Optimal Link Scheduling on Improving Best-Effort and Guaranteed Services Performance in Network-on-Chip Systems*

Lap-Fai Leung
Department of Electrical and Electronic Engineering
Hong Kong University of Science and Technology
Clear Water Bay, Hong Kong SAR, China
eefai@ee.ust.hk

Chi-Ying Tsui
Department of Electrical and Electronic Engineering
Hong Kong University of Science and Technology
Clear Water Bay, Hong Kong SAR, China
eetsui@ee.ust.hk

ABSTRACT

With the advent of the multiple IP-core based design using Network on Chip (NoC), it is possible to run multiple applications concurrently. For applications with hard deadline, guaranteed services (GS) are required to satisfy the deadline requirement. GS typically under-utilizes the network resources. To increase the resources utilization efficiency, GS applications are always complement with the best-effort services (BE). To allow more resource available for BE, the resource reservation for GS applications, which depends heavily on the scheduling of the computation and communication, needs to be optimized. In this paper we propose a new approach based on optimal link scheduling to judiciously schedule the packets on each of the links such that the maximum latency of the GS application is minimized with minimum network resources utilization. To further increase the performance, we propose a novel router architecture using a shared-buffer implementation scheme. The approach is formulated using integer linear programming (ILP). We applied our algorithm on real applications and experimental results show that significant improvement on the overall execution time and link utilization can be achieved.

Categories and Subject Descriptors

C.5.4 [COMPUTER SYSTEM IMPLEMENTATION]: VLSI Systems

General Terms: Algorithms, Design, Performance

Keywords: Network-on-Chip, Latency, Routing

1. INTRODUCTION

Continuous improvement in process technology allows billions of transistors and hundreds of computation resources and processor cores to be put on a single chip in the very near future. System-on-chips (SoCs) with these capabilities require efficient communication architecture to offer scalable bandwidth and parallelism. Network-on-chip (NoC) has emerged to be a viable

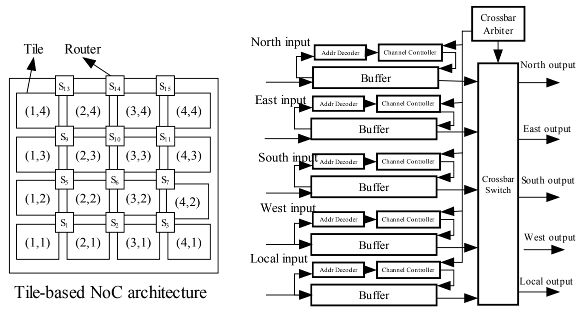


Figure 1(a) Tile-based NoC architecture (b) On-chip router

solution to provide the necessary communication links among different on-chip processing cores [1]. Instead of using dedicated wire or share bus, the computation resources are mapped to a tile-based structure and each tile is connected to its adjacent neighbors by interconnection links. The data are packetized and transferred between links through a router embedded within each tile (Figure 1(a)). A typical router [8] with uniform-sized buffer in each port is shown in Figure 1(b).

Multiple IP-core based design using NoC allows multiple applications running at the same time. Some of them can be applications with hard deadline requirement, such as video/audio applications, that need critical communication requirement while others may have soft real-time requirements and need only non-critical communication. Based on these different requirements, the communication scheduling can be divided into two types. The guarantee service (GS), which caters for the hard deadline requirement while best effort (BE) is used for the other non-critical communication [13]. BE exploits the network resource that is unused by the GS. In particular, GS is always combined with BE in order to avoid the low resource utilization which is mainly due to the resource reservations for the worst-case scenarios. The Aethereal NoC design framework presented in [13] aims at providing a complete infrastructure for developing heterogeneous NoC with end to end quality of service guarantees. The network supports guaranteed throughput (GT) for real-time applications and best effort (BE) traffic for applications with soft or unspecified constraints. Specialized router architecture that supports BE and GS at the same time was also presented in [13].

One of the main challenging issues of designing such kind of NoC is to allocate the network resources for the GS applications such that the hard deadline requirements are satisfied while still allowing maximum available network resources for the

*This work was supported in part by Hong Kong RGC CERG under Grant HKUST6256/04E.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
DAC 2006, July 24–28, 2006, San Francisco, California, USA.
Copyright 2006 ACM 1-59593-381-6/06/0007…$5.00.

remaining BE applications. The network resource allocation problem includes the time-slot reservation of the communication link for the packets; assignment of priority to various packets transmitting on the same communication link; and decision on the time to produce and consume the packets. All the above factors (we collectively denote these as the link scheduling problem) significantly affect the performance of the entire system. At the same time, we want to minimize the overall execution time and hence the maximum latency of the GS applications. To satisfy the hard dead-line requirement under worst case situation, the network resource allocation and scheduling of the GS applications has to be done in the static design phase. In this paper, we focus on the optimal link scheduling at the static phase. The objective is to minimize the maximum latency of the GS applications while allowing as much as possible network resources for the BE applications.

On-Chip network is by far resources limited, e.g. limited by wiring, buffer, area and energy consumption. In standard computer networks, network routers use fairly large amount of buffer spaces. The network performance is drastically impacted by the amount of buffering resources available, especially when the traffic is congested [8]. In contrast, on-chip networks should use the least amount of buffer space due to the limited energy and area budget. Since the traffic characteristics varies significantly across different applications [8], the straightforward uniform distribution of buffering resources among all the ports of the router fails to guarantee QoS under tight buffer space budget [8]. In this paper, we increase the buffer space utilization and hence increase the system performance by using a shared buffer scheme.

Wormhole routing [7, 9] is commonly used because it requires less buffer usage to achieve minimal routing latency. Each packet is segmented into flits (flow control unit). The header flit sets up the routing paths at each router. The body flits will then follow the designated path and the tail flit releases the reservation of the link in the routing path. One major advantage for wormhole routing is that it does not require the router to store the whole packet before it is sent out. This drastically reduces the latency. However, one major problem is that contention arises when the down-stream link or the router of the header flit is occupied. Previous works [8,11] used the backpressure mechanism to regulate the contention and the packets are dispatched immediately once they are ready. Under this scheme, all the flits of the packets are held in their current buffers until the downstream link is available. The main drawback is that all the network resources in the routing path are wasted since they are held by the congested packet. This would reduce the network resource efficiency and consequently, increases the latency of the whole system and decrease the resource available for the BE applications. In this paper, we judiciously schedule the packets of the GS applications such that the contention is minimized and the links can be utilized as much as possible to overcome this drawback. We propose a static link scheduling algorithm based on Integer Linear Programming (ILP) optimization To obtain the optimum static packet schedule if the processor mapping, routing and the operation characteristics of the GS applications are given. The algorithm also considers the condition that only finite buffer space is available on the router when making the scheduling decision.

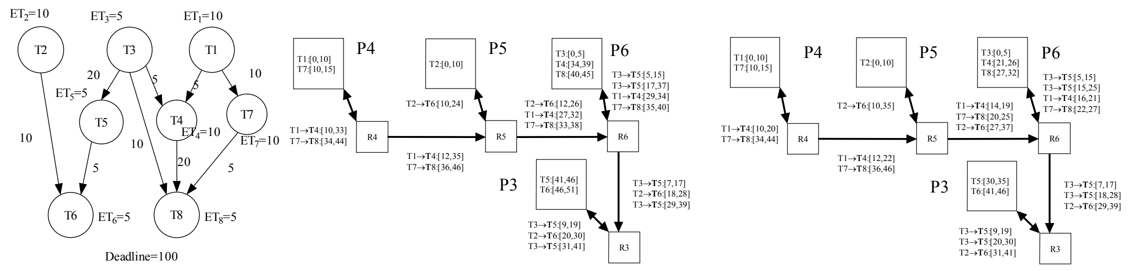
2. RELATED WORK

Wormhole routing is an ideal candidate switching technique for NoC. However, it suffers from deadlock when packets block each other in a circular fashion [7] and contention when the traffic is congested [10]. One way to solve the deadlock is to use virtual channels [9,14] but this requires a lot of buffer space and is not applicable to NoC design with tight area and power budget. Another way to solve the deadlock issue is using the deterministic Dimension-ordered routing (e.g. XY routing , odd-even routing) [9]. Under this scheme, packets are routed in one dimension first and then the other until reaching the destination. In this work we use the XY routing which is widely used in NoC design [8]. The works in [8,6] proposed stochastic communication models which was captured from the traditional marco network. Using this model, the transmission latency is unpredictable and hence the performance constraints like the deadline requirement can not be guaranteed. In this paper, we consider the worst-case characteristic of the given GS applications such that we are able to guarantee the deadline requirement under the worst-case.

Many works [10,7] have been proposed to address the various optimization problems on NoC. In [7], a mapping problem to satisfy bandwidth constraints on the links for regular tile-based Networks on Chip (NoC) architecture was tackled. In [10], a solution was proposed for the problem of mapping applications on regular NoCs for minimizing execution time and energy consumption. However, all the above works do not consider the effect of packet schedule and the computation timing on calculating the latency of the packets. Also, they do not consider the overall execution time and therefore, the hard deadline requirement of the applications can not be guaranteed.

There are many analytical models for wormhole-based network [10,15,12]. Adve and Vernon in [15] modeled a wormhole mesh network as a closed queuing network and use approximate mean value to calculate the latency under random traffic. However it assumed a single-flit buffer which makes the timing model restrictive to apply because the buffer size is usually arbitrary but finite in real applications. On the other hand, the works in [10,14] assumed infinite buffer which is not a valid assumption for real applications. Despite the assumption of infinite buffer size and the fact that they did not consider packet contention in the timing model, the work in [10] provided a detailed timing model for the wormhole mesh network.

The work in [8] proposed a buffer allocation scheme for the NoC router. The buffer size of each input port of the routers were determined according to the given traffic pattern subject to the total buffering space constraint of the whole system and the buffer space for each router can be different by a lot. This method may not be favorable for regular tile-based NoC design as it is easier to design if every router has the same amount of buffer space. Also for different applications, the traffic patterns are different. If a fixed buffering space distribution is determined in the design phase based on the traffic patterns of a set of applications, for other applications, this fixed buffer distribution may not be optimized. The work in [13] assumed infinite buffer size which is infeasible in real NoC design. In this work, we use wormhole routing with finite buffering space in the router. An accurate timing model for wormhole routing which takes packet contention into consideration is proposed.



(a) a task flow graph *G* (b) NoC assignment NA_1 with single flit buffer (c) NoC assignment NA_2 with single flit buffer

Figure 2. A motivational example

3. MOTIVATION EXAMPLE

We show the importance of considering the link scheduling using the example in Figure 2. Figure 2(a) shows the task flow graph *G* of a GS application. It contains eight tasks $T_1,..,T_8$ which are mapped to a 3x2 NoC platform and XY routing path allocation [9] are used for determining the routing of the packets among different cores. Due to the space limit, we only show the relevant resources in the figure. Here the 8 tasks are mapped onto 4 processors P3 to P6. The execution time, ET_i of task T_i's and the communication message volume (in flits) between the tasks are shown in the graph *G*. We assume the router setup time of the header flit t_r=1 time unit (tu) and the time to transmit a flit through a link t_l=1tu also. We assume the packet size is variable and the messages between two tasks are divided into packets. For example, the message $T_3 \rightarrow T_5$ is divided into two packets with equal size of 10 flits. In this example, we assume all the messages, except the one for $T_3 \rightarrow T_5$, are transmitted in 1 packet. We denote the transmission time of the packets $T_i \rightarrow T_j$ on the links as $T_i \rightarrow T_j$ [start-time,end-time] and the execution time of task T_i on processors as T_i [start-time,end-time].
Figure 2(b) shows the resulting packet schedule based on an as-soon-as-possible schedule similar to the work proposed in [10] in which the packets are sent out once they are ready. We can see that the packet $T_2 \rightarrow T_6$ is waiting at router R_6 because the down-stream link $P_6 \rightarrow P_3$ is occupied by packet $T_3 \rightarrow T_5$. Under the backpressure control scheme, the packet $T_2 \rightarrow T_6$ occupies the link $P_5 \rightarrow P_6$ longer than the normal transmission time. We can re-schedule the packets as shown in Figure 2(c). In this case, we can improve the link utilization by sending packets $T_1 \rightarrow T_4$ and $T_7 \rightarrow T_8$ earlier than $T_2 \rightarrow T_6$. By doing so we avoid contention in router R_6 and the network resources can be maximally utilized. Figure 2(c) shows a 9.8% reduction in overall execution time than that of Figure 2(b). From this example, we can see that the static scheduling of the packet has a profound impact on the possible contention and hence on the overall execution time and also the link resource utilization.

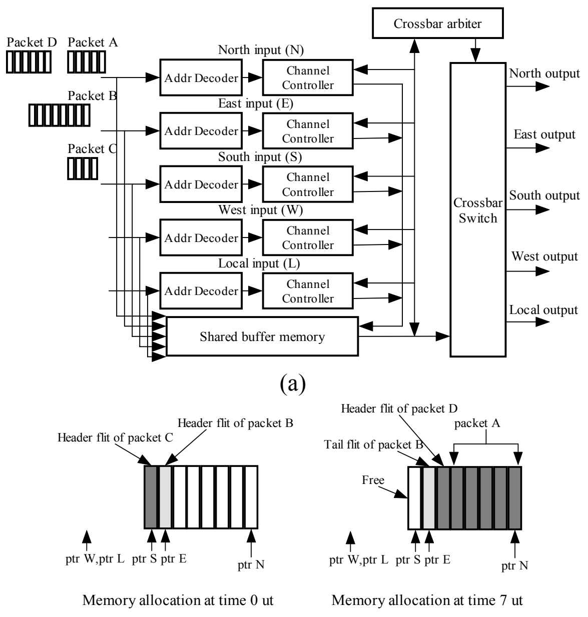
4. SYSTEM CHARACTERIZATION

In this work, we target for NoC with heterogeneous processor elements (PEs). We assume the hard real-time GS applications are periodic, frame-based non-preemptive. The application is represented by a *communication dependence and computation graph* (CDCG) similar to that in [10] which is a directed graph, G(V,E). The CDCG contains *N* periodic tasks $T_i \in V$ and each task T_i has its own worst-case-execution-time (WCET) W_i. We define the mapping function $map(T_i)$ to denote which PE T_i is mapped to. The absolute release time and deadline of task T_i are R_i and D_i, respectively. Each directed arc $E_{ij} \in E$ between T_i and T_j in the CDCG characterizes the communication message. Each message $E_{i,j}$ is divided into packets with variable packet size. PS_{ijm} is the size of the *m-th* packet P_{ijm} of $E_{i,j}$. Also, we use Z_{ijm} to denote the set of the links used in the routing path of the packet P_{ijm}. Similarly, we have Z_{Fijm} and Z_{Lijm} to indicate the first and the last link used in the routing path of the packet P_{ijm}. Suppose packet P_{ijm} pass through a router from the link $e_{k'}$ and directing to the link e_k. Link order $\Re_{ijm}$ of packet P_{ijm}, is a set that contains all the tuple $(e_{k'},e_k)$. Each packet P_{ijm} on a link e_k is characterized by the start-time $ts(P_{ijm,k})$) and end-time $te(P_{ijm,k})$ of the packet using this particular link. Traffic congestion happens when two packets try to use the same link simultaneously. Here, we use first-come-first-served (FCFS) to arbitrate the packets which are going into the same router. A PE can only execute a single task at any time instance. Each PE P_q is connected to a router R_q and each router is connected to five links as shown in Figure 1(b). Each input port that connects to the input link e_k has a buffer size bs_{kq}.

5. SHARED BUFFER ARCHITECTURE

In the router architecture shown in Figure 1b), the buffer of each input port is of the same size. For non-uniform traffic, some of the input port of the router may be less congested and therefore, some of the buffer may be under-utilized. In this work, we extend the shared output buffer proposed in [12] and present a shared buffer architecture which allocates the buffering space for each input port according to the traffics of the applications, thus to maximally utilize the buffering space. The buffer is implemented using multiple-port memory and Figure 3a) shows the proposed router architecture. Each input port has a pair of head and tail pointers (ptr) to indicate the beginning and the end of the corresponding buffer in the memory. The buffer size for each input port can be varied and is determined by solving the ILP formulation. In contrast to the buffer implementation in [8] in which the buffer size for each input port is fixed once the chip is fabricated, our shared buffering scheme allows the buffer size of each input port to be changed when mapping another application onto the NoC.

We use the example in Figure 3 to show how the shared buffer architecture works. Suppose 4 packets A,B,C and D with 5 flits, 8 flits, 4 flits and 6 flits, respectively, reach the router at different time and packets A, B and D are going to output port W(West) while packet C goes to output port E(East). We assume packets B and C reach the router 1 time unit (tu) earlier than the packet A and 7 tu earlier than packet D. In this case, packets A and D are stored at the router and wait for the packet B to release the network resource. We assume the router setup time is 1 tu. Figure 3(b) shows the memory allocation at the



(b) (c)

Figure 3. (a) Proposed router architecture and (b,c) memory allocation

time when only packets B and C arrive at the router. At this moment, the memory contains the header flit of packets B and C. The memory allocation when packet D arrives at the router at time 7 tu is shown in Figure 3(c). In order to avoid the backpressure of the packets, traditional architecture using uniform buffer allocation requires totally 30 flits (6 flits X 5) to store the packets. The shared buffering space scheme requires only a buffer size of 8 flits. The shared buffer architecture requires an extra control circuit to direct the packets to the crossbar switch according to the pointers of each input port. However the increase in complexity is small comparing with the whole router design.

The above problem is formulated as an integer linear programming (ILP) problem. The objective function is to minimize the overall execution time which is equivalent to minimize the end-time of the latest computational task. However, this is not a standard ILP formulation because function *Max*() is endogenous model type which is not allowed inside the objective function [3]. We transform the given CDCG by adding a zero-workload task, T_{dummy}, and connect all tasks to it. By doing so, the T_{dummy} is the only last task to be executed and the objective function is now transformed to

$$Min.\,L = Min.\left(\underset{\forall T_i}{Max}\left(te(T_i)\right)\right) \xrightarrow{transform} Min.\,te(T_{dummy}) \qquad (1)$$

5.1 ILP Formulation Considering Contention with Finite Buffer Size

Marcon et. al. presented a timing model for the wormhole routing algorithm in [10]. However, they assumed the input buffer size is unbounded which cannot be implemented in a real NoC architecture. Since the input buffer size is unbounded, the total packet delay of the wormhole routing algorithm proposed in [10] does not take the packet contention into consideration at the design time. Here, we present a timing model that considers the packet contention with a finite input buffer size bs_{kq}. Let n_{ijm} be the number of flow control units (flits) of the packet P_{ijm} and λ be the period of a clock cycle. Also, let t_r be the number of cycles needed for taking a routing decision in the router and t_l be the number of cycles needed to transmit a flit through a link.

In the following discussion, T_i denotes the current task and $T_{i'}$ is the previous task that executed on the same PE with T_i. Also, let T_j depends on T_i but executed on a PE different from that executing T_i. In addition, we assume there is no communication cost for two tasks that are allocated on the same PE. We use Figure 4 to show the graphical representation of the variables that characterize the packets and how they are used in the timing model. In Figure 4, 3 packets P_{ijm}, $P_{i'j'm'}$ and $P_{i''j''m''}$ arrive at the router R_q in different input ports and all need to be sent to the downstream link $e_{k'''}$. We also assume $P_{i'j'm'}$ reaches R_q first such that P_{ijm} and $P_{i''j''m''}$ are needed to wait at the input port until packet $P_{i'j'm'}$ finishes the transmission and releases the resource of link $e_{k'''}$. The idle time of the links e_k and $e_{k''}$ due to the fact that packets P_{ijm} and $P_{i''j''m''}$ are congested is also shown in Figure 4. The arrival time $ta(P_{ijm})$ of the packet P_{ijm} is the time that the first flit of the packet arrives at the router. The contention delay $d_W(P_{i'j'm',k},P_{i''j''m'',k})$ is the time that packet $P_{i''j''m''}$ needs to wait at the buffer until the previous packet $P_{i'j'm'}$ finishes the transmission and releases the resource e_k. It is defined as the time difference between the end-time of $P_{i'j'm'}$ at e_k and the packet arrival time $ta(P_{i''j''m'',k})$ as shown in Figure 4. To find an optimal link scheduling, we need to know the relative order of the packets on a link before determining the packet transmission time. We use a binary variable o($P_{ijm,k}$, $P_{i'j'm',k}$) to represent the relative order of two packets $P_{ijm,k}$, $P_{i'j'm',k}$ using the same link e_k. **o**($P_{ijm,k}$, $P_{i'j'm',k}$) is given by:

$$o\left(P_{ijm,k},\,P_{i'j'm',k}\right)=\begin{cases}0 \; if\, P_{ijm,k} \; is\ scheduled\ earlier\ than\ P_{i'j'm',k}\\ 1 \; otherwise\end{cases}$$

The value of o($P_{ijm,k}$, $P_{i'j'm',k}$) is related to the start-time and the end-time of the two packets using link e_k. If o($P_{ijm,k}$, $P_{i'j'm',k}$) is 0, then $ts(P_{i'j'm',k})$ should be larger than $te(P_{ijm,k})$ and vice versa. These relations are captured by the following constraints where M is a very large number:

$$te\left(P_{ijm,k}\right)\le ts\left(P_{i'j'm',k}\right)-t_r+o\left(P_{ijm,k},\,P_{i'j'm',k}\right)\cdot M \qquad (2)$$

$$\forall i,i',j,j'\in 1,2,\dots,N,\forall E_{ij},E_{i'j'}\in E,\forall e_k\in Z_{ijm},\forall e_k\in Z_{i'j'm'}$$

$$te\left(P_{i'j'm',k}\right)\le ts\left(P_{ijm,k}\right)-t_r+\left(1-o\left(P_{ijm,k},\,P_{i'j'm',k}\right)\right)\cdot M \qquad (3)$$

$$\forall i,i',j,j'\in 1,2,\dots,N,\forall E_{ij},E_{i'j'}\in E,\forall e_k\in Z_{ijm},\forall e_k\in Z_{i'j'm'}$$

If o($P_{ijm,k}$, $P_{i'j'm',k}$) is 0, eqn. (3) is trivially true and eqn (2) specifies the relationship between the corresponding start time and end time.

The arrival time $ta(P_{ijm})$ is calculated by adding the packet start-time at the uplink $e_{k'}$ with the latency t_l. Thus we have

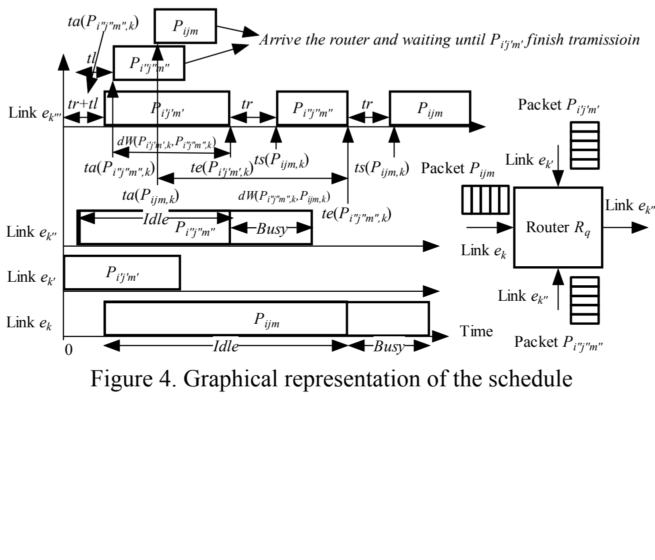


Figure 4. Graphical representation of the schedule

$$ta\left(P_{ijm,}\right)=ts\left(P_{ijm,k'}\right)+t_l \quad (4)$$
$$\forall i,j \in 1,2,...,N, \forall E_{ij} \in E, \forall e_k \in Z_{ijm}, \forall e_{k'} \in Z_{ijm}$$

The definition of the contention delay $d_W(P_{ijm,k},P_{i'j'm',k})$ of P_{ijm} at link e_k is given by:

$$d_W\left(P_{ijm,k},P_{i'j'm',k}\right)=te\left(P_{i'j'm'}\right)-ta\left(P_{ijm}\right) \quad (5)$$
$$\forall i,i',j,j' \in 1,2,...,N, \forall E_{ij},E_{i'j'} \in E, \forall e_k \in Z_{ijm}, \forall e_k \in Z_{i'j'm'}$$

Similar to eqn. (2) and (3), we use the binary variable of $o(P_{ijm,k}, P_{i'j'm',k})$ when calculating the contention delay $d_W(P_{ijm,k}, P_{i'j'm',k})$ in order to specify the order between the packets. Thus we have

$$d_W\left(P_{ijm,k},P_{i'j'm',k}\right)\le te\left(P_{i'j'm'}\right)-ta\left(P_{ijm}\right)+o\left(P_{ijm,k},P_{i'j'm',k}\right)\cdot M \quad (6)$$
$$\forall i,i',j,j' \in 1,2,...,N, \forall E_{ij},E_{i'j'} \in E, \forall e_k \in Z_{ijm}, \forall e_k \in Z_{i'j'm'}$$
$$d_W\left(P_{ijm,k},P_{i'j'm',k}\right)\le te\left(P_{ijm}\right)-ta\left(P_{i'j'm'}\right)+\left(1-o\left(P_{ijm,k},P_{i'j'm',k}\right)\right)\cdot M$$
$$\forall i,i',j,j' \in 1,2,...,N, \forall E_{ij},E_{i'j'} \in E, \forall e_k \in Z_{ijm}, \forall e_k \in Z_{i'j'm'} \quad (7)$$

In order to avoid the backpressure control during packet contention, we need to make sure the buffer bs_{kq} of the router R_q should be able to store the flits coming to the input port k during contention time. Also, the sum of the buffer size of all the input ports is equal to the total buffer budget of each router

$$d_W\left(P_{ijm,k},P_{i'j'm',k}\right)\le bs_{k'q}\cdot t_l\cdot\lambda \qquad \forall i,i',j,j' \in 1,2,...,N,$$
$$\forall E_{ij},E_{i'j'} \in E, \forall e_k,e_{k'} \in Z_{ijm}, \forall e_k,e_{k'} \in Z_{i'j'm'}, \forall (e_{k'},e_k) \in \Re_{i'j'm'} \quad (8)$$

$$\sum_{\forall k} bs_{kq} = BS_q \quad (9)$$

The actual start-time of the packet P_{ijm} on the link e_k is larger than the arrival time plus the router setup time and the wait time. So we have

$$ts\left(P_{ijm,k}\right)\ge ta\left(P_{ijm,k}\right)+t_r+d_W\left(P_{i'j'm',k},P_{ijm,k}\right)$$
$$\forall i,i',j,j' \in 1,2,...,N, \forall E_{ij},E_{i'j'} \in E, \forall e_k \in Z_{ijm}, \forall e_k \in Z_{i'j'm'} \quad (10)$$

The end-time of the packet P_{ijm} on the link e_k is equal to

$$te\left(P_{ijm,k}\right)=ts\left(P_{ijm,k}\right)+t_l\cdot ni_{jm}\cdot\lambda \quad (11)$$
$$\forall i,j \in 1,2,...,N, \forall E_{ij} \in E, \forall e_k \in Z_{ijm}$$

In the above, we show how the communication characteristics on the applications are captured in the ILP formation. We also need to consider the scheduling and timing of the computation tasks when we determine the actual packet schedule. In hard real time applications, the task T_i can start execution after the ready time R_i and has to finish before the deadline D_i in order to guarantee the performance constraints. The end-time $te(T_i)$ is the start-time $ts(T_i)$ plus the WCET W_i. Thus we have

$$R_i \le ts(T_i), te(T_i) \le D \qquad \forall i \in 1,2,...,N \quad (12)$$
$$te\left(T_i\right)=ts\left(T_i\right)+W_i \qquad \forall i \in 1,2,...,N \quad (13)$$

Considering two tasks mapping to the same processor, we need to make sure that T_i start the execution only after its preceding task $T_{i'}$ finish the execution. We have

$$ts\left(T_i\right)\ge te\left(T_{i'}\right) \ \forall i \in 1,2,...,N, map\ (T_i)=map\ (T_{i'}) \quad (14)$$

In the case that two dependent tasks T_i and T_j are executed on two different PEs, T_i sends the message to T_j when T_i finishes the execution and the transfer of the first packet of the message starts when the execution of T_i is finished. When the last packet reaches the destination, task T_j can start the execution.

$$ts\left(P_{ijm,k}\right)\ge te\left(T_i\right)\forall i \in 1,2,...,N, \forall e_k \in Z_{Fijm} \quad (15)$$
$$ts\left(T_i\right)\ge te\left(P_{ijm,k}\right)\forall i \in 1,2,...,N, \forall e_k \in Z_{Lijm} \quad (16)$$

The resulting schedule after solving the proposed ILP formulation does not need the backpressure control mechanism to regulate the packets during packet contention. It is because if a packet is congested at the router, the packet can actually be delayed to be sent out at the source in order to avoid the operation of backpressure. The time that the packet can start the transfer greatly depends on the buffer size available at the input port of the router. The proposed ILP will automatically determine all the variables at the same time and find the best schedule such that the overall execution time is minimum.

6. EXPERIMENTAL RESULTS

6.1 Experiments Using Real Applications

We carried out experiments using several real applications ranging from video processing applications to industrial applications. These include a *computer numerical control applications* (*CNC*) (62 tasks) [5], a generic *Multimedia System* (*MMS*) (40 tasks) [8], a *Video Object Plane Decoder* (16 tasks) [7] and four benchmarks applications obtained from E3S benchmark suites [4], namely *networking* (13 tasks), *telecom* (30 tasks), *auto_indust* (24 tasks) and *consumer* (12 tasks). For each benchmark, we manually mapped the tasks onto a 4x4 tile-based NoCs and used XY routing for the communication link routing. We evaluate the performance of the proposed ILP algorithm on both the overall execution time and the link utilization. We compare the results with the work proposed in [10] where packets are sent out to the router as soon as they are ready at the PE and backpressure mechanism is used when there is packet contention. However, [10] assumes an infinite buffer size which is impossible in real implementation. We modified it by using a uniform buffer implementation of which the size of the buffer at every input port is the same. The ILP is implemented in the GAMS [3] environment and it is solved by Cplex [3].

In this experiment, a message is divided into several packets with the packet size ranging from 16 flits to 64 flits and the flit size is 32 bits. We studied the effect of different traffic patterns on the performance. Here two traffic patterns, namely uniform and hotspot, were generated by using different task mappings. For uniform traffic pattern, a task is randomly mapped on the PE with equal probability. For hot spot traffic pattern generation, the tasks are restricted to be mapped on certain PEs such that a hotspot traffic is generated. In hotspot distribution, the probability that a packet is sent to some nodes (called hotspot nodes) follows a normal distribution with mean is four times of the other nodes. Table 1 shows the improvement of the overall execution time and the link utilization over the work in [10] under different traffic patterns. Here the buffer budget for each router is 40 flits. We can see that the improvements varied with the applications because the traffic characteristics of the

Table 1. Results for real applications

Application	Number of tasks	Improvement of overall execution (40 flits buffer size /router)		Improvement of link utilization reduction (40 flits buffer size /router)	
		hotspot	uniform	hotspot	uniform
CNC	64	11.7%	7.0%	14.3%	4.9%
MMS	40	7.2%	5.9%	12.7%	8.5%
Video Object Plane Decoder	16	10.2%	2.8%	7.8%	5.6%
Networking	13	15.3.%	11.8%	18.3%	3.3%
Telecom	30	30.0%	13.8%	20.9%	13.4%
Auto_indust	24	22.3%	9.1%	20.2%	14.5%
Consumer	12	17.2%	9.5%	6.8%	4.3%

applications are different. Also the improvements under the hotspot traffic are more than that under the uniform traffic. This is because the traffic is more congested and packet contentions occur more frequently. Since the contention is regulated by the backpressure mechanism in [10], the upstream links in the routing path reserved for the packet are idle and wasted. However, the proposed optimal link scheduling can reduce the amount of contention and improve the link utilization by judiciously scheduling and prioritizing the packets. In contrast, the traffic is evenly spread under uniform traffic and less packet contention occurs. Therefore, less improvement was observed. Because of improvement in link utilization, the performance of the BE applications can be improved as the remaining communication resources available for the BE applications is increased.

We also evaluate the improvement of the overall execution time due to the use of the shared-buffer router architecture. Here, we compare the overall execution time using optimal link scheduling with shared-buffer architecture to that using optimal link scheduling with uniform buffer architecture given the same total buffer size of each router. The experimental setup is similar to the previous simulation on a hotspot traffic pattern. Table 2 shows the improvement under hot spot traffic. It can be seen the overall execution time is improved by a few percent without any additional cost incurred.

Table 2. Evaluations of shared –buffer architecture

Application	CNC	MMS	Video Object Plane Decoder	Networking	Telecom	Auto_indust	Consumer
Improvement on overall execution time	7.0%	4.2%	1.9%	2.0%	4.9%	5.0%	2.7%

6.2 Experiments Using Artificial Applications

In this experiment, we evaluate the improvement by using 1000 randomly generated applications. The applications were constructed by using TGFF [2] and each application has about 50 computation tasks. TGFF is used to generate the task characteristic assuming normal distribution for the WCET. Each task is randomly allocated onto a 4X4 tile-based NoC to generate uniform and hotspot traffic patterns. Table 3 shows the average execution time reduction under different buffer sizes of each router over the work in [10]. Similar to the results shown in Table 1, we can see that the improvement under hotspot traffic is always higher than that under uniform traffic. In addition, it shows that a larger improvement is obtained when the buffer size is smaller. It is because larger buffers help to resolve the network contention and narrow its performance gap between the proposed ILP method and the ASAP scheduling in [10]. We also show the link utilization reduction under different router buffer budget in Table 4. Similar to the results in Table 1, a larger improvement is achieved when the buffer size is small. We also evaluate the improvement of using the shared buffer architecture for randomly generated applications. The average link utilization reduction under hot sport traffic is 7.1% and it also shows that the proposed shared buffer architecture helps to reduce the overall execution time.

Table 3. Results of max. latency red. for random applications

Traffic pattern	Number of tasks	Improvement on overall execution time		
		20 flits buffer size /router	40 flits buffer size /router	160 flits buffer size /router
Uniform	50	10.5%	9.3%	9.3%
Hotspot	50	17.6%	16.6%	15.9%

The running time of our algorithm is reasonable. For instance, for systems with 4×4 tiles and 50 tasks, the average run-time of our algorithm is a few minutes when running on a Pentium-4 2.8GHz machine. The actual solving time is varied with the application characteristics, e.g. the number of tasks and number of packets and the task allocation.

Table 4. Results of link utilization red. for random applications

Traffic pattern	Number of tasks	Improvement of link utilization reduction		
		20 flits buffer size /router	40 flits buffer size /router	160 flits buffer size /router
Uniform	50	3.7%	3.2%	1.9%
Hotspot	50	19.2%	13.5%	12.4%

7. Conclusions

We have proposed an optimal link scheduling and a shared-buffer router architecture which strive to minimize the overall execution time and link utilization of the applications that have hard real-time requirement. By using this scheme, more communication resource can be used for the BE communication.

8. References

[1] W. J. Dally et. al. "Route Packets, Not Wires: On-Chip Interconnection Networks," in the Proc. of DAC, pp. 684-689, 2001.

[2] http://helsinki.ee.Princeton.EDU/ dickrp/tgff

[3] A.Brooke et al., "GAMS A user's guide,"1998.

[4] R. Dick. Embedded system synthesis benchmarks suites (E3S). http://helsinki.ee.princeton.edu/_dickrp/e3s/.

[5] Kim, N. et al., "Visual Assessment of a Real-time System Design: A Case Study on a CNC Controller," RTSS, 1996.

[6] T. Dumitras and R. Marculescu, "On-chip stochastic communication," In DATE, pp. 790-795, 2003.

[7] S. Murali and G. De Micheli, "Bandwidth-constrained mapping of cores onto NoC architectures," in Proc. DATE, pp. 16-20, Feb. 2004.

[8] J. Hu, R. Marculescu, "Application-Specific Buffer Space allocation for Networks-on-Chip Router Design," in the Proc. of ICCAD, pp. 354 - 361, 2004

[9] J. Duato, S. Yalamanchili, L. Ni, Interconnection Networks, an Engineering Approach, IEEE Computer Society Press, 1997.

[10] Cesar Marcon et al., "Exploring NoC Mapping Strategies: An Energy and Timing Aware Technique," in the Proc. of DATE, pp. 502-507, 2005.

[11] Tobias Bjerregaard and Jens Sparsø,"A Scheduling Discipline for Latency and Bandwidth Guarantees in Asynchronous Network-on-Chip," in the Proc. of ASYNC, pp. 34-43, 2005.

[12] P. Guerrier and A. Greiner, "A generic architecture for on-chip packet-switched interconnections," in the Proc. of DATE, pp. 250–256, 2000.

[13] E. Rijpkema et.la., "Trades-offs in the design of a router with both guaranteed and best-effort services for networks on chip," in the Proc. On Comput. Digit. Tech. Vol. 150, Issue 5, pp. 294-302, Sept. 2003

[14] W.J. Dally, H. Aoki, "Deadlock -free adaptive routing in multicomputer networks using virtual channels," IEEE Transactions on Parallel and Distributed Systems pp. 466–475, April, 1993

[15] V. S. Adve and M. K. Vernon, "Performance analysis of mesh interconnection networks with deterministic routing," IEEE Tran. on Parallel and Distributed Systems, pp.225.246, March 1994.