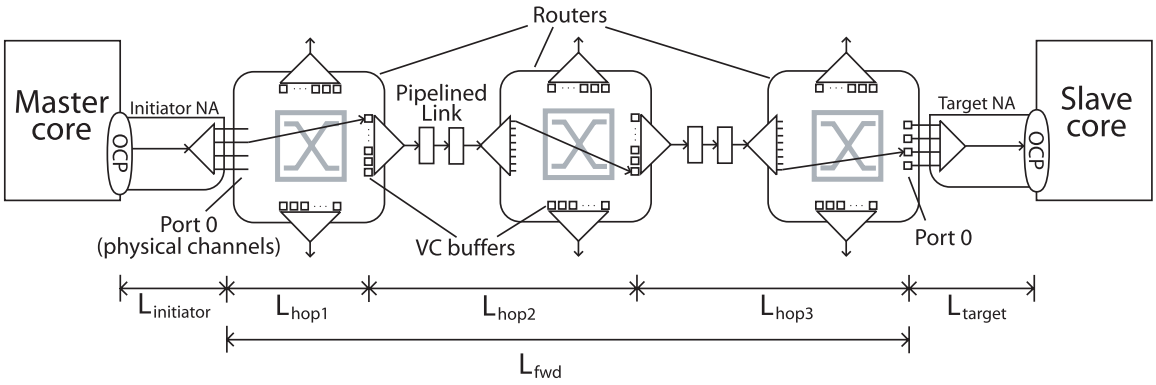

Figure 4. A GS connection constitutes a sequence of reserved VC buffers (a virtual circuit).

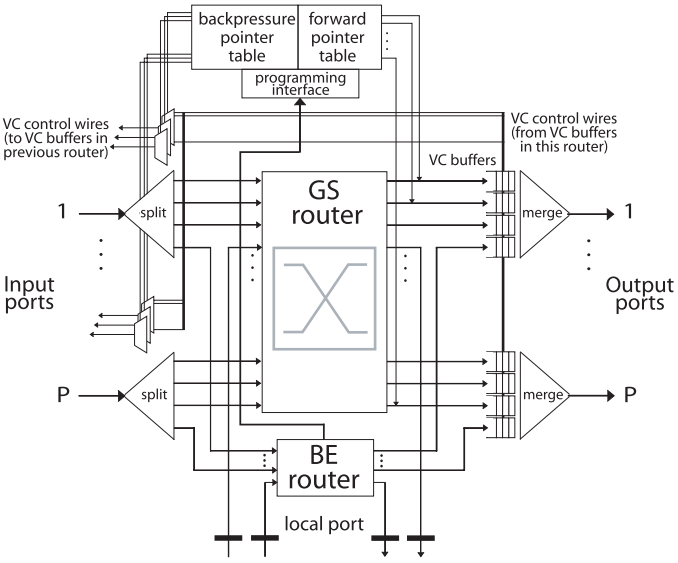

Figure 5. A MANGO router.

used to index the NA routing lookup table [4]. This table maps the global address (8 MSBs of the OCP address field, *MAddr*) to a routing path, as explained in Section 2.

A BE routing path is a hop-by-hop specification of the path through the network. For each hop, a router output port is specified. The BE router module shown in Figure 5 routes the BE packets according to their header. At each hop, the header is rotated, such that the MSBs always specify the next hop. With 5x5 routers, each hop requires 3 bits of routing information. At the final hop, specifying port 0 (the local port) indicates that the destination has been reached. After this follows the router programming bit (*router_pgm_bit*); 0 means that the packet should be used to program the VC hop tables in the router, 1 means it should be routed to the local target NA. If the target NA is the destination, yet another bit follows – the NA programming bit (*NA_pgm_bit*) – indicating whether the packet is meant for programming the NA itself (*NA_pgm_bit* = 0) or whether it is meant as an OCP transaction for the slave IP core attached to the NA (*NA_pgm_bit* = 1). The return path (response e.g. for read transactions) is placed after the forward (request) routing path. Finally, a high bit indicates that the response packet should be routed to the initiator NA and not used to program the router. Thus a complete routing path has the following syntax:

$$BERoutingpath = \\ *[fwd\_hop] :: router\_pgm\_bit :: NA\_pgm\_bit :: *[rtn\_hop] :: 1$$

The BE router could also be implemented such that routing is relative. Instead of hops pointing specifically to an output port, they could be specified as 'go left' or 'go straight', relative to the input port. A forward path then uniquely defines the return path, and the target NA can extract the return path from the header of the incomming packet. This would save header bits. The present scheme however enables higher levels of flexibility, as the return path and the response path can be different, and links in the network do not need to be bi-directional. It requires more bits in the header flit however.

3.3. Guaranteed Service Connections

As explained in Section 2, a GS connection is instantiated by reserving a virtual circuit, a sequence of independently buffered VCs through the network. This is illustrated in Figure 4, where a virtual circuit from the initiator NA to the target NA passes through 3 routers and 2 links. The figure also indicates latencies in different segments of the connection ($L_{initiator}$, L_{hop1}, etc.). These latencies will be used when analyzing the performance of a connection, in Sections 4 and 5. In order to establish a virtual circuit two pointers must be programmed into each router on the path. This is shown in Figure 5 (*forward* and *backpressure* pointer tables), which illustrates a conceptual view of the router architecture. At the VC buffers at the router outputs

a pointer forward on the path, to a VC buffer in the next router, defines the data forwarding channel. This is the *forward* pointer, which uniquely defines an output port in this next router, and a VC on that port. Since flow control is handled on a hop-by-hop basis [6], a flow control channel *backwards* on the path, to the previous VC buffer, must also be established. This is the *backpressure* pointer. At each input VC on the virtual circuit, the *backpressure* pointer selects an output port and a VC on that port. For simplicity Figure 5 only shows 4 VCs on each port, even though the routers in the demonstrator MANGO implement 8 VCs.

The GS router is a non-blocking crossbar. Hence the latency through the router, from the input port to the VC budders on the output, is predictable and bounded.

The pointer tables can be written to the routers by any OCP master in the system, using OCP write commands on connection 0 (the BE connection). To do this a router must be addressed as a slave IP core: a routing path entry must be made in the lookup table of the master's initiator NA, mapping from a global address to a routing path. In this routing path, the router programming bit (see Section 3.2) must be set to 0.

The presented system is constructed of 5x5 routers with 8 VCs on each bidirectional network port, and 5 bits are needed by each pointer. Hereof, 2 bits are used for identifying an output port. Ports are designated 0 to 4, port 0 indicating the local port. A pointer value of 0 however maps to port 4. Routing back on a link is not allowed, and pointing to the same port number as the input, indicates pointing to the local port (port 0, to which the NA – and hence an IP core – is connected). The remaining 3 bits point to one of 8 VCs on each port. Of these, 0 through 6 can be used for GS connections, while the last (VC 7) is used for BE routing. Thus, pointing a GS connection to VC 7 is illegal.

4. Service Guarantees

Each VC on a link is associated with a certain *hop-guarantee*. This is the service guarantee – in terms of latency and bandwidth – in moving flits from the given VC buffer, to a VC buffer on an output in the next router. The end-to-end guarantee on a connection of X hops, $L_{end2end}$ being latency and $BW_{end2end}$ being bandwidth, is determined as:

$$\begin{aligned} L_{end2end} &= L_{hop1} + L_{hop2} + \ldots L_{hopX} \\ BW_{end2end} &= min(BW_{hop1}, BW_{hop2}, \ldots, BW_{hopX}) \end{aligned}$$

The latency is the sum of the per-hop latencies. The latency of the first hop is the time it takes to access a virtual circuit in the first router, while the latencies of the following hops are the link latencies. The bandwidth guarantee is determined by the bottleneck of the path.

The switching of GS flits, inside the routers, is congestion-free, hence apart from a constant element (t_{link}), the hop-guarantees of the connection are determined by the link access guarantee (explained further below). In the presented prototype we use so called ALG (Asynchronous Latency Guarantee) scheduling [7]. While time division multiplexing (TDM) schemes, commonly used for guaranteeing bandwidth on shared links, result in an inverse dependency between latency and bandwidth (the lower the latency required, the more bandwidth must be reserved), the ALG scheduling facilitates a high degree of decoupling of latency and bandwidth guarantees. The access to the link of individual VCs is prioritized. This prioritization together with a special access scheme facilitates a latency guarantee which is linearly proportional to the priority of the given VC. Hence a very low latency guarantee can be given, without the need to also reserve a large portion of the available bandwidth.

In the following, the 8 VCs implemented on each link are denoted $Q \in \{0, 1, \ldots, 7\}$. Since the network is clockless, we specify latency and bandwidth guarantees in terms of the time unit *flit-time* (t_{flit}), which is the cycle time of transmitting one flit on a link. This corresponds to a clock cycle in a synchronous network. In MANGO, t_{flit} depends on the actual link implementation: the link encoding and pipelining, the flit width, etc. We use delay insensitive dual rail encoding [16] in order to improve timing robustness in the system, employ 2-stage pipelining on the links, and the flits are 32-bit wide. This configuration results in t_{flit} = 3.6 ns, in the worst-case process corner.

The hop-guarantees for VC Q are [7]:

$$\begin{aligned} L_{hop} &= ((Q+1) * t_{flit}) + t_{link} \\ BW_{hop} &= \frac{1}{t_{flit} * (N+Q-1)} \end{aligned}$$

Here, N is the total number of VCs on a link and t_{link} is the forwarding latency of a flit from one VC buffer, across the link, through the next router, to the next VC buffer on the connection. VC control [5] ensures that a flit can only gain access to a link, if the target VC buffer is free, hence once access is granted, the flit will experience no congestion. Therefore t_{link} is constant [6]. In the demonstrator network, $t_{link} = 7.9\ ns$. This includes the latency of merging flits from different VCs onto the link, delay insensitive (DI) encoding, the link pipeline latency, DI decoding, the delay of the GS crossbar in the router, and the delay of the VC control circuits and the VC buffers. The latency guarantee is given on a flit by flit basis. The time separation between two consecutive flits determines the bandwidth guarantee. If there are a large number of VCs on a link (large N) this time may become large, since it is dependent on N. Looking at the packet formats in Figure 3, it is seen how non-burst reads, on GS connections, require only a single flit, likewise for non-burst responses and interrupts. This makes them particularly suitable for exploiting the ALG latency guarantees. This will become more clear in Section 5

where we dissect the end-to-end latency of transmitting a packet, into its components.

In the MANGO prototype presented in this work, the links are pipelined. In clocked networks, pipelining has the side effect of increasing the forward latency by one clock cycle per pipeline stage. In clockless pipelines however, the forward latency is only part of the cycle time. Even though we have placed two pipeline stages between each router, the increase in forward latency is only 600 ps, due to the extra pipeline stages. This is not much more than the latency penalty of wire segmentation.

5. Demonstrator and Results

Our test system is composed of three routers, as shown in Figure 4. In order to illustrate the GALS capabilities of MANGO, the master and slave cores are run at different frequencies, 250 MHz and 333 MHz respectively. The master could be a microprocessor, while the slave could be a shared memory. Results are based on simulations of a 0.13 μm standard cell implementation, using worst-case process corner timing parameters. In the following, we first derive connection latency bounds analytically. The latency bound of a connection can be decomposed into local components. We find actual values for these components by performing gate-level simulations. We then calculate the latency bound on two given connections, and finally we verify these analytical results by running gate-level simulations of the complete test system.

The end-to-end latency L_{conn} of using a GS connection for OCP transactions, can be broken into components:

$$\begin{aligned} L_{conn} &= L_{initiator} + L_{fwd} + L_{congestion} \\ &\quad + L_{serialization} + L_{target} \end{aligned}$$

Some of these can be identified in Figure 4. In the following each component is explained.

$L_{initiator}$ is the latency in the initiator NA (the master core's NA). It is one OCP clock cycle plus some forward latency in the clockless part of the NA [4].

L_{fwd} is the forward latency of a flit, in an unloaded network. It is t_{link} for each link traversed, plus a latency for engaging the virtual circuit from the initiator NA, t_{engage}.

$L_{congestion}$ is latency due to stalling in the network. It is the time that a flit is waiting in VC buffers, for access to a link. Its value depends on the link access arbitration.

$L_{serialization}$ is the latency penalty due to serialization of the packetized OCP transactions into flits. This is dictated by the bandwidth guarantee of the connection. As explained in [7], the latency guarantee of ALG is given at the requirement of a time separation between flits on the VC. This is the inverse of the bandwidth guarantee. Hence, in a fully loaded network, the separation between flits will be close to maximum (one over the bandwidth guarantee), while in an uncongested network it will be much lower, as more bandwidth than guaranteed is available.

L_{target} is the forward latency in the target NA. Resynchronization, from the clockless network to the clock of the IP core, takes one OCP clock cycle, while there is also half a cycle latency in the clocked part of the NA. In addition, there is some latency in the clockless part, plus an unknown latency of up to one OCP cycle, due to the uncertainty of the arrival time of the last flit. If this occurs immediately after the local clock tick, the packet will need to wait a complete clock cycle before resynchronization can begin.

By simulating a transaction in an unloaded network, we can measure $L_{initiator}$, L_{fwd} and L_{target}. $L_{serialization}$ is the difference between the arrival time of the first flit and the last (zero for single-flit packets, such as a GS read request). In an unloaded network, $L_{congestion}$ is zero.

Two connections between the master and the slave are tested: *conn1* and *conn2*. The connections consist of three segments: one from the initiator NA to the first VC in the virtual circuit (engage time, t_{engage}), and one across each of the two links in the path. The hop from the initiator NA to the first VC does not require access to a shared link, hence its latency is constant. For *conn1*, low latency VCs have been reserved on the two links that the connection traverses: priority 0 (highest priority) on both links. For *conn2* VCs of priority 3 and 6, i.e. VCs which provide worse latency guarantees, have been reserved.

From gate-level netlist simulations with back-annotated timing, we obtain the values: $t_{engage} = 3.2\ ns$, $t_{flit} = 3.6\ ns$ and $t_{link} = 7.9\ ns$. On the two links between master and slave, having reserved VCs 0 and 0 for *conn1*, and VCs 3 and 6 for *conn2*, the theoretical latency guarantees, of transmitting a flit on the virtual circuits, are thus:

$$\begin{aligned} L_{circuit1} &= t_{engage} + (1+1) * t_{flit} + 2 * t_{link} \\ &= 26.2\ ns \\ L_{circuit2} &= t_{engage} + (4+7) * t_{flit} + 2 * t_{link} \\ &= 58.6\ ns \end{aligned}$$

Note that $L_{circuit} = L_{fwd} + L_{congestion,max}$. The worst case NA latency occurs when the synchronization clock is just missed in the target NA. The worst case serialization penalty for a write (2 flits) is one over the bandwidth guarantee of the connection.

$$\begin{aligned} L_{serialization,conn1} &= t_{flit} * (8+1-1) \\ &= 28.8\ ns \\ L_{serialization,conn2} &= t_{flit} * (8+7-1) \\ &= 50.4\ ns \end{aligned}$$

Now we get the total end-to-end latency guarantee by adding to this the latency of the NAs, the circuit latency and the serialization penalty:

Table 2. Examples of end-to-end latency of a write, on two connections at varying background loads.

	conn1 / 0% load	*conn1* / 100% load	*conn2* / 0% load	*conn2* / 100% load
$L_{initiator}$	4.9 ns	4.9 ns	4.9 ns	4.9 ns
L_{fwd}	17.8 ns	17.8 ns	18.6 ns	18.6 ns
$L_{congestion}$	0 ns	12.3 ns	0 ns	35.0 ns
$L_{serialization}$	12.0 ns	21.3 ns	12.3 ns	25.0 ns
L_{target}	7.0 ns	7.0 ns	7.0 ns	7.0 ns
L_{conn}	41.7 ns	63.3 ns	42.8 ns	90.5 ns

$$
\begin{aligned}
L_{conn1,max} &= L_{initiator} + L_{circuit1} \\
&\quad + L_{serialization} + L_{target} \\
&= 4.9ns + 26.2ns + 28.8ns + 7.5ns \\
&= 67.4ns \\
L_{conn2,max} &= 4.9ns + 58.6ns + 50.4ns + 7.5ns \\
&= 114.2ns
\end{aligned}
$$

In the following we compare these results with a TDM-based NoC with 8 time slots, running at 300 MHz (corresponding to a flit-time of 3.33 ns), with one clock cycle forward latency on the links and one in the routers. To access the time slot allocated to the connection, first a waiting time of up to 8 clock cycles is endured. Adding to this the actual latency through the network of 3 routers plus 2 link (5 cycles total), one gets the circuit delay. The serialization penalty constitutes another 8 cycles, waiting for the reserved time slot to arrive again for the second flit. The worst case latency in the network is:

$$
\begin{aligned}
L_{circuit-TDM} &= 8 * t_{clk} + 5 * t_{clk} \\
&= 43.3\ ns \\
L_{serialization-TDM} &= 8 * t_{clk} \\
&= 26.6\ ns
\end{aligned}
$$

This is comparable to the delays of the MANGO network using the ALG scheduling scheme. However, as the bandwidth granularity is increased, the connection latency of TDM increases fast. Using ALG instead, the latency is independent of the bandwidth reservation. Hence, with regards to latency, TDM is not a scalable solution. For practical examples of using TDM for GS in NoC, please refer to studies made for the Æthereal NoC in [11].

Figure 6 shows end-to-end latencies of issuing OCP write transactions across the tested connections. First GS connections are established between the master and slave IP cores. Then a number of connections are set up between the other network ports, and these are loaded with random traffic. This provides a variable background traffic load, simulating the master/slave subsystem being part of a bigger system. Test results are sampled over sets of 1000 transactions under different background loads. Results are taken

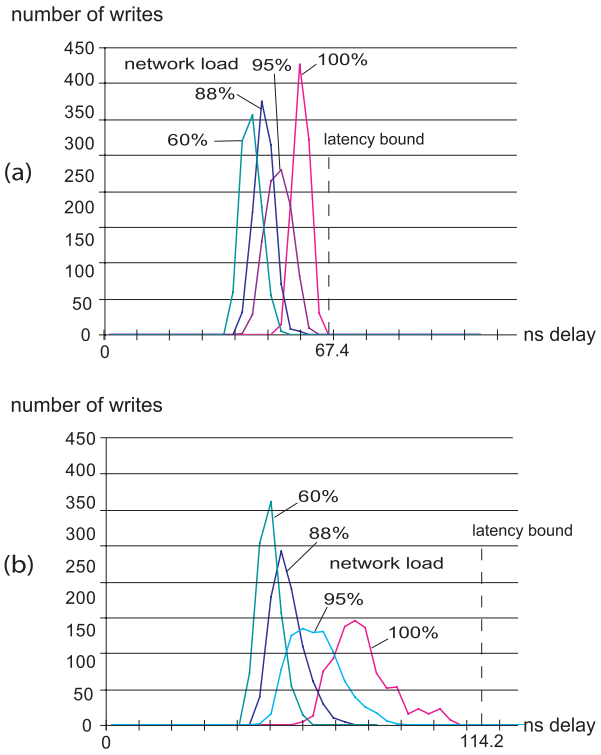

Figure 6. Latency distribution of OCP write transactions on a) connection 1 and b) connection 2.

from gate-level netlist simulations with back-annotated timing. During read transactions, the total latency is the sum of the latency of the request connection, the response time of the slave IP core and the latency of the response connection. Note however (Figure 3) that a GS read request and response both require only a single flit. Hence the serialization penalty is zero, and the transaction latency guarantee is truely decoupled from the bandwidth guarantee. Herein, write commands are sufficient for illustration purposes.

The results illustrate how the latency does not exceed the analytically guaranteed maximum, even under 100% network load. This shows that a high degree of predictability can be obtained in using the shared network, despite a shift

in the use of the network by other entities in the system (different background loads), and despite its asynchronous nature (clockless implementation).

Table 2 shows the breakdown of the latency of two write transactions on each of the two connections; in unloaded and fully loaded network scenarios. Note that these are meassured examples from typical traffic scenarios, i.e. not worst case. It is seen that the forward latency (L_{fwd}) is not exactly the same on the two connections, as one might expect. This is because the clockless implementation of the link access circuits is not symmetrical, since some VCs must be prioritized over others [7]. We see how the effect of $L_{congestion}$ is much smaller on connection 1 which has reserved low latency VCs. We also see that the serialization penalty is quite high (GS writes consist of two flits), up to 34% of the total latency on *conn1* under 100% network load. Note that since the tested connections only traverse two links, the serialization penalty contributes a relatively large portion of the total. This value is independent of the number of hops on a connection, and will dwindle relatively on longer connections. The latency due to congestion is per-link on the other hand, hence it will accumulate on longer connections. The benefit of scheduling for hard latency guarantees in the link access thus increases as the connections get longer. Since ALG facilitates a very low per-link latency, by reserving high priority VCs, it is possible to maintain a low forward latency, even on long connections. The flexibility of the guarantees provided by ALG, compared with those of traditional TDM-based schemes, makes it benefical in scaling, heterogeneous systems.

6. Conclusion

In this paper, we have demonstrated the programmability and end-to-end performance of guaranteed service connections in the MANGO NoC. We have described a programming model for setting up connections in the network, and shown how a tightly bound predictability of the latency can be obtained in issuing OCP commands, despite different levels of background traffic and despite the clockless implementation of the network. Such predictability is important in a modular system-on-chip design flow, as it facilitates analytical verification, and a decoupling of sub-systems. Also, globally asynchronous locally synchronous (GALS) system composition is enabled, by the network adapters synchronizing the clocked OCP interfaces with the clockless network. The work illustrates the advantages of the architecture, from a core-centric point-of-view.

References

[1] International technology roadmap for semiconductors (ITRS) 2003. Technical report, International Technology Roadmap for Semiconductors, 2003.

[2] Open Core Protocol Specification, Release 2.0. www.ocpip.org, 2003.

[3] L. Benini and G. D. Micheli. Networks on chips: A new SoC paradigm. *IEEE Computer*, 35(1):70 – 78, January 2002.

[4] T. Bjerregaard, S. Mahadevan, R. G. Olsen, and J. Sparsø. An OCP compliant network adapter for GALS-based SoC design using the MANGO network-on-chip. In *Proceedings of International Symposium on System-on-Chip 2005*. IEEE, 2005.

[5] T. Bjerregaard and J. Sparsø. Virtual channel designs for guaranteeing bandwidth in asynchronous network-on-chip. In *Proceedings of the IEEE Norchip Conference (NORCHIP 2004)*. IEEE, 2004.

[6] T. Bjerregaard and J. Sparsø. A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip. In *Proceedings of Design, Automation and Testing in Europe Conference 2005 (DATE05)*. IEEE, 2005.

[7] T. Bjerregaard and J. Sparsø. A scheduling discipline for latency and bandwidth guarantees in asynchronous network-on-chip. In *Proceedings of the 11th IEEE International Symposium on Advanced Research in Asynchronous Circuits and Systems*. IEEE, 2005.

[8] D. Chapiro. *Globally-Asynchronous Locally-Synchronous Systems*. PhD thesis, Stanford University, 1984.

[9] W. J. Dally and B. Towles. Route packets, not wires: On-chip interconnection networks. In *Proceedings of the 38th Design Automation Conference*, pages 684–689, June 2001.

[10] J. Dielissen, A. Rădulescu, K. Goossens, and E. Rijpkema. Concepts and implementation of the Philips network-on-chip. In *Proceedings of the international workshop on IP-Based SOC Design (IPSOC 2003)*, Nov. 2003.

[11] K. Goossens, J. Dielissen, O. P. Gangwal, S. G. Pestana, A. Radulescu, and E. Rijpkema. A design flow for application-specific networks on chip with guaranteed performance to accelerate SoC design and verification. In *Proceedings of the Design, Automation and Test in Europe Conference (DATE 2005)*, pages 1182–1187. IEEE, 2005.

[12] R. Ho, K. W. Mai, and M. A. Horowitz. The future of wires. *Proceedings of the IEEE*, 89(4):490 – 504, April 2001.

[13] A. Jantsch and H. Tenhunen. *Networks on Chip*. Kluwer Academic Publishers, 2003.

[14] M. MillBerg, E. Nilsson, R. Thid, and A. Jantsch. Guaranteed bandwidth using looped containers in temporally disjoint networks within the Nostrum network on chip. In *Proceedings of the Design, Automation and Testing in Europe Conference (DATE 2004)*. IEEE, 2004.

[15] J. Muttersbach, T. Villiger, and W. Fichtner. Practical design of globally-asynchronous locally-synchronous systems. In *Proceedings of the Sixth International Symposium on Advanced Research in Asynchronous Circuits and Systems, 2000 (ASYNC 2000)*, pages 52–59. IEEE Computer society, April 2000.

[16] J. Sparsø and S. Furber. *Principles of Asynchronous Circuit Design - a Systems Perspective*. Kluwer Academic Publishers, Boston, 2001.