Real-time Communication for Network on Chip

1 Introduction

As a network on chip gets larger, the communication on the network becomes very complicated congestions on that network can occur resulting in unpredictable delay in communications between network nodes. This non-deterministic behavior makes it very difficult to very verify the timing semantics of a network on chip system, as a result, network on chip systems become unsuitable for real-time applications, especially critical control applications.

Our purpose is to bring the real-time communication to network on chip so that users can always guarantee deterministic behaviors of a critical system. To do so, it requires bounded delay communications between nodes in the network. For example, when processing element A wants to send a critical control packet p to processing element I, how can it know for sure that the packet will reach the destination within a certain amount of time ?

2 Related Work

2.1 Æthereal

This work [5] is from NXP. This architecture tries to avoid contention using contention-free routing by *delaying* some packets.

Guaranteed packets are multiplexed using TDMA. However, this approach requires design-time configuration and verification, which is not flexible for architectures like multicore.

2.2 SoCBUS

SoCBUS architecture [10] is from Linkoping University. It seeks to guarantee real-time properties by setting up a path before ending :

- Init a path by sending a setting up packet.
- The path will be *blocked* until all data have been sent.
- After that the path is unlocked

This approach has the following problems :

- What happens if we have two real-time paths on the same link ?
- Other traffic sharing a link with this real-time path is blocked while data is sent. This seems to be a good solution when sending a large bulk of data but not good for periodic, un-continuous flows.
- Utilization of this approach can be low if real-time flows only require low bandwidth.

2.3 QNoC

This work is from Technion [2]. Packets are sent synchronously. This architecture supports multi service levels as in table 1.

Service-Level	Description	Priority
Signaling	Urgent Messages, Short Packets, Interrupts, Control signals requiring low transport latency	Highest
Real-Time	Real-time and streaming packets	
RD/WR	Short memory and register access	
Block Transfer	Long messages and blocks of data	Lowest

Tab. 1 – QNoC Service Levels

This seems to be a good approach for soft real-time applications like video streaming but this is not really suited for hard real-time applications since what happens when multiple real-time flows have to share the same link : :

– Non-deterministic behaviors for flows.
– Signaling packets can block real-time packets.

To solve this problem, we need to keep track of the number and specifications of real-time flows on a link to make sure that the link is never overloaded.

3 Our Idea

We can achieve the real-time communication betweens some nodes in a network on a chip by borrowing the resource reservation idea [12] from the Internet to the network on chip. In that, all the real-time communications have to be previously reserved on the network.

Real-time flows can be multiplexed [4, 11] on links in networks without violating real-time requirements. Other best-effort flows can still use remaining bandwidth on reserved links.

An admission control mechanism is implemented, thus when a reservation for a real-time flow is initiated by a sender in a network, the network will determine if it can accept that reservation or not based on its current state of other reservations of other real-time flows on the network.

We will design an architecture with the following advantages :

– Multiple real-time flows can be multiplexed on one link [4].
– Utilize the spatial data paths between sources and nodes to avoid the conflicts between real-time flows.
– Does not block links completely as SoCBUS [10], best-effort flows still can travel links used by real-time flows.
– Avoid unpredicted behaviors networks as in QNoC [2], when there are multiple real-time flows suddenly travelling on the same link and their total bandwidth exceeds the bandwidth of the link. The admission control

in our architecture can prevent that. Senders should always know if their required specifications for their communications can be met or not.
– Provide a reconfigurable state for real-time flows on a network, we do not need to pre-calculate that at design time as in Æthereal [5], which is really not suitable for the multi-core architecture.

4 Formal Definitions

– T_i is the *minimum* packet interval time on a real-time flow i.
– L_i is the *maximum* length of a packet in flits of real-time flow i.
– S_{fe} is the service time for a packet in a node including header processing, transmission time, and any other operations. S is often a function of L : $S = f(L)$. In some simple router model (non-pipeline), se can set $S = L$, this means that it takes 1 cycle for a router to process one flit. In pipeline router model as in [7, 6], we have to add some pipeline stages to L to have S.
– $T \subseteq \mathbf{N}$ is the set of the min interval values between packets of flows.
– $L \subseteq \mathbf{N}$ is the set of packet lengths.
– $D \subseteq \mathbf{N}$ is the set of packet delays of real-time flows.
– $\mathcal{V} \subseteq \mathbf{N}$ is the set of virtual channels between pairs of nodes.
– D_{fe} is the *maximum* delay of flow f on link e.
– C is the set of *cores* on the on-chip network.
– R is the set of *routers*.

Then the set of *nodes* on the network is defined as :

$$V = C \cup R \tag{1}$$

And E defined as :

$$E \subseteq V \times V \tag{2}$$

is the set of *directed edges* between nodes in the network.

The set of *flows* on the network is defined as :

$$F \subseteq C \times C \times \mathcal{V} \tag{3}$$

And P is a *mapping* between a real-time flows with its specifications :

$$P : F \rightarrow \tau \times T \times L \times D \tag{4}$$

in which τ is the set of flit types.

$$y_{fe} = \begin{cases} 1 \text{ if flow } f \in F \text{ uses link } e \in E \\ 0 \text{ otherwise} \end{cases} \tag{5}$$

$$I_{ve} = \begin{cases} 1 \text{ if } e \in E \text{ is outgoing edge from } v \in V \\ -1 \text{ if } e \in E \text{ is incoming edge from } v \in V \\ 0 \text{ otherwise} \end{cases} \tag{6}$$

$\forall f = (s, d, id) \in F$ let b be a vector s.t. $b_f(s) = 1$, $b_f(d) = -1$ and $b_f(i) = 0, \forall i \in C$ and $i \neq s, d$, then we have $Iy_f = b_f$, this condition is for unique path between s and d.

Configuration $\mathcal{C}(F) = (P, \{y_f\}_{f \in F})$.

5 Architecture

At each node in a router, we employ the Internet stack to each node in the network on chip.

Application layer at processing units
Transport layer at processing units
Network layer at routers
Data link layer at routers
Physical layer

TAB. 2 – Network stack model

When in need of a real-time communication, the processing unit at each node will issue a request for setting up a path :

Setuppath.request(source_id, destination_id, T, L, D, Hard/Soft) ;

This request will be sent over the network by a PATH message. Its demand has to be met at each node on the network before reaching the destination node.

The destination node will then send back an ACCEPT message to the source saying that the path has been set up. After that, the source can start sending data at the specified rate.

If the set up request cannot reach the destination, the source then will receive a REJECT message from an intermediate node. The source can receive in the reject message about the lower possible rate and delay. It then can try another specification for the path.

So the setting up path response can be : ACCEPT, REJECT

Questions :

- Should we specify the path directly in this request ? Static initialization can be useful in some programming model like Giotto.
- Is it possible that the acceptance message will be blocked on the network or it reaches the source too late ? We should give priority to such kind of control message.
- Real-time flows block other best-effort packets traveling on the link when the utilization of all the packets on the link is 1. Need a mechanism to reroute all best-effort packets.

To reduce the size of each router, we can use processing elements to do complicated tasks like calculating admission criteria for a new flow at each node and rerouting for finding a new suitable path on a network.

At data link layer, heterogeneity communications, *synchronous* communications for best effort packets and *asynchronous* communications for real-time packets since the data buffers for real-time communications can be bounded [4].

6 Theoretical Foundations

6.1 Configuration

A valid configuration $\mathcal{C}(F) \models (7)(8)(9)$

$$Iy_f = b_f \tag{7}$$

The total delay at each hop of a real-time flow has to satisfy the delay constraint of the real-time flow.

$$\sum_{e \in E} D_{fe} y_{fe} \leq P(f)(4), \forall f \in F \tag{8}$$

When multiple real-time flows share the same link, the following conditions have to be met on each shared link [4, 9] :

$$\sum_{f \in F} \frac{S_{fe}}{P(f)(2)} y_{fe} \leq 1, \forall e \in E \tag{9}$$

Since S is a function of the packet length, for a simple (non-pipeline) router model, we often have $S = P(f)(3)$, then (9) becomes :

$$\sum_{f \in F} \frac{P(f)(3)}{P(f)(2)} y_{fe} \leq 1, \forall e \in E \tag{10}$$

One interesting characteristic of this scheme is that the buffer for each real-time flow at each node is bounded, and thus we can send real-time packet *asynchronously* resulting in better bandwidth since we do not have to send *acknowledgement* for each flit sent.

At each node, we use Earliest Deadline First (EDF) [9] to schedule packets, the deadline for each packet at node n is computed as follows :

$$dl_n = t_0 + \sum_{k=1}^{n} D_{i,k} + P_n \tag{11}$$

where t_0 is the time the packet is sent from the source and P_n is the propagation delay from the source till node n (communication time). We can set $P_0 = 0$ since propagation on a network on a chip is immediate.

6.2 Delay Model

From [4], for a simple case, if we assume that for all K real-time flows going through a node n satisfying the following condition :

$$T_i \geq \sum_{j=1}^{K} S_{j,n}, \forall i = 1, ..., K \tag{12}$$

the (12) can be expressed in the flowing way :

$$T_f y_{fe} \geq \sum_{f' \in F} S_{f'} y_{f'e}, \forall e \in E \tag{13}$$

or

$$P(f)(2) y_{fe} \geq \sum_{f' \in F} S_{f'} y_{f'e}, \forall e \in E \tag{14}$$

This means that no deadlines for subsequent packets of a real-time flow will fall within the interval between time 0 and $\sum S = \sum_{j=1}^{K} S_{j,n}$. And if we can *order* the sending time of flows through node n by deadlines if packets from the K flows arrive at the same time then the *minimum* delay of flow i^{th} at node n is :

$$D_{i,n} = \sum_{j=1}^{i} S_{j,n} + S_{max} \forall i = 1, ..., K \tag{15}$$

6.3 Delay in Cut-through Networks

However, the above equation is for store-and-forward networks [1] (a packet has to be received *completely* by a router before it can be forwarded), in the cut-through networks (a packet can be sent while being received), if the delay at each node of a packet is defined as the duration from the header of that packet is received until the header is started to send, then we do not need to include the service time of the packet itself at the node. The *minimum* local delay of a packet at a node can be set by :

$$D_{i,n} = \sum_{j=1}^{i-1} S_{j,n} + S_{max} \forall i = 1, ..., K \tag{16}$$

Otherwise, if we *order* the deadlines of packets of real-time flows coming at the same time by their respective *maximum packet length*, we can write :

$$D_{fe} y_{fe} = \sum_{\forall f^* \in F : L_{f^*} \leq L_f, f^* \neq f} S_{f^*e} y_{f^*e} + S_{max} \tag{17}$$

This means that, flows with smaller maximum packet lengths will be given smaller delays at this node (thereby smaller deadlines when packets come at the same time). This ordering scheme will give the smallest *overall* delay at this node. The path delay from (8) becomes :

$$\sum_{e \in E} D_{fe} y_{fe} + S_f \leq P(f)(4), \forall f \in F \tag{18}$$

since we have to include the service time for the packet at the last node to receive the full packet, not only the header.

For simple router models (non-pipeline), we have $S_f = P(f)(3)$, then (18) becomes :

$$\sum_{e \in E} D_{fe} y_{fe} + P(f)(3) \leq P(f)(4), \forall f \in F \tag{19}$$

From (17) and (19) we have :

$$\sum_{e \in E} (\sum_{\forall f^* \in F : L_{f^*} < L_f, f^* \neq f} S_{f^* e} y_{f^* e} + S_{max}) y_{fe} \leq P(f)(4) - P(f)(3), \forall f \in F \tag{20}$$

Now a valid configuration $\mathcal{C}(F) \models (7)(10)(20)$

6.4 Dynamic Path Establishment and Routing

When a new real-time path needs to be set up with some specifications, we have : $F \rightarrow F \cup \{f^{'}\} = F^{'}$ and $T \rightarrow T^{'}$ s.t. $T^{'}(f) = T(f) \forall f \in F$ and $T^{'}(f^{'}) = (\tau^{'}, T^{'}, L^{'}, D^{'})$.

The problem becomes : Find $y_{f^{'}e}$ s.t. $\mathcal{C}(F^{'})$ is valid when we know that $\mathcal{C}(F)$ is valid.

For a new configuration $\mathcal{C}(F^{'})$, we find a new flow $f^{'}$ such that the conditions (7)(10) are satisfied. And the path delay constraint from (20) is :

$$\sum_{e \in E} (\sum_{\forall f^* \in F : L_{f^*} < L_{f^{'}}, f^* \neq f^{'}} S_{f^* e} y_{f^* e} + S_{max}) y_{f^{'} e} \leq P(f^{'})(4) - P(f^{'})(3), \text{ for } f^{'} \in F \tag{21}$$

If we assume the maximum service time for one packet at a node of a flow is the same for all nodes and again the router model is simple (non-pipeline) then we have :

$$\sum_{e \in E} (\sum_{\forall f^* \in F : L_{f^*} < L_{f^{'}}, f^* \neq f^{'}} P(f^*)(3) y_{f^* e} + S_{max}) y_{f^{'} e} \leq P(f^{'})(4) - P(f^{'})(3), \text{ for } f^{'} \in F \tag{22}$$

To add a new real-time flow like this to the network without modifying the paths of other previous real-time flows, at each node we store a *slack* of delay for each real-time flow. Then whenever we add a new real-time flow and modify the local bounded delay of other flows we check if the increasing delay amounts of the flows exceed the slack of the flows and then recompute new slacks for these flows. The slack of a flow f is computed as :

$$slack_f = P(f)(4) - P(f)(3) - \sum_{e \in E} (\sum_{\forall f^* \in F : L_{f^*} < L_{f^{'}}, f^* \neq f^{'}} P(f^*)(3) y_{f^* e} + S_{max}) y_{f^{'} e} \forall f \in F \tag{23}$$

What we should minimize : power or future paths ?

7 Specification

7.1 Message Structures

The structure of a PATH message (this is often a flit) to set up a real-time flow on the network is :

Type	Source ID	Destination ID	Virtual Channel ID	T_{min}	L_{max}	D_{max}	(optional)

TAB. 3 – Setup path message

The structure of a real-time data packet is :

Type	Virtual Channel ID	Time stamp or jitter	Packet Length	Data
Data flit				
...				
Data flit				

TAB. 4 – Data message

7.2 Considerations

network/ router	Topology	Communication Channel	Routing Strategy	Buffering	Crossbar	Flow Control	VC	VC Selection	QoS Support
Kavaldjiev	2D Mesh	NA	Wormhole Source	Input queue	Full	NA	4	TDM	Yes
QNoC	2D Mesh regular or irregular	16 data bits (parameterizable) + 10 control bits	Wormhole XY	Input queue	Full	Credit based	4	Priority and availability in buffer	Yes
Dally	2D Folded Torus	256 data bits + 38 control bits	Wormhole XY Source	Input queue + 1 output position	Multiplexer	Credit based	8	NA	Yes
Marescaux	2D Torus	16 data bits + 3 control bits	Wormhole XY	2 output positions	Multiplexer	Handshake	2	TDM	Yes
Xpipes	Arbitrary (design time)	32, 64 or 128 bits	Wormhole Street sign	Output queue	Multiplexer	Handshake	Parameterizable	Priority	No
Æthereal	2D Mesh	32 bits	Circuit Switching (GT) e Wormhole Source (BE)	Output queue	NA	NA	3	Priority	Yes
MediaWorm	not applicable	NA	Wormhole	Input and output queue	Multiplexer	NA	2	Virtual Clock	Yes
Hermes NoC	2D Mesh	16 data bits + (parameterizable) 6 control bits	Wormhole XY / partially adaptive	Input queue	Full	Credit based	2 - 4	TDM adaptive	No

From this we can consider the format of a data packet since we have the tradeoff between the length of a packet and the bounded delay of the packet. If the packet is long, then our network is more efficient, however, the bounded delay will probably large.

7.3 Routing

When a reservation path packet reaches a node and the calculated routing link to another router cannot afford the service time and transmission rate or delay bound for that reserving flow. What should the router do :

Options	Advantage	Disadvantage
Backtrack to the source node (router)	Possible find another better path in another direction	Possibly cannot find a path
Backtrack to current node (router) and try another link (direction)	- Possibly to find another path (maybe longer). - Can always find a path if the path exists using exhaust search	- The overhead for finding another path using exhaust search is potentially big. - The algorithm routing can be complicated and expensive if implemented in hardware.

TAB. 5 – Routing Considerations

Furthermore, should we employ routing able mechanism, this means that a table is used at each node to store the information like left utilization $\frac{S}{T}$ at each link and delay bound at each link from source to destination or we can use some dynamic routing protocols like dynamic source routing (DSR) as in wireless networks.

8 Implementation

The router architecture can be implemented as in [8, 11] and extend [7, 6] to have better performance.

We use Noxim [3] as the implementation platform. Currently, we have tried to set up real-time paths and admission control for real-time paths when sharing the same link. The real-time packets currently just contain one flit.

Références

[1] William James Dally and Brian Patrick Towles. Principles and practices of interconnection networks.

[2] Rostislav (Reuven) Dobkin, Ran Ginosar, and Israel Cidon. Qnoc asynchronous router with dynamic virtual channel allocation. In *NOCS '07 : Proceedings of the First International Symposium on Networks-on-Chip*, page 218, Washington, DC, USA, 2007. IEEE Computer Society.

[3] Fabrizio Fazzino, Maurizio Palesi, and Davide Patti. Noxim : Network-on-chip simulator.

[4] Domenico Ferrari and Dinesh C. Verma. A scheme for real-time channel establishment in wide-area networks. *IEEE Journal on Selected Areas in Communications*, 8 :368–379, 1990.

[5] Kees Goossens, John Dielissen, Jef van Meerbergen, Peter Poplavko, Andrei Rădulescu, Edwin Rijpkema, Erwin Waterlander, and Paul Wielage. Guaranteeing the quality of services in networks on chip. pages 61–82, 2003.

[6] Li-Shiuan Peh and William J. Dally. A delay model and speculative architecture for pipelined routers. In *HPCA '01 : Proceedings of the 7th International Symposium on High-Performance Computer Architecture*, page 255, Washington, DC, USA, 2001. IEEE Computer Society.

[7] Li-Shiuan Peh and William J. Dally. A delay model for router microarchitectures. *IEEE Micro*, 21(1) :26–34, 2001.

[8] Jennifer Rexford, John Hall, and Kang G. Shin. A router architecture for real-time communication in multicomputer networks. *IEEE Transactions on Computers*, 47 :1088–1101, 1998.

[9] Dinesh C. Verma, Hui Zhang, and Domenico Ferrari. In proceedings of tricomm '91 delay jitter control for real-time communication in a packet switching network, 1991.

[10] Daniel Wiklund and Dake Liu. Socbus : Switched network on chip for hard real time embedded systems. In *IPDPS '03 : Proceedings of the 17th International Symposium on Parallel and Distributed Processing*, page 78.1, Washington, DC, USA, 2003. IEEE Computer Society.

[11] Hui Zhang. 1 service disciplines for guaranteed performance service in packet-switching networks. *Proceedings of the IEEE*, 10 :1374–1396, 1995.

[12] Lixia Zhang, Steve Deering, Deborah Estrin, and Scott Shenker. Rsvp : A new resource reservation protocol. *IEEE Network*, 7 :8–18, 1993.