ECTM: A New Communication Model to Network-On-Chip Schedulability Analysis

Mourad Dridi[1], Frank Singhoff[1], Stéphane Rubini[1], and Jean-Philippe Diguet[2]

[1] Univ. Brest, Lab-STICC, CNRS, UMR 6285, F-29200 Brest, France
[2] Univ. Bretagne Sud, Lab-STICC, CNRS, UMR 6285, F-56100 Lorient, France
{mourad.dridi, stephane.rubini, frank.singhoff}@univ-brest.fr
jean-philippe.diguet@univ-ubs.fr

Abstract. Network-On-chips are widely used in industrial applications since they provide communication parallelism and reduce energy consumption. The use of NoC has been recently extended to real-time systems, whose execution has to meet temporal constraints. Communication delays introduced by the network make the scheduling analysis challenging. In this paper, we propose a new NoC communication model called ECTM. The main goal of this model is to assess the schedulability of dependent periodic tasks exchanging messages on NoC. ECTM transforms NoC messages to tasks in order to take into account communication delays during scheduling analysis. It supports Store-And-Forward and Wormhole NoC. We have implemented ECTM in a real-time scheduling analysis tool called Cheddar and performed experiments to assess its efficiency. ECTM is more efficient than existing solutions with an improvement of 30% for Store-And-Forward NoCs and of 100% for Wormhole NoCs, while the proposed model requires a larger computation time about 17% for Store-And-Forward NoCs and 54% for Wormhole NoCs.

Keywords: Network-On-Chip (NoC) · Real-Time Systems · Communication · Scheduling Analysis · Wormhole · Store and forward

1 Introduction

Since they provide communication parallelism and limit the energy consumption, Networks on Chip (NoC) are widely used in industrial applications. Recently, the use of NoC has been extended to real-time systems [12]. Real-time systems are systems that have deadlines to met [3].

NoC introduce communication delays because of possible resource contentions between different flows in the network. Those delays depend on many factors.

This work and Cheddar are supported by Brest Métropole, Ellidiss Technologies, CR de Bretagne, CD du Finistère and Campus France PESSOA programs number 27380SA and 37932TF.

Some of them are related to the NoC configuration like the switching mode or the arbitration policy, while others result from the network state [17]. Thus, due to the NoC architecture, real-time scheduling analysis is a challenge.

In order to analyse schedulability of periodic tasks in NoC-based parallel architectures, we have to take into account at the same time the task scheduling over processing elements and the message communications scheduling over the network.

Unfortunately, classic multiprocessor real-time scheduling solutions consider worst case communication time instead of the actual delays introduced by the network [14]. Consequently, it leads to pessimistic analysis results.

In this paper, we propose a NoC communication model in order to assess the scheduling analysis of periodic tasks over NoC architectures. The proposed model simplifies combined scheduling analysis of the periodic tasks and the NoC communications.

Our approach converts NoC flows of messages scheduling to periodic tasks scheduling. Each flow of message is transformed to a set of dependent periodic tasks. As a result, the scheduling analysis is simplified as the problem of the tasks and NoC communications scheduling is similar to the scheduling of periodic tasks deployed on a multiprocessor, which has been well studied [15, 1]. By a set of experiments, we show the efficiency of this approach.

The remainder of the paper is organized as follows. The next section presents related works. Section 3 introduces background about the NoCs and the periodic task model we consider. Then, section 4 proposes our approach for scheduling periodic tasks over NoC-based architectures. Implementation and evaluation of the proposed approach are explained in section 5 and section 6 concludes this article.

2 Related Work

NoC-based communication has recently attracted significant attention because of its potential for performance improvement and its impact on task scheduling. Thus, several NoC communication analysis have been proposed.

[18] proposes an analysis approach for real-time on chip communication with Wormhole switching and fixed priority scheduling. [19] focuses on real-time communication service with a priority share policy. These works designed worst case communication time analysis for different NoC configurations.

In order to schedule periodic tasks over NoC architectures, several scheduling algorithms have been proposed. Those solutions have different goals. Some optimize the design by minimizing the power consumption and the run time of an application. Others enforce the timing constraints of the system [13].

G. Varatkar et al. [21] have developed a two-step mapping and scheduling algorithm. It performs simultaneous mapping and scheduling of tasks in order to reduce communication energy by minimizing the inter-processor communication. However, the communication distance is only roughly approximated. This

work does not carry out communication mapping and scheduling. Thus, real-time scheduling cannot supported, because communication latency is completely ignored in the scheduling process.

T. Lei et al. [11, 10] also propose a two-step algorithm for task mapping and task scheduling over NoC architectures. The goal of the scheduling analysis is to check hard deadlines, while the goal of task mapping is to maximize timing performances. The communications are not considered in the mapping and the scheduling process. Communication delays are estimated using average distance in the NoC. Thus, this approach cannot guarantee hard deadlines also.

The analysis approach we propose in the paper starts with a model transformation. The model transformation aims to convert an architectural model into a simplified analysis model [16].

A model transformation approach has been used by Lakshmanan et al in [9] also. They propose a stretching algorithm for a dependent task model. The stretching algorithm avoids the parallel structure of the architectural model by executing them as sequentially as possible. [16] introduces a DAG Stretching algorithm in order to analyze the scheduling of dependent tasks. In the stretching algorithms, dependent tasks are transformed to a set of independent sequential threads. Intermediate offsets and deadlines are assigned to threads so as to determine their execution interval. However, all these transformation models do not consider the communications in a NoC.

To conclude, there are few approaches which perform NoC communication scheduling [13]. Furthermore, most of the proposed works for task and communication scheduling over NoC architectures do not consider the exact communication time. They consider worst case communication time or average distance in the NoC to estimate the communication time. Finally, existing transformation models also ignore the NoC communications.

The next sections detail a new NoC communication model that addresses the limitations highlighted above.

3 Background

This section contains the background required to understand the contributions proposed in the next sections. First, we introduce the main concepts about NoCs. Then, we present the models of flow and task assumed in this article.

3.1 Network-on-Chip

A NoC is a network of nodes. Each node may be a processing element, a memory, a peripheral or a cluster.

Fig. 1 illustrates the major components of a NoC. Nodes access the network through a network interface (NI) and receive data encapsulated in packets [12]. Then, processing elements communicate by exchanging messages over the network. The network is composed of routers and unidirectional physical links. We note e_{RxRy} the unidirectional link between the two routers R_x and R_y, and

e_{PExRy} the unidirectional link between the processing element PE_x and the router R_y. Finally, e_{PExRy} is an unidirectional link between the router R_y and the processing element PE_x.

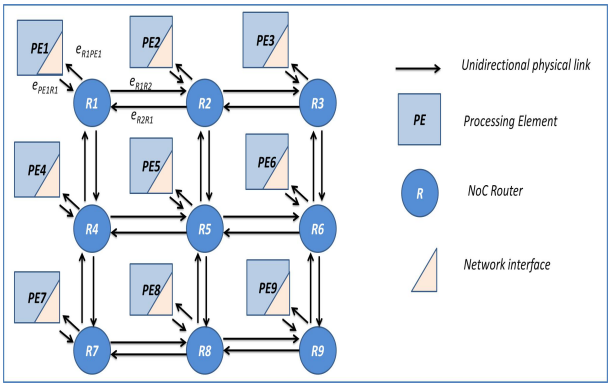

Fig. 1. Network On Chip

In order to ensure the communication in the network, routers implement arbitration policy and switching mode.

Switching Mode: it determines how a packet is allocated to buffers and channels and when it will receive service. In this article, we assume two switching modes: Store And Forward (SAF) and Wormhole.

For the SAF mode, each router waits for a full packet to arrive before sending it to the next router [7].

With the Wormhole mode, the packet is divided into a number of fixed size flits [17]. The packet is split into an header flit, one or several body flits and a tail flit. The header flit stores the routing information and is used to build the route. As the header flit moves ahead along the selected path, the remaining flits follow in a pipeline way and possibly span over multiple routers.

Arbitration Policy: its aim is to select one packet from many in a router. When several incoming packets request the same output port of a router, an arbitration is required to select one of these packets.

Several arbitration mechanisms have been used in NoC routers [12]. Round-robin and priority-based are 2 examples of these mechanisms. Round-robin arbiters give the lowest priority to the last served request in the next arbitration. Priority-based arbiters choose one packet from many requests based on their fixed priority.

3.2 Task model

In this paper, we assume real-time systems designed as a set of dependent periodic tasks deployed on a NoC.

We assume a set of task Γ composed of n periodic tasks: $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\}$ Each task τ_i is defined as follows: $\tau_i = \{ O_i, T_i, C_i, D_i, \Pi_i, Node_i, E_i \}$ where:

- O_i is the first release time of the task τ_i.
- T_i is the period of the task.
- C_i specifies the Worst Case Execution Time of the task.
- D_i is the deadline to meet.
- Π_i is the fixed priority of the task. The value 1 denotes the highest priority level while a larger value is a lower priority.
- $Node_i$ identifies the NoC node (i.e. the processing element) running the task. This parameter allows us to introduce the mapping configuration, i.e. on which processing element each task is assigned to.
- $E_i : \Gamma \longrightarrow \Gamma^p \ \tau_i \longrightarrow E(\tau_i) = \{\tau_j, \ldots, \tau_k \}$
  The function E introduces precedence constraints of the task model. For a given task τ_i, the function E determines all the tasks τ_j that receive messages from the task τ_i.

From the previous task model, in [4], we have proposed DTFM (Dual Task and Flow Model) in order to compute the flow model from the task model, the mapping of the tasks on the processing elements and the NoC model. The flow model specifies the messages that have to be transmitted in the NoC to enforce the communications between the tasks.

With DTFM, the set of flows ψ related to Γ can be defined as follow. The flow model comprises m periodic traffic flows $\psi = \{\rho_1, \rho_2, \ldots, \rho_m\}$.

Each flow ρ_i raises a sequence of messages in a similar way that a task raises a sequence of jobs. Each flow ρ_i is defined as follows: $\rho_i = \{ O_i, T_i, D_i, \Pi_i, NodeS_i, NodeD_i, F_i \}$

where:

- O_i is the release time of the messages, i.e. the first time when a message of the flow becomes ready to be transmitted.
- T_i is the period of the message.
- D_i is the deadline of the message.
- Π_i is the priority of the messages. The value 1 denotes the highest priority level while a larger value indicates a lower priority.
- $NodeS_i$ is the node (i.e. the processing element) running the transmitter task.
- $NodeD_i$ is the node running the receiver task.
- $F_i : \psi \longrightarrow \Omega^p$
  $\rho_i \longrightarrow F(\rho_i) = \{e_{Rp,Rk}, \ldots, e_{PEi,Rj}\}$
  where Ω is a set of physical links in the NoC.
  F_i is a function that computes the links used by a flow. In other words, F_i identifies the physical links that will be used by any messages of the flow.
  The function F_i helps us to understand the relationships between the various flows that transit through the network and to determine the interferences between messages sent by the tasks.

Next, we explain the proposed approach.

4 NoC Communication Time Models

In order to analyze the scheduling of periodic tasks running on a NoC architecture, communication delays in the NoC have to be investigated.

Shi et al. in [17–19] proposed NoC communication models based on worst case scenario. Those communication models produce worst case communication delays that can be used to assess schedulability of periodic tasks running on the NoC.

In the sequel, we formalize WCCTM, the Worst Case Communication Time Model based of the Shi. et al analysis. Then, we propose ECTM (Exact Communication Time Model), an exact communication model that improves WCCTM.

Next, we present an overview of our schedulability approach before introducing both WCCTM and ECTM.

4.1 Overview of our approach

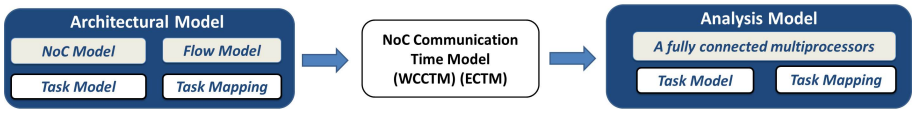

Fig. 2. Overall approach

Fig. 2 shows an overview of the approach we propose.

We assume the system to analyze is expressed by a model of the NoC, a model of the application as a set of dependent periodic tasks and how such tasks are deployed on the processing elements. In the sequel, we call such input model the architectural model.

Then, to achieve schedulability analysis of the overall system, the communication delay model of the architectural model is transformed to an analysis model to validate the scheduling of the communications. For such transformation, we use ECTM and WCCTM. WCCTM is used in this article to evaluate ECTM efficiency.

In the next paragraphs, we define the architectural model and the analysis model.

Architectural model : For the architectural model, we consider a set of periodic tasks exchanging messages deployed over a NoC. We use DTFM [4] in order to model the NoC model, the task mapping, and the flow model.

DTFM generates the flow model from the task model, the mapping of the tasks on the processing elements and the NoC model.

Analysis model : The analysis model is a set of periodic tasks running on a multiprocessor execution platform. We assume a multiprocessor with identical processors [5] and no shared resources.

Scheduling analysis of such model can be performed with list scheduling [6, 15]. List scheduling algorithms build scheduling list of tasks when assigning them their priorities. There are several ways to determine the priorities of tasks such as Highest Level First, Longest Path and Longest Processing Time [8].

Highest Level First with Estimated Time (HLFET) algorithm, Modified Critical Path (MCP) algorithm, Earliest Time First (ETF) algorithm and Dynamic Level Scheduling (DLS) algorithm are examples of list scheduling algorithms which we can be applied on the analysis model we assume [15, 1].

Now we describe the communication time models considered in this article: WCCTM and two ECTM models, one for SAF switching and the second for Wormhole switching. Table 1 gives all the assumptions for these communications models.

Noc Communication Model	WCCTM	$ECTM_{SAF}$	$ECTM_{Wormhole}$
Topology and dimension	2D mesh	2D mesh	2D mesh
Routing Algorithm	XY / YX	XY / YX	XY / YX
Switching mode	Wormhole / SAF	SAF	Wormhole
Arbitration policy policy	Fixed priority Round Robin	Fixed priority Round Robin	Fixed priority Round Robin
Virtual channel	With / without	With	With
Preemption level	Flit / Packet	Packet	Flit

Table 1. Assumptions on the NoC for each communication model

4.2 Worst Case Communication Time Model (WCCTM)

Now we explain how the architectural model is transformed to an WCCTM analysis model. To achieve this transformation, we apply the following rules:

- All routers and unidirectional links of the architectural model will be removed in the analysis model while keeping all the processing elements.
- For each flow ρ_i of the architectural model, we consider a new processing element PE_{ρ_i} in the analysis model.
- Each flow ρ_i of the architectural model will be replaced by one task τ_{ρ_i} in the analysis model.
  If the architectural model contains two periodic tasks τ_{source} and $\tau_{destination}$ and if τ_{source} sends the flow ρ_i to $\tau_{destination}$, then applying WCCTM leads to the flow ρ_i transformed in the analysis model as a periodic task τ_{ρ_i} with the following parameters:

- $O_{\tau_{\rho_i}} = O_{\rho_i}$
- $T_{\tau_{\rho_i}} = T_{\rho_i}$
- $C_{\tau_{\rho_i}} = WCCT_i$
  Where $WCCT_i$ is the Worst Case Communication Time of a ρ_i's message. We compute $WCCT_i$ with the methods proposed in [17–19] for Wormhole NoCs or in [17] for SAF NoCs.
- $D_{\tau_{\rho_i}} = D_{\rho_i}$
- $Node(\tau_{\rho_i}) = PE_{\rho_i}$
- $E_{(}\tau_{\rho_i})= \tau_{destination}$

We illustrate the WCCTM transformation by the Fig. 3 example. We consider two periodic tasks τ_1 and τ_2. τ_1 sends the flow ρ_1 to the task τ_2. In addition, τ_1 is executed by the processing element PE_1, while τ_2 is executed by PE_3.

After applying the WCCTM transformation rules, routers and links are removed. The flow ρ_1 is transformed into a task τ_{ρ_1}. τ_{ρ_1} is modeled by the processing element PE_{ρ_1}.

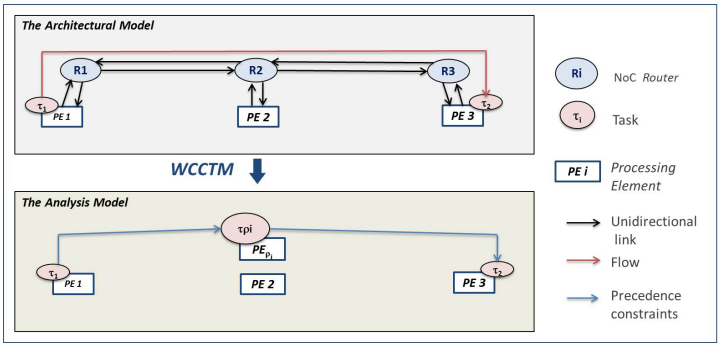

Fig. 3. Worst Case Communication Time Model (WCCTM): Example

4.3 Exact Communication Time Model for SAF NoC ($ECTM_{SAF}$)

For SAF NoCs, we propose $ECTM_{SAF}$. Next, we detail the transformation rules of $ECTM_{SAF}$.

- Each router of the architectural model will be removed in the analysis model while keeping all the processing elements.
- Each unidirectional link in the network between two routers (e_{RxRy}) of the architectural model will be replaced in the analysis model by a new processing element (PE_{RxRy}).
- Each unidirectional link in the network between router and processing element (e_{PExRy}) (respectively e_{RxPEy}) of the architectural model will be replaced in the analysis model by a new processing element (PE_{PExRy}) (respectively PE_{RxPEy}).

– Each flow ρ_i of the architectural model will be replaced by a set of $nbrlink_i$ tasks Γ_{ρ_i}, where $nbrlink_i$ denotes the number of links used by the flow ρ_i.
$\Gamma_{\rho_i} = \{ \tau_{\rho_{i,1}}, \tau_{\rho_{i,2}}, \ldots, \tau_{\rho_{i,nbrlink_i}} \}$.
If the architectural model contains two periodic tasks τ_{source} and $\tau_{destination}$ and if τ_{source} sends the flow ρ_i to $\tau_{destination}$, then applying $ECTM_{SAF}$ leads to the flow ρ_i transformed in the analysis model as periodic taskset Γ_{ρ_i}. Parameters of each task $\tau_{\rho_{i,j}}$ of the task set Γ_{ρ_i} are computed as follow. For j in [1, $\ldots nbrlink_i$] $\tau_{\rho_{i,j}}$ is characterized by :

- $O_{i,j} = O_{\rho_i}$
- $T_{i,j} = T_{\rho_i}$
- $C_{i,j} = PD_{1Link}$
  PD_{1Link} is the Path Delay of one link.
  The path delay represents the communication delay for a given flow ρ_i
  PD_{1Link} represents the communication delay for the transmission of one flow over one link without considering the possible conflicts in the network.
- $D_{i,j} = D_{\rho_i}$
- $Node(\tau_{i,j}) = \begin{cases} PE_{R_j PE_j} \; if \; j = nbrlink_i \\ PE_{PE_j R_j} \; if \; j = 1 \\ PE_{R_x R_y} \quad if \; 1 < j < nbrlink_i \end{cases}$
- $E_{i,j} = \begin{cases} \tau_{i,j+1} \qquad if \; j < nbrlink_i \\ \tau_{destination} \; if \; j = nbrlink_i \end{cases}$

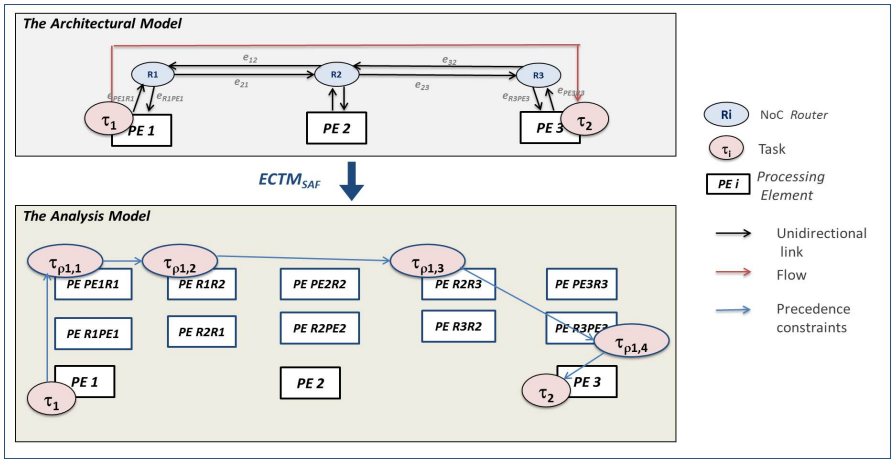

Fig. 4. Example of $ECTM_{SAF}$

We apply the $ECTM_{SAF}$ transformation, to the previous example. Fig. 4 shows the architectural and the analysis models with $ECTM_{SAF}$.

After applying the $ECTM_{SAF}$ transformations, routers have been removed and the physical links are transformed in the analysis model to processing elements. The flow ρ_1 uses 4 physical links. Thus, the flow ρ_1 has been transformed in the analysis model to a task set Γ_{ρ_1} of 4 tasks. $\Gamma_{\rho_1} = \{ \tau_{\rho_{1,1}}, \tau_{\rho_{1,2}}, \ldots, \tau_{\rho_{1,4}} \}$.

$\tau_{\rho_{1,1}}$, $\tau_{\rho_{1,2}}$, $\tau_{\rho_{1,3}}$ and $\tau_{\rho_{1,4}}$ are respectively executed by the processing element PE_{PE1R1}, PE_{R1R2}, PE_{R2R3} and PE_{R3PE3}.

4.4 Exact Communication Time Model for Wormhole NoC ($ECTM_{Wormhole}$)

For Wormhole NoCs, we propose $ECTM_{Wormhole}$. In the following, we detail rules of $ECTM_{Wormhole}$:

- Each router of the architectural model will be removed in the analysis model while keeping all the processing elements.
- Each unidirectional link in the network between two routers (e_{RxRy}) of the architectural model will be replaced in the analysis model by a new processing element (PE_{RxRy}).
- Each unidirectional link in the network between router and processing element (e_{PExRy}) (respectively e_{RxPEy}) of the architectural model will be replaced in the analysis model by a new processing element (PE_{PExRy}) (respectively PE_{RxPEy}).
- Each flow ρ_i of the architectural model will be replaced by a set of nb tasks Γ_{ρ_i}, where $nb = nblink_i \times nbsize_i$
  $nblink_i$ represents the number of links used by the flow while $nbsize_i$ represents the size of message of the flow ρ_i.
  $\Gamma_{\rho_i} = \{ \tau_{\rho_{i,1,1}}, \tau_{\rho_{i,a,b}}, \ldots, \tau_{\rho_{i,nbsize_i,nblink_i}} \}$.
  For example, with $ECTM_{SAF}$, a flow of 2 flits which uses 3 physical links will be transformed in the analysis model to a task set of 2×3 tasks.

If the architectural model contains two periodic tasks τ_{source} and $\tau_{destination}$, τ_{source} and $\tau_{destination}$ are executed respectively into PE_s and PE_d. If τ_{source} sends the flow ρ_i to $\tau_{destination}$, then applying $ECTM_{Wormhole}$ leads to the flow ρ_i transformed in the analysis model as periodic taskset Γ_{ρ_i}. Parameters of each task $\tau_{\rho_{i,a,b}}$ of the task set Γ_{ρ_i} are computed as follow :

For (a,b) $\in$ [1, $size_i$] x [1,$nbrlink_i$], $\tau_{\rho_{i,a,b}}$ is characterized by :

- $O_{\tau_{\rho_i,a,b}} = O_{\rho_i}$
- $T_{\tau_{\rho_i,a,b}} = T_{\rho_i}$
- $C_{\tau_{\rho_i,a,b}} = PD_{Oneflit/Onelink}$
  $PD_{Onelink}$ is the path delay of one flit over only one link.
- $D_{\tau_{\rho_i,a,b}} = D_{\rho_i}$
- $Node(\tau_{\rho_i,a,b}) = \begin{cases} PE_{PE_sR_s} \; if \; a = 1 \\ PE_{R_xR_y} \quad if \; 1 < a < nbrlink_i \\ PE_{R_dPE_d} \; if \; a = nbrlink_i \end{cases}$
  We note here that e_{RxRy} presents one of the used physical links by the flow ρ_i.

$$- \ E_{\tau_{\rho_i,a,b}} = \begin{cases} \tau_{\rho_i,a+1,b}, \tau_{\rho_i,a,b+1} & if \ a < size_i \ and \ b < nbrlink_i \\ \tau_{\rho_i,a+1,b} & if \ a < size_i \ and \ b = nbrlink_i \\ \tau_{\rho_i,a,b+1} & if \ a = size_i \ and \ b < nbrlink_i \\ \tau_{destination} & if \ a = size_i \ and \ b = nbrlink_i \end{cases}$$

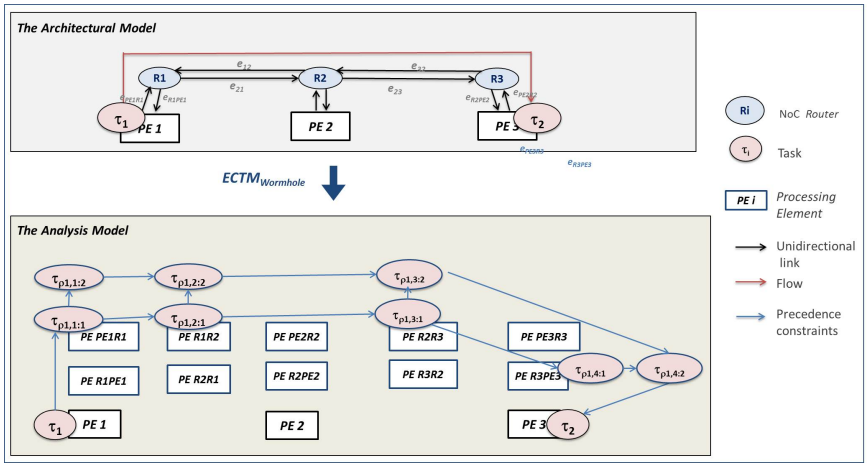

Fig. 5. Exact Communication Time Model for Wormhole: Example

In order to illustrate transformation of the proposed model for Wormhole NoCs, we apply $ECTM_{Wormhole}$ to the previous example. Fig 5 shows the architectural and the analysis models with $ECTM_{Wormhole}$.

After applying $ECTM_{Wormhole}$ transformation, routers are removed. Physical links are transformed in the analysis model to processing elements. The flow ρ_1 uses 4 physical links. The flow size is 2 flits. Thus, the flow ρ_1 is transformed in the analysis model to a task set Γ_{ρ_1} of 8 tasks. $\Gamma_{\rho_1} = \{ \tau_{\rho_{1,1,1}}, \tau_{\rho_{1,1,2}}, \ldots, \tau_{\rho_{1,4,2}} \}$.

We note here that for n flows, ECTM generates a task set of n^2 tasks for both Wormhole and SAF NoCs, which may lead to a more expensive schedulability analysis.

In order to validate the proposed approach, we have demonstrated theoretically the correctness of the transformations of ECTM. First, we have identified properties of both architectural and analysis models. Then, we have shown equivalence between the two properties set. Unfortunately, we cannot expose the validation details in this article because of the page limitations.

In the next section, we present the evaluation of the proposed approach.

5 Implementation and Evaluation

We have produced several experiments in order to evaluate the accuracy and the scalability of the proposed models. To perform these evaluations, we have implemented our communication models into a real-time scheduling simulator called Cheddar [20]. Then, with a first experiment, we evaluate the accuracy of ECTM and WCCTM. A second experiment evaluates the scalability of our approach.

5.1 Implementation of ECTM and WCCTM

Cheddar is a GPL framework that provides a scheduling simulator, schedulability tests and various features related to the design and the scheduling analysis of real-time systems. To model the system to analyze, Cheddar provides a specific architecture design language, called CheddarADL [20]. CheddarADL allows users to describe both the software and the hardware parts of the system they expect to analyze.

Fig 6 is an overview of the software architecture of our prototype and a subset of Cheddar framework libraries.

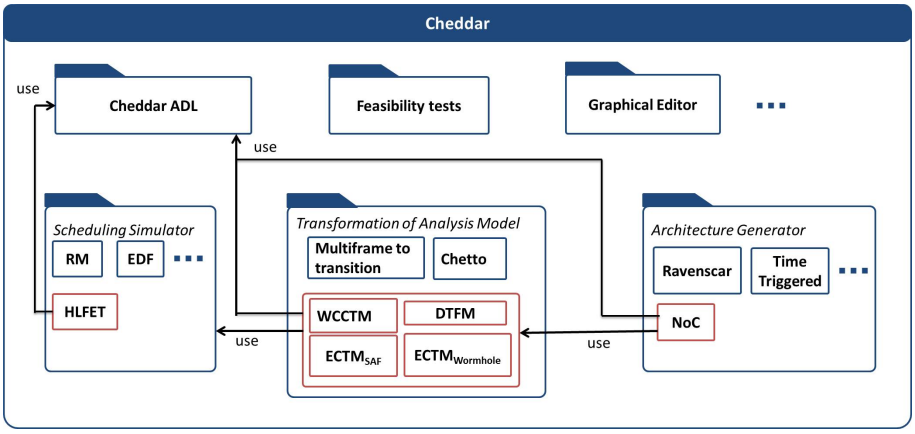

Fig. 6. Implementation into Cheddar. The NoC Architecture Generator produces a random real-time system deployed over a NoC architecture. WCCTM and ECTM boxes are for the NoC communication models described in this paper. Finally, the HLFET module extends the scheduling algorithms implemented in the simulator.

5.2 Accuracy of WCCTM and ECTM

The purpose of this evaluation is to make a comparative study between the two ECTM and WCCTM models in terms of accuracy. To do so, we evaluate the rate of schedulable task sets found as schedulable by ECTM and WCCTM models.

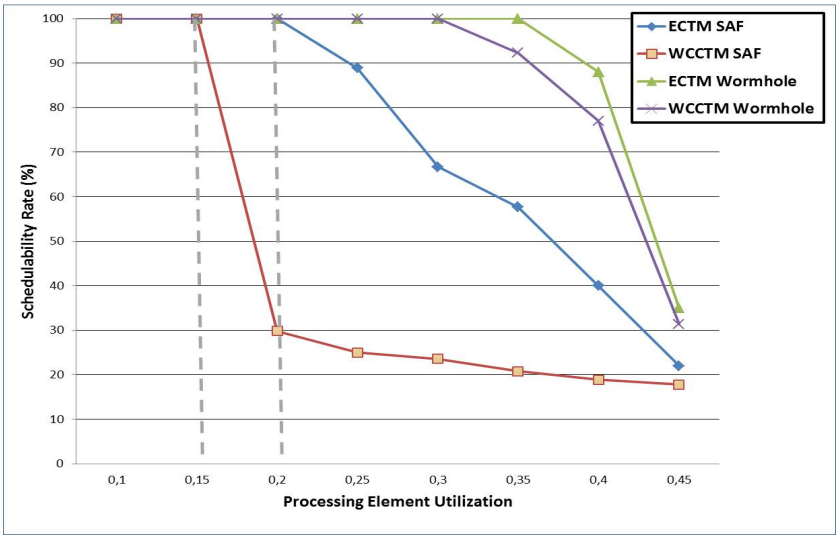

Fig. 7. Accuracy of the NoC analysis models for All-To-One traffic

In this experiment, we randomly generate 100 sets of dependent periodic tasks using UUniFast [2]. We choose a random task mapping where we can have a maximum of two tasks per processing element.

We consider two traffic patterns: All-To-One and One-To-One.

With an All-To-One traffic pattern, The source node will be randomly selected using UUniFast, while the destination node is fixed arbitrarily. All flows of the same set have the same destination node.

With an One-To-One traffic pattern, the destination node and the source node are selected randomly using UUniFast. The chosen packet size is 4 flits. Like for the previous experiment, we use a Wormhole NoC and a SAF NoC.

To perform the experiment, we first use DTFM in order to compute the flow model from the task model, the task mapping and the NoC model. Then, we apply ECTM or WCCTM and we compute the scheduling with one of the list scheduling algorithms. Since it presents the best results in term of robustness and complexity, scheduling analysis is performed with HLFET [1]. Simulation intervals are choosen according to [5].

We also consider two NoC models. The first one is a Wormhole NoC, while the second is a SAF NoC. The two NoCs have the same size and the same topology (4×4 2D mesh).

We present Fig. 7 the rate of task sets considered as schedulable according to the overall processing element utilization rate for the All-To-One traffic pattern. The figure shows that ECTM is more accurate than WCCTM with an improvement of 30% for Store-And-Forward NoCs, From a use rate equal to 0.15, some schedulable task sets are no more considered as schedulable by $WCCTM_{SAF}$

model. However, with the $ECTM_{SAF}$, these task sets remain seen as schedulable up to 0.2 of use rate.

Fig. 8 presents the same results than Fig. 7 but for the One-To-One traffic pattern. It confirms the last interpretation: WCCTM is more pessimistic than ECTM. For Wormhole NoCs, ECTM is more accurate than WCCTM with an improvement of 100%. Using $WCCTM_{Wormhole}$, the schedulable task sets are no more seen as schedulable from a use rate equal to 0.08. However, with the $ECTM_{Wormhole}$, the schedulable task set is seen as schedulable up to 0.16 of use rate.

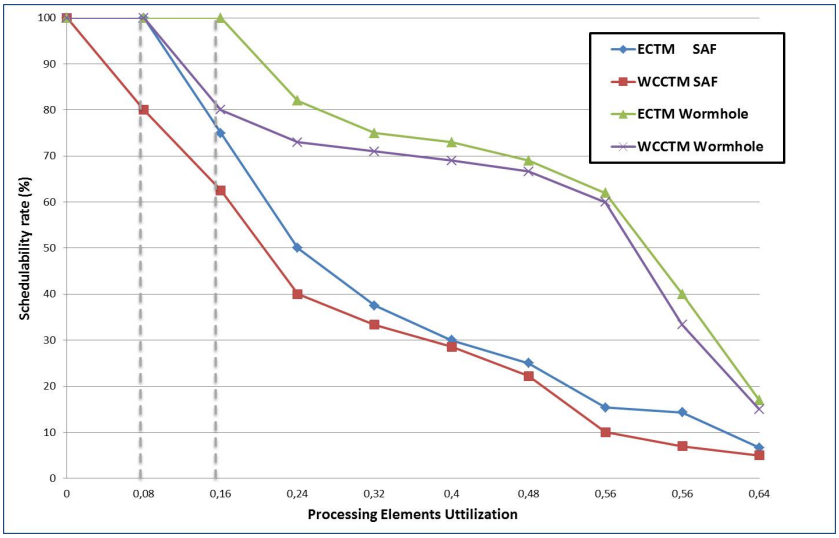

Fig. 8. Accuracy of the NoC analysis models for One-To-One uniform traffic

5.3 Evaluation of the scalability of our approach

In order to evaluate the scalability of our approach, we measured the computation time of the WCCTM and ECTM transformations. We keep the same configuration than the previous experiments. Fig 9 presents this result for different numbers of flows ranging from 15 to 120.

WCCTM provides shorter computation time than ECTM. For 105 flows in the network, WCCTM takes 2.32 seconds to compute the analysis model, while $ECTM_{SAF}$ takes 2.86 seconds, and $ECTM_{Wormhole}$ takes 6.80 seconds for 2 flits flows and 8.13 seconds for 3 flits flows. In terms of computation time, WCCTM has a shorter computation time comparing to ECTM, with a reduction of 17% for Store-And-Forward NoCs and of 54% for Wormhole NoCs.

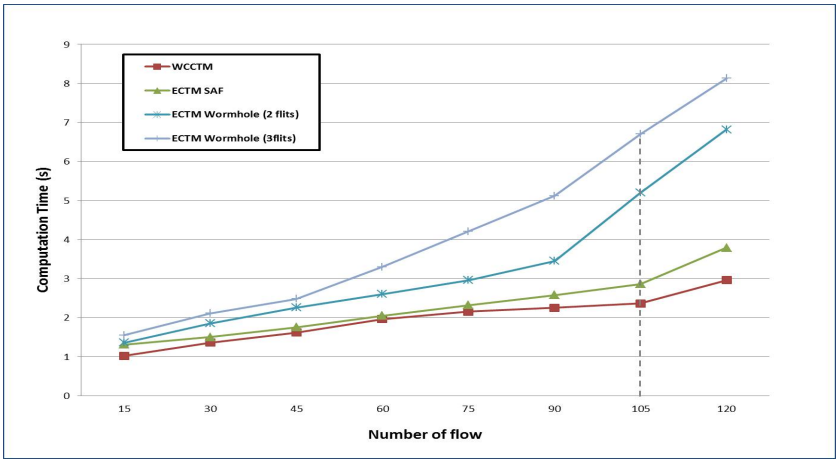

Fig. 9. Computation time for the model transformations

6 Conclusion

Delays introduced by a NoC make the schedulability analysis challenging. In this paper, we propose a new NoC communication model called Exact Communication Time Model (ECTM) in order to analyze the scheduling of periodic tasks exchanging messages over a NoC. Our approach supports Wormhole and Store-And-Forward NoC switching technics. With ECTM, we perform scheduling analysis with a list scheduling algorithm called HLFET.

We have implemented our approach in a real-time scheduling simulator called Cheddar. Results show that ECTM is more efficient than the classic solution WCCTM with an improvement of 30% for Store-And-Forward NoCs and of 100% for Wormhole NoCs, while in terms of computation time of the analysis model, WCCTM is better than ECTM with 17% for Store-And-Forward NoCs and with 54% for wormhole NoCs.

In future work, we will evaluate the overhead on the computation time introduced by the proposed model on the scheduling analysis. Furthermore, we intend to use ECTM and its associated tools to investigate the scheduling analysis of mixed criticality systems deployed over NoC architectures.

References

1. Adam, T.L., Chandy, K.M., Dickson, J.R.: A comparison of list schedules for parallel processing systems. Communications of the ACM **17**(12), 685–690 (Dec 1974)
2. Bini, E., Buttazzo, G.C.: Measuring the performance of schedulability tests. Real-Time Systems **30**(1-2), 129–154 (2005)
3. Cheng, S.C., Stankovic, J.A., Ramamritham, K.: Tutorial: Hard real-time systems. chap. Scheduling Algorithms for Hard Real-time Systems: A Brief Survey, pp. 150–173. IEEE Computer Society Press, Los Alamitos, CA, USA (1989)

4. Dridi, M., Rubini, S., Singhoff, F., Diguet, J.P.: DTFM: a flexible model for schedulability analysis of real-time applications on noc-based architectures. In: Proceeding of the 4^{th} IEEE International Workshop on Real-Time Computing and Distributed systems in Emerging Applications (REACTION). pp. 43–49 (Nov 2016)
5. Goossens, J., Grolleau, E., Cucu-Grosjean, L.: Periodicity of real-time schedules for dependent periodic tasks on identical multiprocessor platforms. Real-Time Systems **52**(6), 808–832 (Nov 2016)
6. Hagras, T., Janeček, J.: Static vs. dynamic list-scheduling performance comparison. Acta Polytechnica **43**(6) (2003)
7. Jetly, K.: Experimental Comparison of Store-and-Forward and Wormhole NoC Routers for FPGA's. Ph.D. thesis, University of Windsor (Nov 2013)
8. Kwok, Y.K., Ahmad, I.: Static scheduling algorithms for allocating directed task graphs to multiprocessors. ACM Computing Surveys **31**(4), pp. 406–471 (Dec 1999)
9. Lakshmanan, K., Kato, S., Rajkumar, R.: Scheduling parallel real-time tasks on multi-core processors. In: proceedings of the 31^{st} IEEE Real-Time Systems Symposium. pp. 259–268 (Nov 2010)
10. Lei, T., Kumar, S.: Algorithms and tools for network on chip based system design. In: Proceedings of the 16^{th} Symposium on Integrated Circuits and Systems Design. pp. 163–168 (Sep 2003)
11. Lei, T., Kumar, S.: A two-step genetic algorithm for mapping task graphs to a network on chip architecture. In: Proceedings of the 2003 Euromicro Symposium on Digital System Design. pp. 180–187 (Sep 2003)
12. Ma, S., Huang, L., Lai, M., Shi, W.: Networks-on-Chip: From implementation to programming paradigms. Morgan Kaufmann (2014)
13. Nejad, M.B., Nejad, E.B., Sayahi, A., Hashemi, S.M., Chaharlang, J.: Mapping and scheduling techniques for network-on-chip architecture. International Journal of Basic Sciences and Applied Research **2**(7), 686–693 (2013)
14. Pop, R., Kumar, S.: A survey of techniques for mapping and scheduling applications to network on chip systems. Tech. Rep. 04:4, Department of Electronics and Computer Engineering School of Engineering, Jönköping University (2004)
15. Qamhieh, M.: Scheduling of parallel real-time DAG tasks on multiprocessor systems. Ph.D. thesis, Université Paris-Est (2015)
16. Qamhieh, M., George, L., Midonnet, S.: A Stretching Algorithm for Parallel Real-time DAG Tasks on Multiprocessor Systems. In: Proceedings of the 22^{nd} International Conference on Real-Time Networks and Systems. pp. 13–22. Versaille, France (Oct 2014)
17. Shi, Z.: Real-Time Communication Services for Networks on Chip. Ph.D. thesis, University of York (Nov 2009)
18. Shi, Z., Burns, A.: Real time communication analysis for on-chip networks with wormhole switching. In: Proceedings of the Second ACM/IEEE International Symposium on Networks-on-Chip (NOCS). pp. 161–170 (Nov 2008)
19. Shi, Z., Burns, A.: Real-time communication analysis with a priority share policy in on-chip networks. In: Proceedings of the 21^{st} Euromicro Conference on Real-Time Systems (ECRTS). pp. 3–12 (July 2009)
20. Singhoff, F., Legrand, J., Nana, L., Marcé, L.: Cheddar: a flexible real time scheduling framework. In: ACM SIGAda Ada Letters. vol. 24, pp. 1–8. ACM (2004)
21. Varatkar, G., Marculescu, R.: Communication-aware task scheduling and voltage selection for total systems energy minimization. In: Proceedings of the 2013 International Conference on Computer Aided Design. pp. 510–517 (Nov 2003)