A Router Architecture for Networks on Silicon

Edwin Rijpkema, Kees Goossens, and Paul Wielage
Philips Research Laboratories
Prof. Holstlaan 4, 5656 AA Eindhoven, The Netherlands
E-mail: {edwin.rijpkema, kees.goossens, paul.wielage}@philips.com.

Abstract— **To deal with the increasing design complexity of integrated systems reuse of intellectual property (IP) blocks is promoted. A system architecture then becomes a composition of a heterogeneous set of such IP blocks together with a network that interconnects these blocks. The main challenge of system design therefore shifts from computation (IP blocks) to communication and storage (interconnect and memories). This means that applications become dynamic compositions of IP blocks which requires that the network is scalable (in the number of attached IP blocks), programmable and behaves predictably under the traffic offered by those blocks.**

As the feature size decreases the relative cost of wires increases. We therefore search for an interconnect network that efficiently uses wires through sharing by introducing routers.

For a flexible and efficient solution at least two traffic classes must be support by the network, viz., guaranteed-throughput (GT) and best-effort (BE). For GT traffic communication channels are set up to transport data between IP blocks (possibly via memory). Best-effort traffic is never lost, but no latency or through-put guarantees are given. We also address the conflicting requirements of GT and BE traffic [1]. Our router is packet-switched and uses input-queuing with an efficient packet/flit scheduling [2] for BE traffic, whereas efficient time division multiplexing scheme is used is used for GT traffic.

The focus of this paper is on the derivation of a cost-effective router and network suitable for on-chip integration.

I. Introduction

As the feature size of transistors decreases more functionality can be put on a single chip which implies an increased design complexity. To deal with this complexity the reuse of previously designed IP is to be supported. IP reuse can be done between two extremes.

In one extreme approach the granularity of reuse does not change: the IP blocks of old systems are reused directly in the new system. All memory that appears in the old system is mapped onto off-chip shared memory. Typically the communication bandwidth within a system is an order of magnitude higher than the bandwidth that is communicated over the ports of that system. So, *opening* such a system causes the communication bandwidth that is visible in the architecture to increase rapidly with the speed[1] of the circuits and the number of IP blocks. This approach benefits from the relatively low cost of off-chip memory compared with that of on-chip memory but suffers from the high off-chip memory bandwidth.

In the other extreme approach IP reuse is coarser: a complete old system defines a single new IP block for reuse in the new system. The the memory of the old system is mapped to local embedded memory in the new system, which is composed now of subsystems exclusively. This approach preserves the locality in the subsystem. As a consequence the total communication that is visible at the top level in the architecture may be roughly the same or even less than in the old system. Clearly, this approach suffers from the relative high cost of embedded memory.

The way system designers design systems will be somewhere in between these extreme approaches but we observe a shift from the first to the second approach. To be precise, this shift is not so much in the size of the memories but rather in the communication bandwidth to and from the memories. This means that with relatively small on-chip embedded memories, the potential communication bottleneck to off-chip memory is avoided.

From this we conclude that the communication bandwidth requirements at the top-level architecture are moderate.

To allow the communication between the IP blocks and between IP blocks and memory a communication network is required. In order to derive such a network we take a look to it from two different perspectives, viz., the system-level design and circuit-level design perspectives.

System-level design perspective

In general the IP blocks have various traffic characteristics and require differing services of the interconnection network. For example, real-time signal processing IP blocks may require a service that provides them with a guaranteed bandwidth and, at the same time, minimize the jitter (delay variation), whereas a CPU-like IP block requires a latency as low a possible for it to operate efficiently.

[1] Up-scaling of the speed of a circuit from one technology to a newer one is a problem on itself and is not addressed in this paper.

To design a system a communication network is required that meets the set of all simultaneous communication requirements of the individual IP blocks in a certain application. To support the design of such system predictable behavior of the network is essential. For example, when a set of streaming IP blocks is to be connected to the network, it is essential to know whether or not a particular network can simultaneously provide the IP blocks with their guaranteed-throughput requirements.

The IP blocks that will appear in future systems may contain applications on itself. Thus new applications are compositions of smaller applications. To allow the mapping of a complete application domain onto such a system the network must allow a flexible composition of IP blocks to form a new application. It also requires a high degree of connectivity. In addition, communication patterns may change dynamically, for example as the result of user interaction.

Circuit-level design perspective

The decrease of feature size speeds up the IP in every new process generation. When the global wires that connects these IP blocks are scaled in width and spacing with a factor of about 0.7 every process generation, then the available bandwidth on these wires decreases [3]. Since not only GT services but also BE services are required and since BE traffic may be latency constrained, low latency interconnect is required. This means that it is not sufficient to just segment all long wires into short wires (only increases available bandwidth) but that long wires must have low latency themselves. This can be accomplished by not scaling down the width and spacing of the wires to meet lower latency requirements than was needed in a previous generation. Since the cost of these low latency wires is relatively high, it is important to exploit the offered bandwidth of the global wires as most as possible without degrading the latency of the packet too much.

From these two perspectives we conclude that we need a high degree of connectivity to support both GT and BE traffic while global wires are efficiently utilized. A network of routers allows sharing of wires. A router provides both GT and BE services to keep the utilization of the links high.

The rest of this paper is organized as follows: Section II deals with a number of issues that arise when designing such a network. Section III and Section IV describe an architecture for GT traffic and BE traffic, respectively. In Section V we draw some conclusions.

II. Network Design Issues

This section deals with a number of issues that arise when designing a network. Services that are to be provided by the network and how the network internally is organized in order to enable such services are discussed in Section II-A. Section II-B introduces the *switching mode* of a network. These are *circuit switching* and *packet switching*. Section II-C evaluates the *routing mode* of a packet-switched network. The routing modes are *store-and-forward routing*, *virtual cut-through routing*, and *wormhole routing*.

A. Network services and organization

This section deals with the services the network must provide for the GT and BE traffic classes, and the internal organization of the network on top of which these services are to be implemented.

Two important issues in the design of a network (the network layer in the OSI model [4]) are the services provided by the network (to the transport layer) and internal organization of the network [4]. The services to be provided must support the traffic classes GT and BE efficiently. Proper internal organization of the network helps in implementing the required services. For example, when the network is such that no reordering of data can ever happen, no additional ordering hardware at the boundary of the network is required to implement in-order delivery.

Basically, interconnection networks may provide two types of services, viz., *connectionless* and or *connection-oriented* which can either be reliable or unreliable [4]. In a connection-oriented service a connection is set up prior to the communication of the actual data packets. When a connection has been established the packets arrive at the destination in the same order as they have been sent by the source. In a connectionless service all packets are sent independently and may arrive out of order at the destination. Moreover, a reliable service is one in which all packets that has been sent arrive at the destination (no packets are lost).

Services provided by the network

Clearly, for GT traffic a reliable connection-oriented services is required. In addition to guaranteed, in-order delivery the service must secure the required bandwidth for each connection. To set up a GT connection admission control is done to decide whether or not the connection with the requested bandwidth is established. The network will not accept more data from a GT source than was agreed to prevent interference with other GT connections.

Guaranteed delivery is important for BE and GT traffic because retransmission will increase the observed latency, decrease the throughput of the network, and increase contention (leading to more packet loss). No packets are therefore ever lost by the network. For BE traffic we believe that in-order delivery is also a desired property since it avoids the necessity to reorder at the network boundary. We will see that this does not complicate the internal organization of the network. Since for BE traffic we require that latency is as low as possible it is not acceptable to go through a connection set-up phase before the actual data may be communicated. Therefore we require that network provides a lossless connectionless service that provides in-order delivery.

Internal organization of the network

As already pointed out in the introduction, we aim for high utilization of the physical wires. At the same time the internal rate the routers operate on is comparable to the rate on which the links are to used. We therefore restrict the routers to the functionality required to implement the previously mentioned services (lossless, order-preserving).

B. Switching mode

There are basically two switching modes: circuit switching and packet switching. In a circuit-switched network connections are set up by establishing a conceptual physical path from a source to a destination. In this basic scheme links are shared in the sense that two connections are allowed to use the same link only at different points in time. A way to share a link over multiple connections is to use a time-division multiplexing (TDM) scheme which is discussed in Section III.

In packet switching data is segmented into *packets*. A packet is composed of a header and a payload. The payload is the actual data to be transmitted and the header contains information that is used by the router to forward the packet to the destination. Classically the header contains the destination address but other approaches, like source routing, exist as well. In a packet-switched network links are not reserved for connections; sharing of a link is interpreted as interleaving of packets routed over that link.

An advantage of circuit switching with TDM over packet switching is that it naturally provides us with guaranteed-throughput connections. A disadvantage of circuit switching with TDM is the bad utilization of the links when there is much bursty traffic [5]. We therefore propose a router that supports both time-division multiplexed streams of GT traffic and packet switching for BE traffic to allow high link utilization. In Section IV-B we show that the proposed scheme can use the unreserved GT and unused GT bandwidth for BE packets. Most alternative approaches are based on output queuing or use priority queues for time-stamped packets, both of which are expensive to implement. A packet-switched alternative that we know of is presented in Hung *et al.* [6]. However, our circuit-switching approach requires one buffer per input port instead of N buffers for packet switching of Hung *et al.* (N is the number of output ports of the router).

C. Routing mode

The routing mode is also referred to as network flow control and applies to packet-switched networks. It deals with the way packets are transmitted between the routers. There are three routing modes, viz., store-and-forward routing, virtual cut-through routing, and wormhole routing [7].

In store-and-forward routing, an input packet is received in its entirity and stored in the router. Only then is it forwarded to the next router in the network. This requires storage for the complete packet. Moreover, a packet is sent only after complete reception, introducing a latency of the packet per router of at least the time required to receive the packet.

In virtual-cut-through routing a packet is forwarded as soon as the next router gives a guarantee that a packet will be accepted completely. So, when at some router the header of a packet arrives and the next router guarantees that it will consume a complete packet, the packet at the first router is sent immediately without first storing the complete packet. When at some time no guarantee is given, the packet whole packet is not forwarded and stored, in its entirety, in the router. Thus, virtual-cut-trough routing requires buffers for a complete packet, like store-and-forward routing, but allows low-latency communication.

In wormhole routing packets are partitioned in so called flits (flow control digits). A flit is passed to the next router when that router is indicates it accepts that flit. 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. So, a greedy choice of starting to communicate the first flit of a packet can cause a packet to be spread over a number of routers and when it blocks, it occupies all links in between them. So, like virtual-cut-trough wormhole routing allows for low-latency communication. In addition wormhole routing requires less memory because it stores a flit rather than a packet. On the other hand, wormhole routing is more sensitive to deadlock and generally results in lower link utilization than virtual-cut through routing. One way to improve on wormhole routing's performance is by multiplexing multiple logical ports onto a single physical port. However, this substantially increases the cost of the router.

So, for low-latency BE traffic we must select between virtual cut-through and wormhole routing. A complicated issue is how to combine the TDM circuit switching with packet switching with either one of these routing modes. In [1] it is shown that combining traffic that uses wormhole routing with traffic for which throughput guarantees are to be given can degrade link utilization drastically. In our context this is illustrated in Figure 1. Two 2×2-routers are indicated by R_1 and R_2, and the network terminals are identified by t_i, $i = 1, 2, \cdots 6$. Assume that R_1 receives BE packets via terminal t_1 that are all destined to t_5 and that the bandwidth of these packets require 10% of the capacity of a link. Similarly, packets go from t_2 to t_6 and require only 1% of the link capacity. Finally there is a GT stream from t_4 to t_6. The GT stream claims and uses 99%

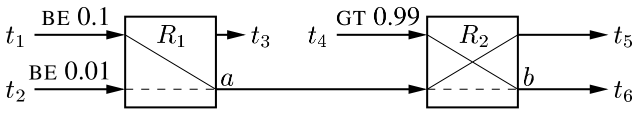


Fig. 1. Combining GT with wormhole routing can drastically degrade link utilization.

of the bandwidth and thus occupies the output link from output port b to terminal t_6 for 99% of time. The BE stream shares port b can send a flit only in the remaining 1% link capacity. Every time GT data arrives for port b the transmission of the BE packet over port b is preempted. This can cause long latencies for the packets that belong to the 1% BE stream. It also causes the link between R_1 and R_2 to be occupied almost continuously by the 1% BE stream (because flits of different packet are not interleaved). Thus the BE packets of the 10% stream obtain less than 10% of the rate of the link. This means that in this example, the link between R_1 and R_2 has a utilization that is even below 11% of its theoretical capacity.

The cause of the problem in Figure 1 is clear. It is the combination of *flit preemption* and wormhole routing that blocks links for significant part of the time. To get around this problem there are basically two approaches: 1) use virtual cut-through routing rather that wormhole routing, and 2) perform GT communication in relatively large blocks of data and large periods of no data.

The first approach guarantees that a complete packet will be accepted in the next router such that the incoming link of the next router does not block. As mentioned before this is at the cost of extra memory.

The second approach ensures that flit preemption rarely occurs. Referring to the example this is seen as follows. When the 99% of GT data is grouped in blocks of 10 time units then this bandwidth is obtained by alternative sending 99 blocks of data followed by 10 time units nothing. Now, when the packet size of the BE stream is small compared to these 10 time units, a complete packet of the 1% be stream is sent in the 10 time units and the link between R_1 and R_2 can be used by the 10% BE stream immediately after the packet has been send. While the first approach suffers from additional memory requirements in the router, this second approach may suffer from additional latency in the GT stream.

III. A GUARANTEED-THROUGHPUT (GT) ROUTER ARCHITECTURE

This section explains how time-division-multiplexed circuit switching enables us to guarantee throughput. Since the operation rate of the links is about the same as the internal operation rate of the router, the router internally switches in space. We propose a crossbar switch for this. To prevent contention on the switch a separate set-up phase takes place. When a new GT connection would introduce a clash with existing connections on the switch, the requesting party is informed about this. Every router has a *router table* in which every row contains the information to program the crossbar switch. A row of the router table is referred to as a *slot*. We avoid contention by: first ensuring there is a notion of synchrony between all routers such that all routers go from one slot to the next simultaneously, and secondly filling slots such that the set of simultaneous GT connections is contention free.

Section III-A introduces the model of computation that specifies the required notion of synchrony in an abstract manner. Section III-B deals with the internal working of the GT router. Section III-C explains how GT connections are programmed.

A. Model of computation

We now explain the network of routers in the context of the synchronous dataflow (SDF) model [8]. An SDF network is a set of so called *actors* that are connected by *arcs*. An actor is an executable piece of computation and may contain state. When the piece of computation is executed the actor is said to *fire* and consumes a number of *tokens* from its input arcs, and produces a number of tokens on its output arcs. A token is an abstract representation of quantum of information and allows the actors to communicate. Every input port and output port of an SDF actor specifies the number of tokens that are produced (for an output port) or consumed (for an input port) when the actor fires. An actor may only be fired when it is *enabled*, i.e when all its input arcs contain at least as many tokens as the number on corresponding input ports.

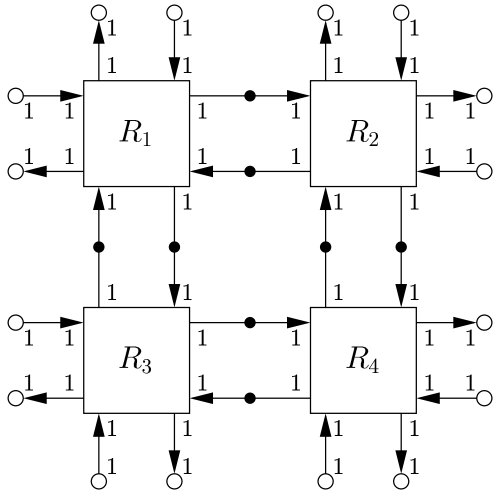


Fig. 2. Example of a router network in the SDF model of computation.

In SDF networks arcs may contain delays. A delay may be seen as a token that is initially present at the arc at which the delay is placed. In this paper we consider the case that every internal arc in the network has one delay, that is, one initial token.

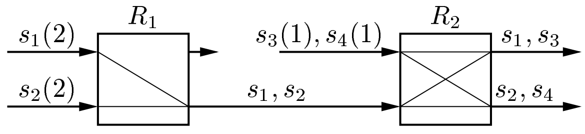
The numbers associated with the ports are constant during the execution of the system but may vary from port to port. However, in this paper all ports consume one token per firing.[2] In this case an *SDF iteration* of the network is the single firing of every actor in the network, and the number of tokens on every arc before an after an SDF iteration is equal. From now we will iteration as shorthand for SDF iteration.

Every router of the router network is thus represented by an actor and every link is represented by an arc. An example of a simple router network in the SDF model with four 2×2-router is depicted in Figure 2. The circles in the figure represent network interfaces that connect the network to the sources and sinks. Note that after every iteration all internal arcs contain a token. Thus, when the network interfaces provide enough data to the network, all routers in the network can operate in parallel.

B. Internal GT router architecture

To explain the internal router architecture consider an $N \times N$ router, a router with N input ports and N output ports. The ports of the router are denoted by i_n and o_n, $n = 0, 1, \cdots, N-1$, respectively. The router table R is specified by an $S \times N$ matrix. The rows of R are denoted

[2]In future work we will consider the production and consumption of more than one token per arc per firing to model multi-rate execution. In order to avoid "dummy" communication we might switch to cyclo-static dataflow [9], [10] as well.



$$\begin{bmatrix} \varnothing & 0 \\ \varnothing & 1 \\ \varnothing & 0 \\ \varnothing & 1 \end{bmatrix} \qquad \begin{bmatrix} \varnothing & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & \varnothing \end{bmatrix}$$

Fig. 3.

by vectors $\mathbf{r}_s$, $s = 0, 1, \cdots, S-1$. The slot a router is in is called the current slot of that router. All routers are in the same current slot. The elements $R(s, n)$ are in the set $\{\varnothing, 0, 1, \cdots, N-1\}$. Let $m = R(s, n)$. When $m \neq \varnothing$ output port o_n has been reserved. This means that GT tokens read at input port i_m in slot $(s-1)\%S$ are sent to destination port o_n in slot s. When $m = \varnothing$ it means that no such reservation is made for any of the input ports.

In our GT routing approach, every GT token that is read in slot $\mathbf{r}_s$ in some router is read in slot $\mathbf{r}_{(s+1)\%S}$ in the next router in the path the token follows. (The symbol $\%$ represents the remainder after integer division.)

An example of a simple router network with two 2×2 routers with a router tables size $S = 4$ is shown in Figure 3. In this figure four GT connections are represented by the streams s_1, s_2, s_3, and s_4. The number of slots allocated for that stream shown in parentheses.

The first output port of R_1 is unused and consequently the first column of the routing table is empty. The second column of the routing matrix of R_1 indicates that tokens from its inputs are written alternately on the second output port. Consequently both s_1 and s_2 are routed with the desired bandwidth without contention in the first router. In the second router the first output port receives tokens belonging to stream s_1 and stream s_3. Since the tokens from s_1 are routed in slots 0 and 2 in the first router, they are routed at slots 1 and 3 in the second router. This is seen by the two ones in the first column of R_2's router table. The single slot that s_3 requires is scheduled in slot 2 of the first column. Similarly, as indicated by the ones in the second column of R_2's routing table, tokens of s_2 are scheduled in slots 0 and 2. Finally, s_4's tokens are scheduled in slot 1.

It is not required a GT token is available in every reserved slot. When no GT packet arrives in a reserved slot, a BE packet can be sent over the claimed but unused slot of the link. Since the SDF model requires a token to be written every iteration, when no BE or GT data is available, *empty tokens* are produced on the output arcs.

C. Programming model

This section deals with the way GT connections are set up and torn down. Initially every router table is empty. Therefore, the only way to communicate with the routers is by means of BE packets. *System packets* are a special type of BE packets that are dedicated to manage the setting up and tearing down of GT connections. The way these system packets are routed is dealt with in Section IV-B.

There are three packet types that are used to set up and tear down connections, viz., `SetUp`, `AckSetUp`, and `TearDown`. In addition there are two packet types to reset and start a complete router network, viz., `Reset` and `Start`. There are more types of packet types (also to manage connections) but these are outside the scope of this paper.

A `SetUp` packet is used to set up a GT connection from a source to a destination. It can be sent by either the source or the destination of the GT packets. The same holds for `TearDown` and `AckSetUp` packets. However, here we assume that the `SetUp` is sent by the source of the GT tokens and that the `TearDown` and `AckSetUp` are sent in the direction of the source of the GT tokens.

`SetUp` packets contain three pieces of information: the source of the GT tokens, the destination of the GT tokens, and a slot number. The `SetUp` packet is routed towards the destination and tries to reserve the the slot specified by the slot number. When this reservation is successful the slot number is incremented by one and the `SetUp` packet is routed to the next router (or the destination). When the `SetUp` packet successfully arrives at the destination, a `AckSetUp` packet is sent back to the source to inform it about the successful set-up. If, however, at some point the slot requested is already occupied the `SetUp` is discarded and a `TearDown` packet is sent back to the destination. This `TearDown` packet follows the same route the corresponding `SetUp` packet followed in order to remove the reservations made from the routing table. When the source receives the `TearDown` packet it knows the set-up hast been unsuccessful.

The `Reset` packet initializes the entire network to bring it in a well-defined idle state. The `Start` packet is then used to activate the network services. Both packets flood the network such that every router receives these packets.

IV. A BEST-EFFORT ROUTER (BE) ARCHITECTURE

The term queuing strategy is used in the context of packet switching. Basically there are two queuing strategies, viz., input queuing and output queuing. Classically, in input queuing a router has a single input queue for every incoming link while in output queuing there are N output queues for each output link[3] where N is the number of input links for the output queued router. Most research on giving guarantees assumes non-blocking output-queued routers [11]. When the internal rate of operation of the router is about the rate at which the wires are driven, output-queued routers require N^2 physical buffers making them expensive for system-on-chip implementations. To avoid these high costs, we use an alternative queuing strategy, known as virtual output queuing (VOQ) [12], that tries to combines the advantages of input-queuing and output-queuing. Unlike input-queuing, VOQ does not suffer from head-of-line blocking. Not all virtual output queues cannot be accessed simultaneously, but with a proper scheduling algorithm 100% throughput can be achieved.

A. Virtual-output-queuing and iSLIP

A basic scheme of an virtual-output-queued architecture for an $N \times N$ router is shown in Figure 4.

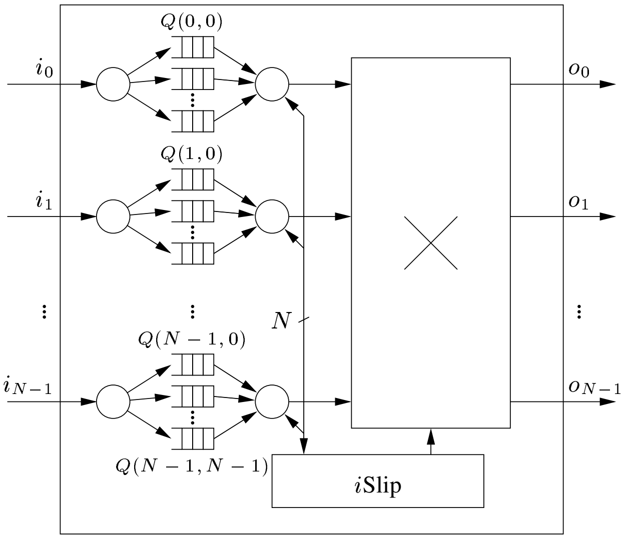


Fig. 4. Schematic of an virtual-output-queued router architecture.

For every input port there is a set of N logical queues, one for each output port. Queue $Q(m,n)$ is the queue that contain the tokens that are read from input port i_m and are destined for output port o_m. One essential difference between VOQ and traditional output queuing is that in VOQ only one queue per input port can be read per iteration. This allows all logical queues of one input port to be implemented with a physical single memory. We assume that

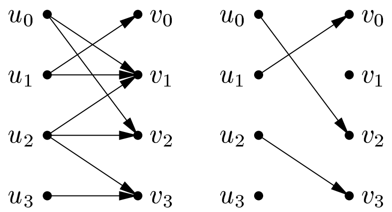
[3] Instead of having N output queues per output port a single output queue that allows writing with a factor N times the rate of the link can be used. This is just a matter of switching in time or switching in space.

the actual switching is done with a crossbar switch. In order to meet the constraints of only one read per physical memory, of conflict free routing, and at most one write per output port, the read accesses on the queues must be scheduled. These two constraints are formulated as

$$\begin{aligned} \forall m.\ \sum_n A(m,n) &\leq 1 \\ \forall n.\ \sum_m A(m,n) &\leq 1 \end{aligned} \tag{1}$$

where A is a matrix with binary elements $A(m,n)$ that indicate whether or not a token from input i_m is switched to output o_n in the current iteration of the router. The problem how to find A such that as many elements of A are 1 (at most N) can be formulated as a bipartite graph matching problem. A bipartite graph is constructed as follows. Create a vertex u_m and v_n for every input port i_m and output port o_n, respectively. Create an edge (u_m, v_n) if and only if queue $Q(m,n)$ is non-empty. Let R be the *request matrix* whose elements $R(m,n) \in \{0,1\}$ indicate whether or not $Q(m,n)$ contains data. Figure 5 gives an example of a request matrix for $N = 4$, the corresponding bipartite graph and a matching for that graph.

$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



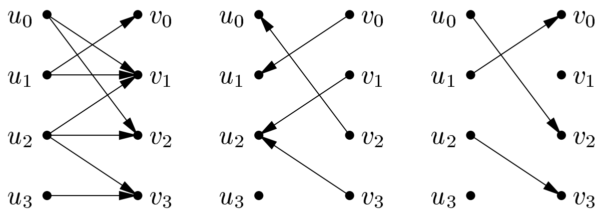
(a) (b) (c)

Fig. 5. Example of (a) a request matrix, (b) its derived bipartite graph, and (c) a corresponding matching.

The matching in the figure has *size* three. The matching is a *maximal matching* since no edges from the original graph can be added to increase the size of the matching. It is not a *maximum size matching* since there exists a matching of size four.

An efficient algorithm to find a maximal matching is iterative SLIP (iSLIP) [2]. The output of iSLIP can be used directly control a crossbar switch to perform the actual switching of the tokens. iSLIP iteratively computes a maximal matching in $\log_2(N)$ iterations. However, for a low-latency implementation of iSLIP with low complexity we consider to use iSLIP with only one iteration.

Basically an iSLIP iteration works in three steps. In the first step for every non-empty queue $Q(m,n)$ a *request* to access output port o_n is made. In the second step among all requests made to output port o_n a single request is *granted*. In the third step for every input port at most one grant is *accepted*. For the bipartite graph in Figure 6(b) the three steps are illustrated in Figure 6.



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

Fig. 6. Single iteration of iSLIP.

An integer value between 0 and $N-1$ is associated with every node u_m and every node v_n in the bipartite graph. Of all requests for output port o_n the request from the input port o_m with the smallest value m that is higher (modulo N) than the value associated with o_m. Similarly, among the set of grants at input port i_n the one with the lowest index higher than the value associated with i_n is selected. In iSLIP, the values associated with the nodes are set to the selected indices only after an accept. An implementation of this arbitration method is done by means of grant arbiters [2].

B. *Combining the* BE *with the* GT *Architecture*

To efficiently use the bandwidth all slots that are not reserved and all slots that are reserved but not used by GT connections can be used by BE traffic. Only when there are no GT tokens and BE packets available to switch to an output port does the output port remain idle (recall that an empty packet is written to the output port in such cases). In our combined architecture the routing of the GT tokens is never affected by the BE traffic.

In order to deal with the system packets these packets are routed to an internal unit to process them and then pass them on to the proper output port. This internal unit has the responsibility to handle the system packets properly and to access the routing table accordingly.

V. CONCLUSIONS

We have motivated that high link utilization and high connectivity are two objectives in current and future system-on-chip designs. On the other hand, as system design becomes a composition (interconnection) of IP blocks, their communication requirements are to be met in a predictable and systematic way. In order to do so we introduced guaranteed-throughput (GT) and best-effort (BE) traffic classes and proposed a router architecture that supports both. Internally the router architecture uses time-division-multiplexed circuit switching for GT traffic and uses virtual-input-queuing packet switching for BE traffic. We further proposed to use single iteration iSLIP for the contention resolution of the crossbar switch.

REFERENCES

[1] Jennifer Rexford, *Tailoring Router Architectures to Performance Requirements in Cut-Through Networks*, Ph.D. thesis, University of Michigan, Dept. of Computer Science and Engineering, 1999.

[2] Nicholas William McKeown, *Scheduling Algorithms for Input-Queued Cell Switches*, Ph.D. thesis, University of California, Berkeley, 1995.

[3] Robert H. Havemann and James A. Hutchby, "High-performance interconnects: An integration overview," *Proceedings of the IEEE, special issue on Interconnections-Addresssing the next Challange of the IC Technology*, vol. 89, no. 5, pp. 586–601, may 2001.

[4] Andrew S. Tanenbaum, *Computer Networks*, Prentice-Hall, Inc., 1996.

[5] Jean Walrand, *Communication Netwokrs: A First Course*, Richard D. Irwin, Inc., and Aksen Associates, Inc., 1991.

[6] A. Hung, G. Kesidis, and N. McKeown, "ATM input-buffered switches with the guaranteed-rate property," in *IEEE ISCC'98*, Athens, Greece, july 1998.

[7] Anujan Varma and Raghavendra C.S., *Interconnection Networks for Multiprocessors and Multicomputers: Theory and Practice*, IEEE Computer Society Press, 1994.

[8] Edward A. Lee and D. G. Messerschmitt, "Synchronous data flow," *Proceedings of the IEEE*, september 1987.

[9] G. Bilsen, M. Engels, R. Lauwereins, and J. PeperStaete, "Cyclo-static data flow," in *ICASSP-95., 1995 International Conference on Acoustics, Speech, and Signal Processing*, 1995, vol. 5, pp. 3255–3258.

[10] Thomas M. Parks, José Luis Pino, and Edward A. Lee, "A comparison of synchronous and cycle-static dataflow," in *Conference Record of the Twenty-Ninth Asilomar Conference on Signals, Systems and Computers*, 1996, vol. 1, pp. 204–210.

[11] Hui Zhang, "Service disciplines for guaranteed performance service in packet-switching networks," *Proceedings of the IEEE*, vol. 83, no. 10, pp. 1374–96, october 1995.

[12] M.K.M. Ali and M. Youssefi, "The performance analysis of an input access scheme in a high-speed packet switch," in *INFO-COM '91. Proceedings. Tenth Annual Joint Conference of the IEEE Computer and Communications Societies. Networking in the 90s., IEEE*, 1991, vol. 2, pp. 454–461.