Trade Offs in the Design of a Router with Both Guaranteed and Best-Effort Services for Networks on Chip

E. Rijpkema, K.G.W. Goossens, A. Rădulescu,
J. Dielissen, J. van Meerbergen, P. Wielage, and E. Waterlander
Philips Research Laboratories, Eindhoven, The Netherlands

Abstract

*Managing the complexity of designing chips containing billions of transistors requires decoupling computation from communication. For the communication, scalable and compositional interconnects, such as networks on chip (NoC), must be used. In this paper we show that guaranteed services are essential in achieving this decoupling. Guarantees typically come at the cost of lower resource utilization. To avoid this, they must be used in combination with best-effort services. The key element of our NoC is a router consisting conceptually of two parts: the so-called guaranteed throughput (*GT*) and best-effort (*BE*) routers. We combine the* GT *and* BE *router architectures in an efficient implementation by sharing resources. We show the trade offs between hardware complexity and efficiency of the combined router, and motivate our choices. Our reasoning for the trade offs is validated with a prototype router implementation. We show a lay-out of an input-queued wormhole 5×5 router with an aggregate bandwidth of 80 Gbit/s. It occupies 0.26 mm^2 in* CMOS12*. This shows that our router provides high performance at reasonable cost, bringing NoCs one step closer.*

1 Introduction

Recent advances in technology raise the challenge of managing the complexity of designing chips containing billions of transistors. A key ingredient in tackling this challenge is *decoupling the computation from communication* [10,14]. This decoupling allows IPs (the computation part), and the interconnect (the communication part) to be designed independently from each other.

In this paper, we focus on the communication part. Existing interconnects (e.g., buses) may no longer be feasible for chips with many IPs, because of the diverse and dynamic communication requirements. *Networks on a chip* (NoC) are emerging as an alternative to existing on-chip interconnects because they (a) structure and manage global wires in new deep-submicron technologies [2–5,7], (b) share wires, lowering their number and increasing their utilization [5, 7], (c) can be energy efficient and reliable [3], and (d) are scalable when compared to traditional buses [8].

Decoupling the computation from communication requires that the *services* that IPs use to communicate are well-defined, and hide the implementation details of the interconnect [10], Figure 1(a). NoCs help, because they are traditionally designed using layered protocol stacks [13], where each layer provides a well-defined interface which decouples service usage from service implementation [4, 14], see Figure 1(b).

In particular, *guaranteed services* are essential because they make the requirements on the NoC explicit, and limit the possible interactions of IPs with the communication environment. IPs can also be designed independently, because their use of guaranteed services is not affected by the interconnect or by other IPs. This is essential for a compositional construction (design and programming) of systems on chip (SoC). Moreover, failures are restricted to the IP configuration phase (a service request is either granted or denied by the NoC) which simplifies the IP programming model [7]. We view the guaranteed services to be offered by an interconnect as a requirement from the applications, see Figure 1(c).

The drawback of using guaranteed services is that they require resource reservations for worst-case scenarios. This is not acceptable in a SoC where cost constraints are typically very tight, see Figure 1(d). Therefore, we also provide *best-effort services* to exploit the network capacity that is left over, or reserved but unused. Guaranteed services are then used for the critical (e.g. real-time) traffic, and best-effort services for non-critical communication.

The combination of guaranteed and best-effort classes is known from general computer network research [15], but not for on-chip networks. As on- and off-chip networks have different characteristics, the trade offs in their design are different. In this paper, we present the trade offs between hardware complexity and efficiency for networks on chip, and motivate our choices.

We present a prototype router architecture that reflects one particular set of design choices. It has an aggregate bandwidth of 80 Gbit/s, and its CMOS12 lay-out occupies 0.26 mm^2. We list other feasible variations that either increase performance, or lower the router cost.

In this paper, we first list a set of network-independent communication services that are essential in chip design (Section 2). Then, we show the trade-offs between efficiency and cost that we make in our NoC. In Section 3, we present some general network-related issues that are used in the sections to follow. In Section 4, we zoom into the internals of the key component of our NoC: a router that efficiently provides both guaranteed and best-effort services. In Section 5, we demonstrate the feasibility of our router design through a prototype implementation in CMOS12.

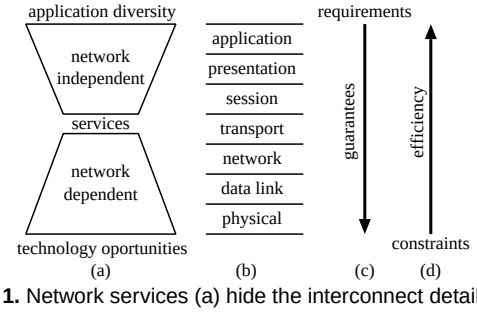


Figure 1. Network services (a) hide the interconnect details and allow reusable components to be build on top of them, (b) are build using a layered approach (c) are driven by the application requirements, and (d) their efficiency relies on technology and network organization.

1530-1591/03 $17.00 © 2003 IEEE

2 Services

The NoC services that we consider essential for chip design are: *data integrity*, meaning that data is delivered uncorrupted, *lossless data delivery*, which means no data is dropped in the interconnect, *in-order data delivery*, which specifies that the order in which data is delivered is the same order in which it has been sent, and *throughput* and *latency* services that offer time-related bounds. As shown in Section 1, guaranteed services are essential to simplify IP design and integration. With the current technology, we assume data integrity is solved at the data-link layer. All the other services can be guaranteed or not on request. In the next section, we describe briefly how these services are provided by our NoC, and in Section 4 we describe how our router architecture enables an efficient implementation of these services.

Guaranteed services require resource reservation for worst-case scenarios, which can be expensive. For example, guaranteeing throughput for a stream of data implies reserving bandwidth for its peak throughput, even when its average is much lower. As a consequence, when using guarantees, resources are often underutilized.

Best-effort services do not reserve any resources, and hence provide no guarantees. Best-effort services use resources well because they are typically designed for average-case scenarios instead of worst-case scenarios. They are also easy and fast to use, as they require no resource reservation. Their main disadvantage is their unpredictability: one cannot rely on a given performance (i.e., they do not offer guarantees). In the best case, if certain boundary conditions are assumed, a statistical performance can be derived.

The requirements for guaranteed services and the efficiency constraint (i.e., good resource utilization) are conflicting. Our approach to a predictable and low-cost interconnect is to integrate the guaranteed and best-effort services in the same interconnect. Guaranteed services would be used for critical traffic, and best-effort services for non-critical traffic. For example a video processing IP will typically require a lossless, in-order video stream with guaranteed throughput, but possibly allows corrupted samples. Another example is cache updates which require uncorrupted, lossless, low-latency data transfer, but ordering and guaranteed throughput are less important. In Section 4.3 we show how integrated guaranteed and best-effort services efficiently can use common resources. In the remainder of this section we analyze the minimum level of abstraction at which the communication services must be offered to hide the network internals.

Traditionally, network services have been implemented and offered using a layered protocol stack, typically aligned to the ISO-OSI reference model [13], see Figure 1(b). NoCs also take this approach [3, 4, 7, 14], because it structures and decomposes the service implementation, and the protocol stack concepts aid positioning of services.

To achieve the decoupling of computation from communication, the communication services must be offered at least at the level of the transport layer in OSI reference model. It is the first layer that offers end-to-end services, hiding the network details; see Figure 1(a, b) [4].

The lowest three layers in the protocol stack, namely physical, data-link, and network layers, are network specific. Therefore, these services should not be visible to the IPs when decoupling between computation from communication is desired. However, these layers are essential in implementing the services, because constructing guarantees without guarantees at the layer below is either very expensive, or even impossible. For example, implementing a lossless communication on top of a lossy service requires acknowledgment, data retransmission, and filtering duplicated data. This leads to an increase in traffic, and possibly larger buffer space requirements. Even worse, providing guarantees for time-related services is impossible if lower layers do not offer these guarantees. For example, latency can not be guaranteed if communication at a lower layer is lossy. As a consequence, guarantees can only be built *on top* of guarantees, see Figure 1(c). Similarly, a layer's efficiency is based on efficient implementations of the layers below it, see Figure 1(d).

3 Networks on chip

General computer network research is a mature research field [15] which has many issues in common with NoCs. However, two significant differences between computer networks and on-chip networks make the trade offs in their design very different [5]. First, routers of a NoC are more resource constrained than those in computer network, in particular in the control complexity and in the amount of memory. Second, communication links of a NoC are relatively shorter than those in computer networks, allowing tight synchronization between routers.

We identify three important issues in the design of the router network architecture. These are: the *switching mode*, *contention resolution*, and *network flow control*. Equally important, *end-to-end flow control* and *congestion control* are handled in our NoC at the network edge instead of the routers; we therefore omit their discussion here. Similarly, we assume guaranteed data integrity at the link level and retain it at the network layer and and higher.

3.1 Switching mode

The *switching mode* of a network specifies how data and control are related. We distinguish *circuit switching* and *packet switching*. In circuit switching data and control are separated. The control is provided to the network to *set up* a *connection*. This results in a *circuit* over which all subsequent data of the connection is transported. In *time-division circuit switching* bandwidth is shared by time-division multiplexing connections over circuits. Circuit-switched networks inherently offer time-related guaranteed services after resources are reserved during the connection set up.

In *packet switching* data is divided into *packets* and every packet is composed of a control part, the *header*, and a data part, the *payload*. Network routers inspect, and possibly modify, the headers of incoming packets to switch the packet to the appropriate output port. Since in packet switching the packets are self contained, there is no need for a set-up phase to allocate resources. Therefore, best-effort services are naturally provided by packet switching.

3.2 Contention resolution

When a router attempts to send multiple data items over the same link at the same time *contention* is said to occur. As only one data item can be sent over a link at any point in time, a selection among the contending data must be made; this process is called contention resolution.

In circuit switching, contention resolution takes place at set up at the granularity of connections, so that data sent over different connections do not conflict. Thus, there is no contention during data transport, and time-related guarantees can be given.

In packet switching contention resolution takes place at the granularity of individual packets. Because packet arrival cannot be predicted contention can not be avoided. It is resolved dynamically by scheduling in which data items are sent in turn. This requires data storage in the router, see Section 4.2.1, and delays the data in a non predictable manner which complicates the provision of guarantees, see Section 4.1.1.

3.3 Network flow control

Network flow control, also called *routing mode*, addresses the limited amount of buffering in routers and data acceptance between routers. In circuit switching connections are set up. The data sent over these connections is always accepted by the routers and hence no network flow control is needed. In packet switching, data items must be buffered at every router before they are sent on. Because routers have a limited amount of buffering they accept data only when they have enough space to store the incoming data.

There are three types of network flow control, namely *store-and-forward*, *virtual cut-through*, and *wormhole* routing. In store-and-forward routing, an incoming packet is received and stored in its entirety before it is forwarded to the next router. This requires storage for the complete packet, and implies a per-router latency of at least the time required for the router to receive the packet.

In virtual cut-through routing a packet is forwarded as soon as the next router guarantees that the complete packet will be accepted. When no guarantee is given, the router must be able to store the whole packet. Thus, virtual cut-trough routing requires buffer space for a complete packet, like store-and-forward routing, but allows lower-latency communication.

In wormhole routing packets are split in so-called *flits* (flow control digits). A flit is passed to the next router when the flit can be accepted, even when there is not enough buffer space for the complete packet. As soon as a flit of a packet is sent over an output port, that output port is reserved for flits of that packet only. When the first flit of a packet is blocked the trailing flits can therefore be spread over multiple routers, blocking the intermediate links. Wormhole routing requires the least buffering (buffer flits instead of packets) and also allows low-latency communication. However, it is more sensitive to deadlock and generally results in lower link utilization than virtual cut-through routing.

To allow low latency we consider both virtual-cut through and wormhole routing, which are both feasible in terms of buffer area, as shown in Section 5.

4 A combined GT-BE router

Section 2 defines our requirements for NoCs in terms of services that are to be offered, in particular, both guaranteed and best-effort services. Using the general network issues of the previous section we show in the following two subsections that the guaranteed and best-effort services can conceptually be described by two independent router architectures. The combination of these two router architectures is efficient and has a flexible programming model, as described in Section 4.3. Section 5 then shows a prototype implementation.

4.1 A GT router architecture

Our guaranteed-throughput (GT) router guarantees uncorrupted, lossless, and ordered data transfer, and both latency and throughput over a finite time interval. As mentioned earlier, data integrity is solved at the data-link layer; we do not address it further. The GT router is lossless because we use a variant of circuit switching, described in the next section. Data is transported in fixed-size blocks. As only one block is stored per input in the GT router, data items remain ordered per connection. We now turn to the more challenging time-related guarantees, namely throughput and latency.

4.1.1 Time-related guarantees

Latency is defined as the duration a packet is transported over the network. Guaranteeing latency, therefore, means that a worst-case upper bound must be given for this time. We define throughput for a given producer-consumer pair as the amount of data transported by the network over a finite, fixed time interval. Guaranteeing throughput means giving a lower bound.

We observe that guaranteeing latency even in a lossless router is difficult because contention requires scheduling and hence cause delays. Guaranteeing throughput is less problematic. Rate-based packet switching (for an overview see [16]) offers guaranteed throughput over a finite period, and hence a latency bound. This bound is very high, however, and the cost of buffering is also high. Deadline-based packet switching [12] offers preferential treatment for packets close to their deadline. This allows differential latency guarantees (under certain admissible traffic assumptions), but also at high buffer costs.

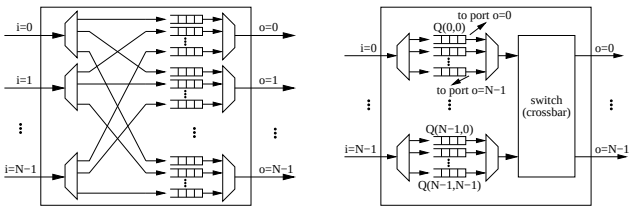
Circuit switching solves the contention at set up, so naturally providing guaranteed latency and throughput. Circuits can be pipelined to improve throughput [6], at the cost of additional buffering and latency. Time-division multiplexing connections over pipelined circuits additionally offers flexibility in bandwidth allocation. This requires a logical notion of router synchronicity, which is possible because a NoC is better controllable than a general network. We explain this variation in more detail in the next subsection. The associated programming model is described in Section 4.3.2.

4.1.2 Contention-free routing

A router uses a *slot table* to (a) avoid contention on a link, (b) divide up bandwidth per link between connections, and (c) switch data to the correct output. Every slot table T has S time slots (rows), and N router outputs (columns). There is a logical notion of synchronicity: all routers in the network are in the same fixed-duration slot. In a slot s at most one *block* of data can be read/written per input/output port. In the next slot $(s+1)\%S$, the read blocks are written to their appropriate output ports. Blocks thus propagate in a store and forward fashion. The latency a block incurs per router is equal to the duration of a slot and bandwidth is guaranteed in multiples of block size per S slots.

The entries of the slot table map outputs to inputs for every slot: $T(s, o) = i$. An entry is empty, when there is no reservation for that output in that slot. No contention arises because there is at most one input per output. Sending a single input to multiple outputs (multicast) is possible.

The slots reserved for a block along its path from source to destination increase by one (modulo S). If slot s is reserved in a router, slot $(s + 1)\%S$ must be reserved in the next router on the path. The assignment of slots to connections in the network is an optimization problem, and is described in Section 4.3.3. Section 4.3.2 explains how slots are reserved in our network.



(a) output queued architecture (b) virtual output queued architecture

Figure 2. Schematics of two router architectures.

4.2 A BE router architecture

Best-effort traffic can have a better *average* performance than offered by guaranteed services. This depends on boundary conditions, such as network load, that are unpredictable. Best-effort services thus fulfill our efficiency requirement, but without offering time-related guarantees. This section describes an architecture for a best-effort service with uncorrupted, lossless, in-order data transport.

The BE router cost and performance are largely dependent on the contention resolution scheme of the router. The contention resolution scheme has two components: buffering and scheduling. The main trade off in Section 4.2.1 is between total buffer size, buffering strategy, and link utilization. Without taking global network requirements into account, no decisions will be made, rather we present a router that allows different instances, to trade off hardware complexity for link utilization at instantiation time. In Section 4.2.2 the trade off is between link utilization and schedule complexity and we select an efficient scheduling algorithm that is easily specialized to the different instances.

4.2.1 Buffering strategy

The buffering strategy determines the location of buffers inside the router. We distinguish *output queuing* and *input queuing*. In the following, N is the number of inputs, equal to the number of outputs, of our router. In output queuing N^2 queues are located at the outputs of the router as in Figure 2(a). From the inputs to the outputs there is a fully connected bipartite interconnect to allow every input to write to every output. Output queuing has the best performance among the buffering strategies, however, the interconnect will make the router wire dominated and expensive already for small values of N.

In input queuing the queues are at the input of the router. A scheduler determines at which times which queues are connected to which output ports such that no contention occurs. The scheduler derives contention-free connections, a switch matrix (crossbar switch) can be used to implement the connections. In traditional input queuing, or input queuing for short, there is a single queue per input, resulting in a buffer cost of N queues per router. However, due to the so-called *head-of-line blocking*, for large N, router utilization saturates at 59% [9]. Therefore, input queuing results in weak utilization of the links.

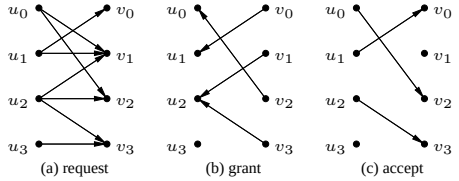
Another version of input queuing is *virtual output queuing* (VOQ) [1]. VOQ combines the advantages of input queuing and output queuing. It has a switch like in input queuing and has the link utilization close to that of output queuing; 100% link utilization can still be achieved, when N is large [11]. As for output queuing, there are N^2 queues. For every input i there are N queues $Q(i, o)$, one for each output o, see Figure 2(b). Typically the set of N queues at each input port of a VOQ router are mapped onto a single RAM. However, for NoCs we strive at a small router and therefore we require the RAMs to have few addresses. But such RAMs have large overhead. Therefore, we use in-house developed dedicated fifos, which have almost no overhead, see Section 5.

The decision to select traditional input queuing or VOQ depends on system-level aspects like topology, network utilization, and global wiring cost, and is outside the scope of this paper. In Section 5 we show a prototype of an input queued router with dedicated hardware fifos and explain that VOQ is a valid option with minor additional cost.

4.2.2 Matrix scheduling

The switch matrix, present in input queued architectures, is controlled by a contention resolution algorithm, known as matrix scheduling, to properly connects inputs to outputs.

The matrix scheduling problem can be modeled as a bipartite graph matching problem. Every input port i is modeled by a node u_i and every output port o by a node v_o. There is an edge between u_i and v_o if and only if queue $Q(i, o)$ is non-empty. A *match* is a subset of these edges such that every node is incident to at most one edge. For example, Figure 3(c) is a match of Figure 3(a).



(a) request (b) grant (c) accept

Figure 3. The three stages of a schedule iteration.

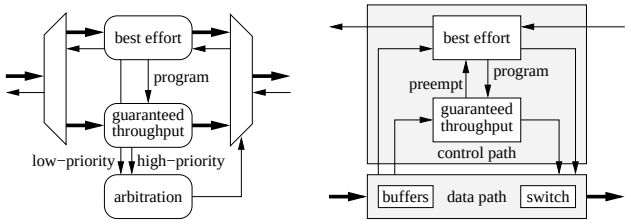
Matching can be done optimally, but because of time complexity and fairness, a non-optimal algorithm is preferred [11].

Our matching algorithm is iterative and one iteration has three stages, illustrated by an example in Figure 3 for $N = 4$. In the first stage, see Figure 3(a), every non-empty queue $Q(i, o)$ *requests* access to output port o from input port i. In the second stage, see Figure 3(b), every output port o *grants* one request, solving link contention at the output ports. In the third stage, see Figure 3(c), every input port i *accepts* one grant, to resolve memory contention at the input port. A next iteration then starts with the matching found so far. This scheme is used in various scheduling algorithms, including parallel iterative matching, round robin matching, and SLIP [11], and applies to both input queuing and VOQ. For input queuing, however, stage (c) in Figure 3 is omitted since on contention on input ports can occur. To keep schedule latency as low as possible we use one iteration only.

4.3 Combining the GT and BE routers

The GT and BE router architectures are combined to share resources, in particular the links and the switch. Moreover, best-effort traffic enables a packet-based programming model for the guaranteed traffic, as shown later, in Section 4.3.2.

The principal constraint for a combined router architecture is that guaranteed services are never affected by best-effort services. Figure 4(a) shows that, conceptually, the combined router contains both router architectures (fat lines represent data, thin lines represent control). Incoming data is switched to either the GT or the BE router. The GT traffic, the traffic that is served by the GT router, has the higher priority, to maintain guarantees. This is ensured by the arbitration unit, which therefore affects the best-effort scheduling. Furthermore, best-effort packets can program the guaranteed router, as shown by the arrow labeled program. Thin lines going from the right to the left indicate network flow control, which



(a) conceptual view (b) hardware view

Figure 4. Two views of the combined GT-BE router.

is only required for best-effort packets because guaranteed blocks never encounter contention.

Figure 4(b) shows that the data path, consisting of buffers and switch matrix, is shared, and that the control paths of the BE and GT routers are separate, yet interrelated. Moreover, the arbitration unit of Figure 4(a) has been absorbed by the BE router. The following subsection shows how this can be done.

4.3.1 Arbitration and flit size

When combining GT and BE traffic in a single network the impact on the network flow control scheme must be taken into account. Recall from Section 3.3 that a BE flit is the smallest unit at which flow control is performed. In other words, the BE scheduling can only react to GT blocks at flit granularity. To avoid alignment problems, the block size (B words) is a multiple of the flits (F words, $B = \ell F$) with ℓ being constant. We prefer a small ℓ to decrease the store-and-forward delay and reduce the buffer size for guaranteed traffic, and a small F for fine-grained switching and better statistical multiplexing.

The router architecture contains a data path and a control path, see Figure 4(b). The data path maximizes throughput for high link utilization, and the control path maximizes the rate of scheduling and switching. They can be designed and optimized independently. Given any combination of their operating frequencies, the router has both maximum throughput and switching rate by using the appropriate flit size F_{opt}. For $F > F_{opt}$, the control path is ready while data is still being transported, lowering the switching rate. For $F < F_{opt}$, flits have been transported before the control path finishes, wasting bandwidth.

We extend the schedule algorithm in Section 4.2.2, to handle the combination of GT and BE traffic. In this combination GT traffic always has priority over BE traffic. This is to ensure that guarantees are never corrupted.

4.3.2 Programming model

In this section we show how GT connections are set up and torn down by means of BE packets to avoid introducing an additional communication infrastructure only to program the network. To ensure scalability, programming must not require a global view or centralized resources. Section 4.1.2 explains why our contention-free routing uses slot tables; we now see that they are distributed over routers for scalability.

Initially the slot table of every router is empty. There are three system packets: SetUp, TearDown, and AckSetUp. They are used to program the slot table in every router on their path. The SetUp packet creates a connection from a source to a destination, and travels in the direction of the data ("downstream"). When a SetUp packet arrives at the destination it is successful and is acknowledged by returning an AckSetUp. TearDown packets destroy (partial) connections, and can travel in either direction. SetUp packets contain the source of the data, the destination or a path to it, and a slot number. Every router along the path of the SetUp packet checks if the output to the next router in the path is free in the slot indicated by the packet. If it is free, the output is reserved in that slot, and the SetUp packet is forwarded with an incremented (modulo S) slot. Otherwise, the SetUp packet is discarded and a TearDown packet returns along the same path. Thus every path must be reversible; this is the only assumption we make about the network topology. These upstream TearDown packets free the slot, and continue with a decremented slot. Downstream TearDown packets work similarly, and remove existing connections. A connection is successfully opened when an AckSetUp is received, else a TearDown is received.

The programming model is pipelined and concurrent (multiple system packets can be active in the network simultaneously, also from the same source) and distributed (active in multiple routers). Given the distributed nature of the programming model, ensuring consistency and determinism is crucial. The outcome of programming may depend on the execution order of system packets, but is always consistent. The next section shows how to use this programming model.

4.3.3 Compile- and run-time slot allocation

This section explains how to determine the slots specified in SetUp packets. A slot allocation for a single connection requires that, at every router along the path, the required output is free (not reserved by another connection) in the appropriate slot. Computing an optimal slot allocation for all connections requires a global network view and may be expensive. To reduce computational cost, heuristics can be used, possibly leading to non-optimal solutions.

SetUp packets of different connections do not fail if connections are set up with conflict-free slots or paths. All execution orders of SetUp packets then give the same result, so that compile-time slot allocations can be recreated deterministically at run time.

Optimal run-time slot allocation is hard without a global (and central) slot table view, which is non-scalable and slows down programming. Distributed run-time slot allocation is scalable, but lacks a global view and is, therefore, perhaps suboptimal. Moreover, SetUp packets may interfere, making programming more involved, and perhaps non-deterministic. However, dynamic connection management at high rates will require distributed slot allocation. In a simple distributed greedy algorithm, all sources repeatedly generate random slot numbers for each set up until their connection succeeds. We conclude that our programming model allows both compile-time and run-time slot allocation. Computational complexity, deterministic results, and scalability can be balanced according to system requirements.

5 Current results and future work

The previous section shows a prototype combined GT-BE architecture. We have synthesized an input-queued router using wormhole routing with arity 5, a queue depth of 8 flits of 3 words of 32 bits, and 256 slots in CMOS12 technology. The lay-out is shown in Figure 5. It has an aggregate bandwidth of 5×500 MHz $\times$ 32 bit = 80 Gbit/s. The area of the router is 0.26 mm^2.

The area of 0.26 mm^2 depends on the use of dedicated hardware fifos, labeled GQ and BQ in Figure 5. The router would have been at least three times larger with register- or RAM-based fifos. The RAMs required for input queuing and VOQ in an on-chip router have few addresses so that their overhead makes them

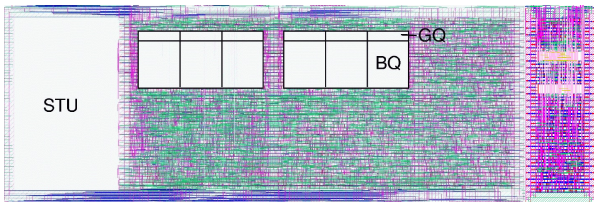


Figure 5. Lay-out of a combined GT-BE router.

as large as (area-inefficient) register files. Decreasing the queue depths reduces the buffering area (with registers at least), but also degrades the router performance.

Dedicated hardware fifos enable both input and virtual output queuing strategies using wormhole routing because of the reasonable buffering cost. For example, VOQ with two-flit deep fifos is only moderately larger than the input queuing with fifos of depth 8 of Figure 5. Virtual cut-through routing in combination with input queuing is also affordable now, because for packets of at most 8 flits, it has the same cost as the prototype.

The slot table (labeled STU in Figure 5) occupies a significant part of the router, for two reasons. Logically the slot table is very large (256 slots). It is not worthwhile to reduce the number of slots because the RAM is very area inefficient. We are investigating more advanced slot table schemes and new memory architectures to reduce the size and area of the slot table. The cost of offering time-related guaranteed services is then lower.

We separately synthesized the data and control paths (cf. Figure 4) with arities ranging from 3 to 13 to verify their speeds. With increasing arity, the speed of the data path reduces little. The speed of the control path decreases by a factor of two, corresponding to the complexity increase of the scheduling. For each arity, we balance the performance of the data and control paths by adjusting the flit size as needed, as shown in Section 4.3.1. The data and scheduling frequencies of the prototype router are 500 MHz and 166 MHz, respectively, with a flit size of 3.

Our results show that the cost and performance of the combined GT-BE router can make it the basis of a router-based network on chip. It further shows that dedicated hardware fifos significantly reduce buffering area and so enable both input queuing and VOQ, with wormhole and virtual-cut through routing.

6 Conclusions

In this paper we show that guaranteed services are essential to provide predictable interconnects that enable compositional system design and integration. However, guarantees typically utilize resources inefficiently. Best-effort services overcome this problem but provide no guarantees. So, integrating guaranteed and best-effort services allows efficient resource utilization, yet still providing guarantees for critical traffic.

Time-related guarantees, such as throughput and latency, can only be constructed on a NoC that intrinsically has these properties. We therefore define a router-based NoC architecture that combines guaranteed and best-effort services. The router architecture has conceptually two parts: the guaranteed-throughput (GT) and best-effort (BE) routers. Both offer data integrity, lossless data delivery, and in-order data delivery. Additionally, the GT router offers guaranteed throughput and latency services using pipelined circuit switching with time-division multiplexing. The BE router uses packet switching, virtual cut-through or wormhole routing, and input queuing or virtual output queuing.

We combine the GT and BE router architectures efficiently by sharing router resources. The guarantees are never affected by the BE traffic, and links are efficiently utilized because BE traffic uses all bandwidth left over by GT traffic. Connections are programmed using BE packets. The programming model is robust, concurrent, and distributed. It enables run-time and compile-time, deterministic and adaptive connection management.

For all our architecture choices, we show the trade offs between hardware complexity and efficiency. Our choices are motivated by a prototype router which has an area of 0.26 mm^2 in CMOS12 and offers 80 Gbit/s aggregate throughput. We use dedicated hardware fifos to significantly reduce the area of the data queues. With RAM- or register-based queues the router area would have been at least 3 times larger.

Dedicated hardware fifos enable (a) input queuing using both wormhole and virtual cut-through routing, and (b) virtual output queuing using wormhole routing. The buffer costs are too high, however, for virtual output queuing with virtual cut-through routing.

The cost of offering time-related guaranteed services is still high for our router. We are investigating how to reduce this cost.

An attractive feature of our router architecture is the ability to combine separately optimized data and control paths by adjusting the flit size.

In conclusion, we describe and motivate a choice of architectures for routers, which are an essential component in a NoC. They fulfill our NoC requirements by providing guaranteed services, and satisfy the efficiency constraint by offering best-effort services.

References

[1] M. Ali and M. Youssefi. The performance analysis of an input access scheme in a high-speed packet switch. In *INFOCOM*, 1991.
[2] J. Bainbridge and S. Furber. CHAIN: A delay-insensitive chip area interconnect. *IEEE Micro*, (5), 2002.
[3] L. Benini and G. De Micheli. Powering networks on chips. In *ISSS*, 2001.
[4] L. Benini and G. De Micheli. Networks on chips: A new SoC paradigm. *IEEE Computer*, 35(1):70–80, 2002.
[5] W. J. Dally and B. Towles. Route packets, not wires: On-chip interconnection networks. In *DAC*, 2001.
[6] A. deHon. Robust, high-speed network design for large-scale multiprocessing. TR 1445, MIT, AI Lab., 1993.
[7] K. Goossens et al. Networks on silicon: Combining best-effort and guaranteed services. In *DATE*, 2002.
[8] P. Guerrier and A. Greiner. A generic architecture for on-chip packet-switched interconnections. In *DATE*, 2000.
[9] M. J. Karol et al. Input versus output queueing on a space-division packet switch. *IEEE Trans. on Communications*, COM-35(12):1347–1356, 1987.
[10] K. Keutzer et al. System-level design: Orthogonalization of concerns and platform-based design. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 19(12):1523–1543, 2000.
[11] N. McKeown. *Scheduling Algorithms for Input-Queued Cell Switches*. PhD thesis, Univ. of California, Berkeley, 1995.
[12] J. Rexford. *Tailoring Router Architectures to Performance Requirements in Cut-Through Networks*. PhD thesis, Univ. Michigan, 1999.
[13] M. T. Rose. *The Open Book: A Practical Perspective on OSI*. 1990.
[14] M. Sgori et al. Addressing the system-on-a-chip interconnect woes through communication-based design. In *DAC*, 2001.
[15] A. S. Tanenbaum. *Computer Networks*. 1996.
[16] H. Zhang. Service disciplines for guaranteed performance service in packet-switching networks. *Proc. of the IEEE*, 83(10):1374–96, 1995.