Safe and Efficient Power Management of Hard Real-Time Networks-on-Chip

Thawra Kadeed*, Sebastian Tobuschat, Adam Kostrzewa, Rolf Ernst

Technische Universitt Braunschweig, Germany

Abstract

The power overhead of Networks-on-Chip (NoCs) becomes tremendous in high density Multiprocessor Systems-on-Chip (MPSoCs). Especially in hard real-time and safety-critical systems, power management mechanisms must be developed and efficiently adhered to real-time requirements. However, state-of-the-art solution typically induces a high timing overhead, thus challenging safety, or has limited power saving capabilities. Additionally, current power-gating mechanisms do not provide an upper bound of the latency overhead, and thus no timing guarantees.

We propose a safe and enhanced approach for power-gating that allows a global and dynamic power management under timing guarantees, i.e., all deadlines of critical tasks are met. It introduces a control-layer to save power on the NoC data layer using multiple Power-Aware Traffic-Monitor (PATM) units, which apply knowledge of the global state of the system to efficiently save power on NoC routers even at high NoCs utilizations. To safely apply the PATMs in hard real-time systems while meeting the deadlines, we provide a formal worst-case timing analysis to derive PATMs upper bound latency overhead. Experimental results show that our approach efficiently reduces static power consumption, and provides scalability inducing very small area overhead.

Keywords: Networks-on-Chip, Hard real-time systems, Dynamic power management, Safe power-gating.

1. Introduction

The increase in Multiprocessor Systems-on-Chip (MPSoCs) complexity has also caused an increase in their power consumption. A higher power consumption leads to high temperature, which in turn has severe consequences for the whole system, i.e., prohibitive degradation of service, accelerating aging, and even increasing the probability of failures. Thus, power consumption needs to be thoroughly addressed especially when it comes to hard real-time and safety critical systems in automotive and avionic domains, as the previous phenomena jeopardize safety.

Networks-on-Chip (NoCs) have replaced the bus structures in today's MPSoCs. They have been utilized to provide high performance, quality of services, and fault-tolerance [7, 31]. However, such complex microarchitectures come at the cost of high NoC power consumption. Hence, a NoC by itself consumes a substantial portion of total chip power consumption. As an example, Intel 80-tile Teraflop's router consumes up to 28% of tile power [19]. The router and the links in a Mellanox server blade consume 37.5% ($15W$) of the total power budget ($40W$), with the processor consuming the same power budget ($15W$) [27, 33]. And, in the Alpha 21364 microprocessor, the interconnect consumes 20% of the total chip power [41]. Furthermore, the 16-tile MIT RAW NoC consumes 36% of total chip power, with each router dissipating 40% of individual tile power [40]. SCORPIO, a 36-Core research chip design, the network interface controllers (NIC) and router consume 19% of tile power [11]. Other research also reports that NoC consumes up to 35% of an overall chip power [8].

Moreover, a significant portion of NoC power consumption arises from static power [8]. In addition to that, static power is anticipated to be exacerbated in future transistor size. On the other hand, [4, 21, 28] state that the clock-tree power is a key contributor to total NoC power. As up to 81.3% of a router quiescent power (i.e. no load) is caused by the activity on the clock pins of high-level non-clock gated synchronous elements [21]. That in turn constitutes a big power loss in idle cases.

Thus, power management mechanisms for NoCs have to be developed to alleviate the static (leakage and clock-tree) power loss. Especially for hard real-time NoCs this is of high importance, as a high power consumption and the resulting aging can challenge the timing guarantees. A wide range of research efforts have investigated the power reduction mechanisms, and power-gating mechanisms have been successfully employed. Besides NoC routers, power-gating has been applied to cores and execution units [20]. This application of power-gating at core level emphasizes the impact of this approach on power savings.

In this paper, we introduce an enhanced power-gating mechanism in order to safely save power at NoC routers. While power gating is a promising solution to tackle the high power consumption, it typically leads to an additional

*Corresponding author.
E-mail address: kadeed@ida.ing.tu-bs.de
Postal address: Hans-Sommer-Str. 66, 38106 Braunschweig, Germany

overhead the packet may experience due to accumulated wake-up latency at each turned off router [15]. Additionally, this makes the traffic patterns more unpredictable, thus jeopardizing the timing guarantees of critical tasks in hard real-time systems. This leads to a trade-off between power saving capability and predictability (or performance) of the system.

To solve this, we propose a global power management control layer, which employs multiple Power-Aware Traffic-Monitors (PATMs). It uses the actual system state to decide at run-time if it is safe to apply power gating (e.g. ensure that no deadline misses occur) to save power. The approach has an upper bound on the latency overhead the packet may experience, thus making the approach predictable while saving power.

The PATMs units are NoC controllers that save power by monitoring the NoC traffic, and thus the router states. This mechanism efficiently applies power-gating to save power even at high network utilization. The control layer is built on top of the basic NoC (data layer), giving the user flexibility of turning it on/off based on the system requirements. Moreover, as we account for hard real-time NoCs, where all tasks are characterized with their deadlines, a formal timing analysis of our approach has been provided, deriving safe applicability of PATMs.

Hence, **the contributions** of this paper are as follows: we propose an effective and safe power-gating mechanism for NoCs by developing the global control layer approach. It utilizes local and predictable NoC controllers (PATMs) that seek for power savings with complying to timing guarantees. PATMs utilize two existing slacks, *Deadline Slack* and *Hop-Count Slack*, in order to save power even at high data rates. The approach covers different traffic patterns (e.g. periodic, sporadic). However, an optimization of PATMs in case of periodic systems is introduced, applying a grouping policy in order to better save power. Moreover, a formal timing analysis is provided in this work in order to account for NoCs hosting hard real-time systems, which require safety guarantees of critical tasks (e.g. meeting their deadlines). Furthermore, the scalability of our approach is demonstrated by employing multiple local PATMs in large NoCs.

2. State of the art

Due to shrinking of transistor size and relative reduction in supply voltage, the leakage power has been increased and constituted a significant portion of the chips total power [15]. Moreover, clock-tree power, which can be classified to static power group, has been addressed by previous work [21, 4, 28] as a major contributor to power consumption.

As motivated, NoC's power consumption constitutes a significant ratio of total chip power, which in turn requires intensive efforts in order to tackle the power problem. In addition to that, when it comes to hard real-time and safety-critical systems, efficient power management for NoCs is indispensable. Indeed, the problem of mitigating the NoC power consumption has been intensively studied by multiple research efforts [18, 30, 9, 15, 14]. Bufferless routing [14] saves router power by eliminating buffers, but it might induce issues like livelock, misrouting, and packet reassembly that must be appropriately tackled. Furthermore, Dynamic Voltage and Frequency Scaling (DVFS) [18, 42] has been proposed in order to save power on NoCs. Overall, these methods save power by adjusting the voltage and frequency according to NoC utilization rates. A major contribution w.r.t. critical systems was done by [42]. However, DVFS-based power management methods are design process limited. In other words, the physical properties (e.g. voltage) of transistors are limited by design and cannot be violated by reducing them below their limited lower bound.

On the other hand, power-gating mechanisms are widely used. Either portion of a router buffer is powered off [6], some blocks of a router [6, 30] are powered-off, or the whole router [8, 26, 9, 15] in order to reduce static power. Additionally, path-oriented fine-grained power-gating mechanism (PAPM) [13] is also presented. It selectively powers on/off paths on the network, partially shared by different sources. That is, unused queues for congested traffic can be powered off. However, the presented solutions with powering off only part of a router have limited power savings capabilities.

Panthre [30] and NoRD [8] are reconfiguration-based NoC power saving schemes. Panthre is based on the observation that only 10% of network traffic flows through 30% of the links, providing a potential for power-gating by detouring packets to exclude lightly used portions of the NoC. However, in addition to area penalty, this method suffers from issues associated with reconfiguration such as detour. Moreover, it increases some routers' load in order to keep others sleep, thus accelerating aging of these and leading to an age unbalance between routers [16]. NoRD, on the other hand, uses bypass paths to circumvent powered-off routers. It relies on packet detours which has the same aforementioned reconfiguration drawbacks. Furthermore, Catnap [10], uses multiple networks to increase the efficiency of power-gating, but it is mainly proposed for Chip-MultiProcessors (CMPs) with high-bandwidth.

Overall, the big challenges of all power-gating schemes are two-fold: when and how one may power the routers on/off in order to save power under negligible latency overhead. Moreover, a power-gating itself comes with power overhead, which should be addressed thoughtfully. The latter mainly comes from the power lost in turning routers on/off, i.e., turn on/off the relative power switch, connecting router to power supply. Hence, the power loss should be compensated so that the application of power-gating does not adversely increase power. A variety of research work has been made to fulfill that goal and overcome the power-gating challenges.

Conventional power-gating schemes turn off routers as soon as they go to idle mode. Matsutani [26] proposes an optimization of the conventional schemes by introducing an

early wake-up technique. It is mainly based on look-ahead routing that sends a wake-up signal two hops ahead of the packet arrival. However, the method only sends the control signals at most 2 hops ahead, and in turn hides only a small fraction of the wake-up latency. An optimization to the look-ahead routing is introduced by the work of Chen [9]. He proposes power punch technique which sends multi-hop wake-up signal in order to almost hide the overall latency overhead. However, the method still suffers from the fragmented power cycling that decreases the potential power savings. On the other hand, the method generates wake-up signals by the Network Interface (NI) to bunch all routers along the packet's path, which might be already on. That in turn requires generating and monitoring many useless power punch signals.

TOOT (Turn-on on Turn) is a mechanism that reduces the total number of wake-ups by providing a bypass path for straight/eject packets [15]. Consequently, TooT improves the efficiency of power-gating mechanism in power perspective under performance penalty as packets have to wait for each other in Toot's bypass latch.

Our aim is to overcome the related work limitations and shortcomings. *First*, none of the aforementioned power-gating schemes provides guarantees for hard real-time constrains as they focus on Best-Effort (BE) systems. *Second*, we do not limit the work to certain usecase observations [15, 30], and to estimate the overhead of the power-gating approach independently from the NoC topology, as opposed to [9, 15]. *Third*, as it can be summarized from all previous work, they either save power efficiently at the cost of performance [15], or provide very light latency overhead but less power savings [9].

Hence, we introduce in this paper a global power management to save power through a control layer. It is independent from the actual implementation of the NoC data layer, and mainly based on the PATMs units. The latter have a global knowledge of the system, and apply this knowledge in order to take safe decisions concerning the power state of NoC routers. That way, PATM effectively saves power inducing a small area penalty. Furthermore, a formal timing analysis is employed in order to derive the safe applicability of our approach to hard real-time applications. Such protocol-based synchronisation at runtime was already successfully applied for NoCs to increase predictability or performance [22]. However, it cannot handle power issues, and hence we extend this concept to include safe power management.

The rest of this paper is organized as follows: the PATM approach through the control layer is introduced in Section 3. Section 4 presents PATM optimization for purely periodic traffic to better overcome power-gating power overhead. Formal timing analysis is provided in Section 5 from which PATM derives its safe applicability. The evaluation methodology, necessary to evaluate the efficiency of the proposed approach, is introduced in Section 6. In Section 7, the experimental results are demonstrated employing both realistic application and synthetic workloads. Section 8 demonstrates the scalability of our approach in large NoCs. Finally, Section 9 concludes the paper.

3. Dynamic power management using PATMs

In this work, we propose a predicable and safe power-gating mechanism for NoC routers, which covers the aforementioned shortcomings of the state of the art. We use a power-aware traffic-monitor as a global controller, the PATM. It implements a protocol-based synchronisation to know the global state of the NoC, and applies this knowledge to save power while keeping the system predictable. To do this, each sender first synchronises its NoC access with the PATM, and waits for the respective acknowledgement before using the NoC. That in turn implies an additional delay for each task access to the NoC. However, as we conform our approach to hard real-time systems where timing guarantees should be provided, the additional overhead has to be bounded. In other words, the analysis of our approach should provide the timing overhead at design time, from which the safe applicability of PATM can be derived as introduced in Section 5.

Moreover, we use an *Acquisition, Execution, and Restitution* (*AER*) task-model, which decouples task's execution phase from communication one [12]. It defines a predictable access pattern of a safety critical task by decomposing it into three consecutive phases: acquisition (read), execution, and restitution (write). The *AER* model is a typical model in embedded systems, especially for real-time and safety-critical domains. The task-model not only increases the potential predictability [12], but also the power savings using PATMs as a task needs to synchronise its NoC access once for the entire transmission (burst of packets). That allows to limit the number of synchronisation messages as well as save power – exploiting the task's execution time to turn off routers.

Furthermore, based on the global state of the NoC, e.g., the concurrent active tasks, and upon the stored database that contains the tasks along with their NoC paths (source-destination routes), PATM controls the Power-Gating (PG) signals of the routers. Each sender is provided with a local supervisor called *client* to fulfil the synchronisation with PATM. Table 1 presents an overview on the different control messages and main timing parameters.

When, in large NoCs, in case of disjoint real-time applications, we can split a NoC into multiple independent regions. Multiple PATMs can be implemented as each of them represents a local controller of one NoC region. However, in case of applications where communications comprise several such regions, PATMs can communicate to each other using an interconnect in order to exchange each other regions' states, as demonstrated and detailed in Section 8.

The general architecture of the PATM approach is shown in Figure 1. We introduce a control-layer, which can be applied on top of the existing NoC data layer. This decouples the mechanisms responsible for power savings

Table 1: Messages and Timing Parameters

Message Type	Message	Description
Control Messages	Req_Msg	NoC access request message from an active sender to PATM, including the destination address
	Ack_Msg	NoC access acknowledgment message from PATM to the sender
	Rel_Msg	Release message from the destination to PATM
Power-Gating Messages	PG_ON	Power-Gating-ON: signal sent by PATM to turn-off a router by turning off the respective power switch
	PG_OFF	Power-Gating-OFF: signal sent by PATM to turn-on a router by turning on the respective power switch
Timing Parameters [20]	T_{WU}	Wake-Up (WU) latency: corresponds to the number of cycles required to charge up the local voltage of a powered-off router
	BET	Break-Even Time (BET): corresponds to the number of sleep cycles required to compensate the energy overhead induced by turning a router on/off

(PATM) from the underlying NoC infrastructure conducting switch arbitration between packets (the data layer). That way, no complex modifications at the (existing) data layer and routers are needed, and the routers can be completely turned-off as they are not required to generate and forward wake-up signals. The control-layer is composed of PATM (or multiple ones in case of large NoCs), local clients, and an interconnect. PATM, as a global module, is responsible for power savings with complying to real-time guarantees. The clients components are responsible for synchronizing the senders NoC accesses with PATM by employing additional direct links, 6-bit wide each. The latter adopts control messages (5 bits, 1 bit to convey the control message, 4 bits to determine the destination/source address in 4×4 NoC in case of Req/Rel, respectively), and power-gating ones (1 bit) between PATM and the clients.

Recall, the use of an additional control layer also allows applying our approach easily to different NoC architectures without the need to modify the routers.

The power-gating itself comes with additional timing parameters, the router wake-up latency (T_{WU}) and the Break-Even Time (BET). T_{WU} corresponds to the additional latency the packet experiences along its path, and

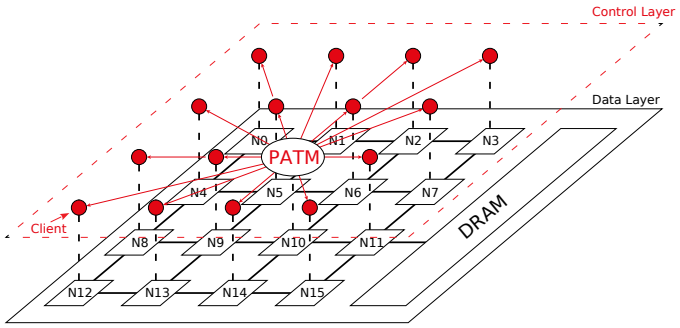


Figure 1: PATM within the control-layer (red color) for power savings by monitoring traffic in data-layer

BET rule corresponds to the additional cycles the router has to stay turned off in order to overcome the power-gating energy overhead. Both parameters are considered as major challenges in conventional power-gating methods. The WU latency can lead to a blocking of packets in the network at each router (i.e. the packet waits at each router until it is turned on), and thus to blocking propagation which can reduce the performance of the network. One possibility to avoid this, is to turn on routers early, which then might decrease the power savings. Similar, if a router is turned on again before the BET is over, the power overhead of turning a router on/off outweighs the power savings and thus might even increase the overall power consumption. Hence, we apply our global controller knowledge as an efficient factor to safely turn the routers on/off with accounting to these timing challenges. The controller gives a sender direct access when all routers along its path are *on*, or it sends PG_OFF signals to wake up all powered off routers at a time. That way, it mitigates the cumulative waiting time the packet may experience along its path.

Furthermore, as PATM knows the global state of the NoC, it efficiently switches a router on/off. It filters out short router idle periods in order to overcome wake-up power overhead. This can be easily fulfilled by considering the calculated BET [20, 29]. The conventional way only checks whether the router buffer and pipeline are empty, then sends a power-gating signal to turn it off. In our approach, the PATM uses a third factor. It checks whether there exists an active task that is going to use this router, and thus turning off the router might violate the BET constraint. If so, the controller keeps the router on.

In addition to that, the global controller applies efficiently a router wake-up policy because it knows which router is powered-off, and sends a wake-up signal only to this router in contrast to sending it by the NI to the whole path [9]. The latter is not aware of the routers' state along the packet path because its knowledge is limited to its sender activities. So when the NI has to send data, it sends a wake-up signal to the whole corresponding path to punch powered-off routers without first accounting for BET, and second for routers' state as they might be all *on* already. That, on the other hand and in addition to power penalty,

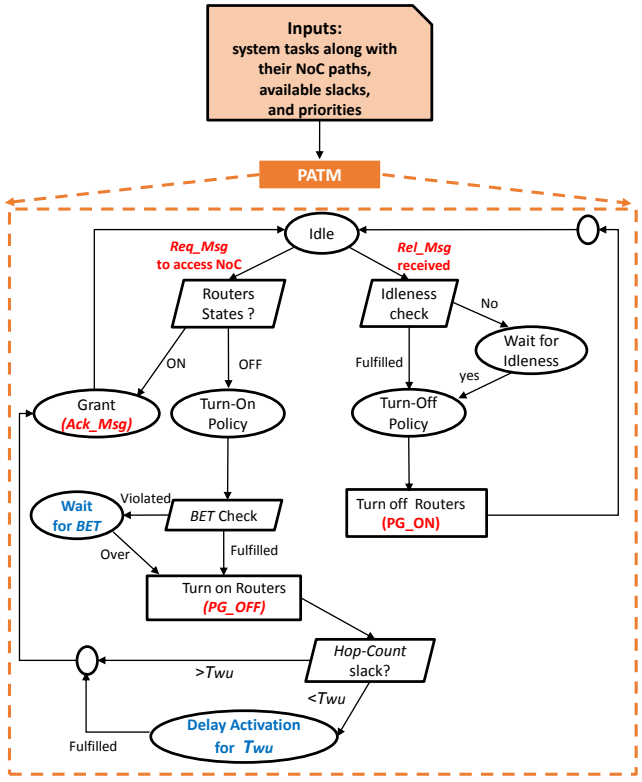


Figure 2: PATM Workflow

requires monitoring multiple useless punch signals.

In the sequence, we detail the mechanisms of PATM used to save power with safe latency overhead. Figure 2 summarizes the overall workflow of PATM where the messages in Table 1 are highlighted in red, and the timing parameters are highlighted in blue.

3.1. Low-Power Safe-Latency turn on policy

In this paper, two mechanisms are employed to overcome violating *BET* rule, with less and safe worst-case packet latency overhead: the *deadline slack* and the *Hop-Count slack*.

3.1.1. Deadline slack

When PATM receives a Req_Msg from an active task and figures out that at least one of the routers in the corresponding path is off, it checks for how long the router has been turned off. If it is not adequate (less than *BET*), PATM safely employs a task NoC access delay policy in order to consider *BET* rule. In real-time systems, each critical task is characterized by its relative deadline under which a task should be fulfilled. However, finishing a critical task too early has the same advantage of completing it by its deadline [37]. Thus, PATM exploits the *task's deadline slack* in order to delay the task NoC access, assuring *BET* and thus saving higher power. The *task's deadline slack* ($DSlack_i$) in turn defines the time budget between the task's deadline and its Worst Case Response Time (WCRT) as determined in the following formulas:

$$DSlack_i = D_i - R_i \tag{1}$$

$$DSlack_i > MLO_{PATM}, \tag{2}$$

where R_i: denotes the WCRT of a task i necessary to be transmitted using basic NoC (no power-gating is employed);
D_i: denotes the deadline of the task i;
MLO_{PATM}: denotes the Maximum additional Latency Overhead induced by PATM approach, as detailed in Section 5 (cf. Equation 20).

The predominant condition to apply the delay policy is to have the task's positive slack greater than PATM overhead. Note that in case $DSlack_i$ is negative, which implies that the deadline is already violated, the applicability of any power-saving mechanisms is obviously forbidden.

Furthermore, when PATM is supposed to turn on a powered-off router that has been turned off for less than *BET*, it checks the available task's slack and delays the task to a number of cycles ($DelCyc_i$) based on the following equations:

$$Roff_r = BET - OffCyc_r \tag{3}$$

$$DelCyc_i = \begin{cases} Roff_r, & \text{if } DSlack_i > MLO_{PATM} + Roff_r \\ 0, & \text{otherwise,} \end{cases} \tag{4}$$

where $Roff_r$ denotes the number of off cycles required to keep the router powered off to account for *BET*; $OffCyc_r$ denotes the number of *off* cycles the router has been turned off. That is, PATM delays a task only if the available slack is greater than the required off cycles ($Roff_r$), and the latency overhead induced by PATM.

After checking *BET*, PATM inactivates power-gating by sending (in parallel) power-gating-off signals (PG_OFF), in order to bring the power supply back to the corresponding routers. At the same time, it investigates sending a grant to the active task using the following *Hop-Count* slack mechanism.

3.1.2. Hop-Count slack

It defines how far the powered-off routers are from the source tile in terms of hops. PATM exploits the *Hop-Count slack* of the powered-off routers in order to mitigate the WU latency the packet may experience along its path. The number of hops that completely hide the WU latency of a certain powered-off router is given by: $(\lceil \frac{T_{WU}}{T_R+T_L} \rceil)$, considering that the packet takes $(T_R + T_L)$ cycles per router and link (hop). For instance, if the powered-off router is the third hop along the packet's path, then a hop-count slack of 2 is typically able to compensate 10 cycles wake-up latency for routers with a 4-stage pipeline. Thus, PATM investigates the *Hop-Count* slack of the powered-off routers along the corresponding path, and if all of them have adequate Hop-Count slack (none of them is one of first two hops in 4-stage router, or first 3 hops in 3-stage

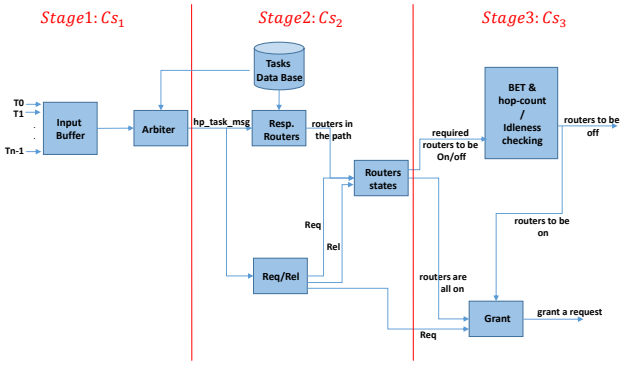


Figure 3: Block Diagram of the PATM Implementation

router), the PATM acknowledges the sender (Ack_Msg), while simultaneously turning on the powered-off router(s). However, when the *Hop-Count* slack is exceeded by at least one router, PATM has to delay the sender accounting for T_{WU}.

As we design our power savings approach to meet hard real-time objectives, the increased latency of a critical task must not jeopardize timing guarantees. Therefore, we always assure a safe application of the aforementioned task delay (induced by BET) by checking, at run time, the available slack against the required delay. Regarding PG_OFF signal required to turn on a router, if PATM wants to delay a *client* by 5 cycles and PG_OFF signal requires 1 cycle, PATM sends it after 4 cycles as it arrives after 5 cycles at the router.

3.2. Turn Off policy

As PATM knows the global state of the NoC, it turns off the NoC routers based on the active tasks and consequently based on the traffic in the NoC. PATM has to ensure the idleness of routers along the packet's path. Therefore, as the approach conforms with timing guarantees, when at least one task is active, PATM waits for a release message (Rel_Msg) from the destination tile (cf. Table 1), which ensures none of the routers is still being used by the corresponding source (the last packet has been arrived). After that, it checks whether there exist other active tasks that are using the same routers. If so, it repeats the same strategy by waiting for their arrival messages, otherwise, it turns off the idle routers.

Figure 3 presents a high level view of the PATM implementation and its processing pipeline for handling requests. Overall, PATM is pipelined within three stages. Arriving requests are stored in an input buffer. An arbiter, in the first stage, selects the highest priority request. In the next two stages, PATM decides the best power state of routers. That is in turn derived from the workflow of PATM (cf. Figure 2). Thus, the second stage corresponds to Req/Rel checking, deriving the respective routers in the task path, and revealing the required routers to be on/off; the third stage corresponds to BET and hop-count/idleness checking in case of Req/Rel, respectively; and then the grant process to acknowledge a request.

In the sequence, we optimize PATM for periodic pattern, exploiting the periodical feature, in order to better save power using the activations grouping policy.

4. PATM optimization for periodic activations

The general approach, demonstrated in Section 3, covers different types of traffic patterns, e.g., *periodic* and *sporadic*. Moreover, it involves different cases of deadline occurrence, e.g., a task has its deadline equal to its period $D_i = T_i$ (implicit-deadline), and the deadline is smaller than the period $D_i < T_i$ (constrained-deadline).

In spite of the fact that PATM can follow the same general policy to acknowledge a request, we exploit the periodicity in order to achieve more efficient power savings. When a purely periodic system is running, and thus PATM knows precisely the next arrival of a task activation, it follows a grouping policy to turn on/off routers and consequently acknowledges the request.

4.1. Grouping tasks activations

As the router switching from $off \longrightarrow on$ is considered one of the most important factors overall power-gating mechanisms because of its energy overhead (e.g. capacitance charge), we investigate in this section how to decrease the number of the router switching activities employing the periodical feature. Figure 4 depicts a periodic task activations scenario, and the PATM reaction in normal case and under Optimized PATM (Opt-PATM). In normal case, the router has to be turned on at every new activation arrival, which in turn leads to a significant power-gating energy overhead.

To this end, we investigate to group efficiently task activations in order to decrease the aforementioned overhead. Therefore, at design time, we seek to group the activations in the hyper-period interval after which the periodic pattern of all tasks is repeated [32]. It applies the resulting grouping model for all other activations after the hyper-period. The maximum number of shifting cycles of a task activation ($Shift_i$) can be bounded by Equation 5:

$$Shift_i = D_i - R_{\{i,PATM\}}, \tag{5}$$

where $R_{\{i,PATM\}}$ denotes the *worst-case response time* (WCRT) of a task i including the latency overhead of PATM in case routers are turned off.

The resulting grouping, as it is depicted in Figure 4, corresponds to shift one activation while fulfilling directly the right next one, decreasing the number of the switching activities to the half. Next, PATM should be augmented with the tasks periods and their shifting parameters, resulting from the optimization at design time. At run time, PATM reschedules the tasks arrivals based on the state of the former ones. To this end, It checks the state of the former activation, in case it has been shifted, it fulfills

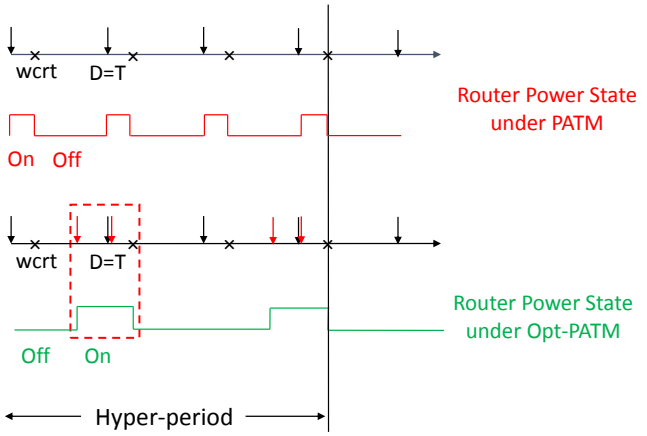


Figure 4: A periodic task activations scenario and the corresponding router power state under PATM (red color) and under optimization (green color)

the current activation once it occurs. In contrast, it shifts the current activation in case the former one has not been shifted.

Moreover, as the grouping policy designed for the worst cast response time of a task activation, PATM has to intelligently decide whether the router has to stay on until the next activation occurs (cf. Figure 4, the red dashed box), or turn it off. Thus, at run time, PATM always compares the time difference between the next activation arrival of any task, and the idleness start of the router, i.e., if all active tasks that utilize the router have arrived to their destinations. The comparison can be expressed by the following formula:

$$\forall i \in \{ROT\}, \quad NxtAct_i - IdleStart_r > BET, \qquad (6)$$

where $NxtAct_i$: denotes the next activation of any task i, which belongs to the set Router Overlap Tasks (ROT);
$IdleStart_r$: denotes the idleness start of the router r.

PATM employs the Equation 6, and turns the router off only if the BET rule is not violated (Equation 7 is fulfilled). In the latter case, the router's state in the red dashed box may adopt an additional transition (turned-off until the next arrival occurs to be on again) in order not to lose power.

On the other hand, the Opt-PATM can decrease the power-gating power overhead at high data rates with short message sizes as it is demonstrated in the experiments (cf. Figure 10a). However, the packet latency will significantly increase from the fact that PATM shifts a task activation until the next one. Therefore, a formal timing analysis of the system tasks is required to prove the applicability of PATM. And, in case the analysis provides timing guarantees, the application of Opt-PATM is then up to the user and his implementation objectives. However, for purely periodic systems the deadline typically corresponds to the period. Hence, an activation can be delayed until shortly before the next activation.

5. Timing analysis

The application of the proposed approach to hard real-time systems requires a formal timing analysis capable of providing an upper bound for the latency overhead. First, we employ one of the standard formal analysis frameworks at design time to calculate the worst-case response time R_i of a task i in the basic NoC, which implies the congestion case. Next, we derive the deadline slack ($DSlack_i$) and compare it against PATM latency overhead (detailed in Definition 6) to derive the safe applicability of PATM.

We consider in our analysis Static-Priority Preemptive (SPP) arbitration policy inside routers to resolve the congestion between tasks. In spite of the fact that our approach is compatible with other analysis frameworks, e.g., real-time calculus [43], composable analysis using network calculus [25], or compositional performance analysis [39], we employ in this work the holistic analysis from [36]. It basically considers the higher priority tasks' periods to calculate the R_i of lower priority tasks. This can be easily included in periodic tasks because the task's period is provided. However, to account for sporadic tasks in the worst-case, we consider a minimum inter-arrival interval T_i between two consecutive activations of a task [34].

Let us first define the transmission time of an activated instance of a task i, when no congestion exists.

Definition 1. *The basic network latency C_i of a task* i *denotes the transmission time of an activated instance of a task* i *via the basic NoC in case of isolation (no congestion exists).*

C_i can be computed simply as specified in (7):

$$C_i = \overbrace{(h_i - 1) \cdot d_R}^{\text{header routing}} + \overbrace{h_i \cdot d_L}^{\text{header traversal}} + \overbrace{(\sigma_i - 1) \cdot d_L}^{\text{payload traversal}}, \qquad (7)$$

where h_i: is the number of hops;
d_R: is the routing delay of the header flit;
d_L: is the link traversal delay;
Overall, the transmission time is equal to the time it takes a header flit to arrive to its destination, augmented by the transmission of the remaining payload $(\sigma_i - 1)$ across the last link (due to the pipelined traversal).

Definition 2. *The worst-case response time R_i of an activated instance of a task* i *in case of congestion can be computed by iteratively solving the following equation:*

$$\mathbf{R_i^{n+1}} = C_i + \sum_{j \in hp(i)} \left\lceil \frac{\mathbf{R_i^n} + J_j^R + J_j^I}{T_j} \right\rceil \cdot (C_j + B_j), \qquad (8)$$

where $\left\lceil \frac{R_i + J_j^R + J_j^I}{T_j} \right\rceil$ denotes the maximum number of activations can be released by tasks that have higher priority than i ($hp(i)$) during the time interval $[0, R_i]$ and B_j accounts for the interference due to backpressure. The Equation 8 also accounts for release jitter J_j^R and indirect interference jitter J_j^I.

We find the variable R_i appears on both sides of the Equation 8. A solution can be computed iteratively [2], because all components grow monotonically with respect to the response time. The iteration starts with $R_i^0 = C_i$ and terminates when two successive iterations provide identical results $R_i^{n+1} = R_i^n$. The iteration also terminates if $R_i^{n+1} > D_i$, which means a deadline miss for this task. As the higher priority tasks may cause interference to the lower ones in case they share with each other the same/part of the network path, we sort the tasks based on their priorities and proceed to analyse them from the highest priority one by one. Finally, the schedulability test consists of checking whether the condition ($R_i < D_i$) holds for every task in the system.

Definition 3. *The interference jitter J_j^I accounts for the jitter through indirect interference between senders via an intermediate priority transmissions. It is defined as:*

$$J_j^I = \begin{cases} R_j - C_j, & \text{if } \exists k \mid k \in hp(j) \wedge k \notin hp(i) \\ 0, & \text{otherwise.} \end{cases} \quad (9)$$

That is, k can delay the first packet of j and hence cause two consecutive packets of j to appear and preempt i in a time interval smaller than T_j.

Definition 4. *The backpressure blocking B_j accounts for the blocking resulting from backpressure effects, i.e., the time during which transmissions are blocked due to lack of buffer availability in the neighbouring router.*

If a transmission from task j inflicts only downstream indirect interference on the transmissions from the task i (under analysis), the backpressure has two upper-bounds. One is the total amount of interference that j can suffer downstream. The other factor is the traversal time across a single link of a packet of j that can be simultaneously buffered within the contention domain (CD) of i and j. It can be computed as follows:

$$B_j = \sum_{\substack{k \in hp(j) \\ dstream(i)}} \left\lceil \frac{R_j + J_k^R + J_k^I}{T_k} \right\rceil \cdot min\{C_k + B_k, \beta \cdot |CD_{i,j}| \cdot d_L\}, \quad (10)$$

where β is the buffer size of each VC, and $CD_{i,j}$ represents the set of links shared by i and j , i.e., their contention domain; k belongs to the set of higher priority tasks than j ($hp(j)$) in downstream of i (dstream) outside the $CD_{i,j}$.

If a transmission from task j inflicts both upstream and downstream indirect interference on i, the backpressure has a single upper-bound equal to the amount of interference that j can suffer downstream. It can be bounded by solving the following equation:

$$B_j = \sum_{\substack{k \in hp(j) \\ downstream(i)}} \left\lceil \frac{R_j + J_k^R + J_k^I}{T_k} \right\rceil \cdot (C_k + B_k). \quad (11)$$

For more details, the reader can refer to the analysis of Indrusiak et al. [36].

Definition 5. *The worst-case response time $R_{\{i,PATM\}}$ of an activated instance of a task i including our approach can be bounded by the following formula:*

$$R_{\{i,PATM\}} = R_i + MLO_{PATM}^{Del}. \quad (12)$$

As described before, R_i denotes the worst-case response time of a task i using the basic NoC (cf. Equation 8); MLO_{PATM}^{Del} denotes the Maximum Latency Overhead induced by PATM and the delay policy.

Definition 6. *The maximum latency overhead MLO_{PATM}^{Del} from the Equation 12, induced by the control-layer based on PATM, a task i may experience once it is issued can be bounded by:*

$$MLO_{PATM}^{Del} = R_i^{Ctrl} + R_i^{Proc} + R_i^{Del}, \quad (13)$$

where R_i^{Ctrl} denotes the worst-case latency overhead of the control messages sent to and from PATM; R_i^{Proc} denotes the worst-case processing time the task instance may experience by PATM; and R_i^{Del} denotes the worst-case delay the task instance may experience by PATM in case of turned off routers.

The control messages that cause latency overhead, as derived directly from the description of the approach, are Req_Msg (including the destination address) and Ack_Msg as follows:

$$R_i^{Ctrl} = R_i^{Req} + R_i^{Ack}. \quad (14)$$

As mentioned before, the control messages are transmitted using an independent network layer (control-layer), similarly to many already commercially available NoCs, e.g., MPPA-256 and Tile64. Moreover, as it is depicted in Figure 1, each processing node has a direct connection to PATM. Thus that the worst-case transmission time of any the control messages is the only time required to send such a message via the link (i.e. the maximum link traversal time).

Definition 7. *The worst-case processing time R_i^{Proc} a task i may experience by PATM can be bounded by:*

$$R_i^{Proc} = C_k^{Busy} + R_i^{Blk} + C_i^{Comp}, \quad (15)$$

where C_k^{Busy} denotes the worst-case busy time PATM requires to process a task k, which is being processed while a task i arrives; R_i^{Blk} denotes the worst-case blocking time (Blk) the task i may experience by the higher priority tasks; and C_i^{Comp} denotes the worst-case computation ($Comp$) time a task i may experience by PATM to be served.

PATM employs Static Priority Non Preemptive scheduling policy (SPNP) in order to arbitrate between tasks. That means a task i might wait for the time the PATM requires to complete processing of task k, which is being processed while a task i arrives [23]. Furthermore, in our work, we consider the critical instant scenario when all senders are activated simultaneously, and they are issuing the consecutive requests with the maximal rate. That means while PATM is processing a request, other requests are waiting in

the input buffer of PATM. Thus, the worst-case processing time of a task i is highly dependent on the higher priority tasks. In order to calculate R_i^{Proc}, let N denote the set of senders assuming that the sender does not send another request unless the first one has been served; and let $C_{s,i}$ denote the computation time it takes PATM to process a task i within one stage (cf. Figure 3). Then the individual factors of the Equation 15 can be calculated as follows:

$$C_k^{Busy} = (C_{s_1,k} + C_{s_2,k} + C_{s_3,k}), \forall k \in N \quad (16)$$

$$R_i^{Blk} = \sum_{j \in hp(i)} \left\lceil \frac{R_i + J_j^R}{T_j} \right\rceil \cdot (C_{s_1,j} + C_{s_2,j} + C_{s_3,j}) \quad (17)$$

$$C_i^{Comp} = (C_{s_1,i} + C_{s_2,i} + C_{s_3,i}), \quad (18)$$

where $(C_{s_1,k} + C_{s_2,k} + C_{s_3,k})$ denotes the computation time it takes PATM within its three stages $(C_{s_1}..C_{s_3})$ to complete processing a current task k; the blocking time (R_i^{Blk}) of a task i corresponds to tasks that have higher priority than i $(hp(i))$ – considering their multiple occurrences $\left\lceil \frac{R_i + J_j^R}{T_j} \right\rceil$ during the time interval $[0, R_i]$.

Definition 8. *The worst-case delay R_i^{Del} a task i may experience by PATM induced by power-gating approach in case the routers are turned off can be bounded by:*

$$R_i^{Del} = R_{PG\_OFF} + T_{WU} + BET, \quad (19)$$

where R_{PG_Off} denotes the worst-case time required to propagate a power-gating-off signal;
T_{WU}: denotes the router wake-up latency;
BET: as described before, it denotes the Break-Even Time necessary to overcome the power-gating energy overhead.

The interpretation can be derived using the following worst-case scenario. When PATM tackles a request from the task i, we assume that one or many routers along the packet's path are *off*. Then, the latency overhead of turning one router on is the same of turning-on multiple routers simultaneously because of parallelism provided by the control-layer (direct connections between PATM and routers). Moreover, we consider in the analysis that the powered-off routers violate the *Hop-Count* slack, and hence the task should account for the wake-up latency T_{WU}. BET factor also counts as we assume in the worst-case that the router is just powered-off and right next cycle another request arrives.

Furthermore, we have included the delay policy overhead in the PATM latency analysis (the Equation 13) in order to demonstrate the worst-case latency overhead the packet may experience using our approach. However, to make PATM much more efficient as it checks the possibility of delay at run time (as introduced in Section 3), we excluded the delay overhead (induced by BET) from the checking of the safe applicability of PATM at design time. That is, the considered worst-case latency overhead induced by PATM is limited to the following equation:

$$MLO_{PATM} = R_i^{Ctrl} + R_i^{Proc} + T_{WU}. \quad (20)$$

Finally, to assure timing guarantees at design time, the constraint $DSlack_i > MLO_{PATM}$ must be satisfied for each critical sender. Note that known relative deadline for each critical task is an essential requirement in real-time systems [37, 23]. Otherwise, meeting deadlines cannot be formally guaranteed. To this end, the designer must asses the feasibility of the introduced approach through: (i) a calculation of the slack for a system in normal case (without PATM), and (ii) a comparison of PATM overhead against the available slack. If the slack is smaller than PATM overhead, the application of PATM must be excluded from routers along the packet's path of the corresponding violated task. In other words, the corresponding routers are always *on*, and the violated tasks do not synchronize their NoC accesses with PATM – utilizing a direct access.

6. Evaluation methodology

For evaluation we implemented the PATM in an OMNeT++ event-based NoC simulation framework with the HNOCS library [5]. The slack budgets of transmissions are computed using the analysis from York [36], implemented in python. Recall, we employ in the analysis SPP arbitration in routers. Moreover, the number of Virtual Channels (VCs) is equal to the maximum number of congestions for any port, which is in our experiments 2 VCs.

Moreover, we use a standard ASIC design flow in order to evaluate PATM area and power overhead. The results are obtained using the VHDL implementation of the IDAMC platform [38] for ASICs. NoC routers are synthesized, placed and routed using ASIC tool flow. Different process technologies from UMC ($65nm$) and TSMC ($28nm$) with core cell libraries of both high and low threshold voltages in worst-case corners (WC, 0.9V, 125°C) and (WC, 0.72V, 125°C) were selected, respectively. Synthesis for ASIC implementation was carried out using Synopsys tool chain. A top-down approach was used for this compilation, while preserving the design hierarchy. After synthesis, place and route design the parasitic extraction was performed. The results are back annotated into the designs to allow accurate power measurements.

Note that due to the significant advantages of Clock Gating (CG) where the effect of clock-tree power is drastically reduced [21, 28], automatic clock-gating was enabled during synthesis. That means the tool automatically inserted low-level clock-gating cells once suitable enabling conditions were detected. However, as introduced in [21], the activity on the clock pins of high-level non-clock gated synchronous elements still leads to a clock-tree power overhead. Thus, the latter is tackled in this paper using PG.

Recall, we use the predictable task accesses model (AER), and therefore we employ DMA engines in order to adopt the long transmissions. Moreover, we employ in the experiments 4×4 2D mesh NoC. A router is composed of 5 ports, with 4 ports connecting to neighboring routers and the fifth one connecting to a tile (e.g. processor, memory).

Table 2: Key Simulation Parameters

Network topology	2D Mesh
Network size	4×4 and 8×8 NoCs
Routing algorithm	XY source routing
Switching technique	wormhole switching
Arbitration	SPP
Link bandwidth	35bits/cycle, consistently with [38]
Flit size	140 bits
Router pipeline	4-stage
Input buffer depth	2 VCs, 5 flits/VC
Technology	65nm, 28nm
Clock Frequency	500MHz

The router power dissipation $P(V, f)$, considered in this work, can be computed as follows:

$$P(V, f) = P_{buffer} + P_{switch} + P_{link}, \qquad (21)$$

where P_{buffer}, P_{switch}, and P_{link} represent the power dissipated at input buffers, switch, and link, respectively.

In addition to that, in our evaluation, the *BET* is 10 cycles and the wake-up latency is 2 cycles, consistently with prior research [8, 26, 10]. As a considerable amount of energy is spent to turn a router *off* and bring it *on* again, the effective sleeping time is considered after accounting for *BET* that compensates the aforementioned energy loss. Note that more details about the simulation setup are summarized in Table 2.

7. Experimental results

In this Section, using the aforementioned evaluation methodology, we evaluate the proposed approach along with its implementation overhead in terms of area, power, and performance. In the sequel, we demonstrate the PATM efficient results employing periodic pattern from two different workloads: *realistic usecase* and *synthetic workloads*. Note that all statistics are collected after sufficiently NoC warm up.

7.1. Usecase workloads

We first employ a realistic usecase extracted from the work of Shi et al. [35], in order to demonstrate the PATM impact on power and performance using a real application. The chosen application is the control of a vehicle with assistance functions, as it considers both large flows as well as control loops with short messages. The vehicle is designed to recognize an unknown space by populating a database of obstacles, obtained by stereo photogrammetry and ultrasonic sensors. Next, several flow periods and data sizes have been changed in this work in order to increase the throughput of the application' tasks. Task periods vary between 0.4 ms to 1 ms, and communication volumes vary between 9 kB and 0.3 MB. The workflow graph and the

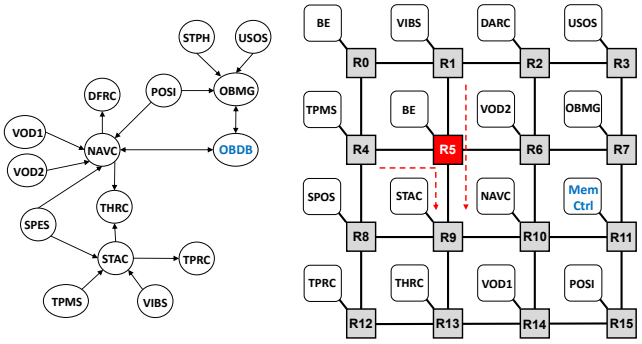


Figure 5: Graph (left) and mapping (right) of selected tasks from the autonomous vehicle usecase [35]

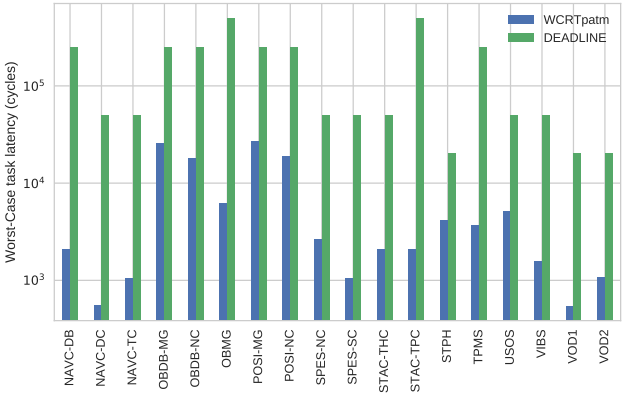


Figure 6: Analysis-based worst-case task latency of the vehicle usecase

corresponding mapping are illustrated in Figure 5 where the router R_5 (highlighted in red) is the tracked one in the experiments. The functionality selected for experiments is composed of 18 communicating tasks where all transmissions are performed periodically. Furthermore, we employ the aforementioned formal analysis [36] on the selected usecase, and the slacks of hard real-time tasks are extracted. Figure 6 plots the worst-case response time of the application' tasks and their corresponding slacks. As it can been seen, the applicability of PATM is very welcomed because of high available slacks.

Figure 7 details the power consumption of the tracked router in non-power-gating (No-PG) scheme (no power-gating is applied), and PATM along with its optimization. We break down the router power into dynamic power (switching power), leakage power, clock-tree power, and power-gating power overhead which is mainly induced by powering on/off routers and PATM unit. To be fair, we refer to the static power as the total of clock-tree, leakage, and even PG overhead. In this experiment, the power cycling overhead is trivial (not-shown). That arises from the fact that real application loads typically have low injection rates (the tracked router load is 7.2%).

As Figure 7 shows, PATM has high impact on static power savings compared with No-PG scheme under different technology libraries $65nm$ and $28nm$. It saves up to 93% of

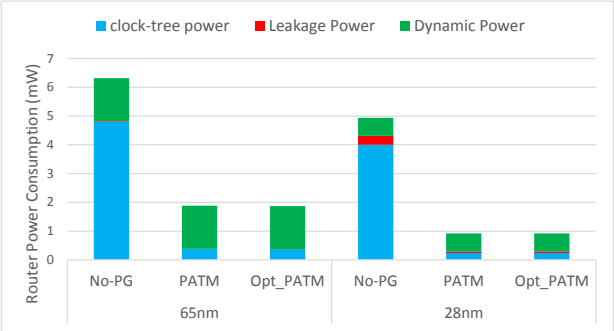


Figure 7: Router breakdown power of the vehicle usecase under 65nm and 28nm technology libraries

Table 3: Power Savings of individual routers in the 4×4 NoC, employing the autonomous vehicle usecase

(y,x)	0	1	2	3
0	90.3%	91.08%	99.37%	78.67%
1	95.90%	92.59%	68.39%	77.85%
2	79.9%	79.08%	62.29%	70.5%
3	99.17	97.94%	73.30	66.82%

the static power (a factor of 12 better). Results conducted by 28nm technology show the efficiency of PATM even at higher leakage power consumption. PATM's efficient static power savings are mainly achieved by its global controlling and then decision making. Thus, it turns off routers directly when all tasks are *off*, and accounts for *BET* rule when it must turn routers on. Moreover, as it is anticipated, the optimized PATM (Opt-PATM) has almost the same impact on power as basic PATM with very slight improvement (not shown). That is attributed to the low data rates with large volume flows in the used usecase. Thus, the router experiences very few number of switches, and enhancing PATM's functionality by optimizing it to decrease the switches number does not have a considerable impact on power. However, in agreement to the explanation introduced in Section 4, higher impact of Opt-PATM arises at higher data rates with short messages (shown in synthetic workload).

Moreover, in order to better illustrate the global insight of our approach across the whole NoC, Table 3 depicts the relative power savings of all routers in 4×4 NoC under PATM, using 65nm technology. While some of routers (highlighted in red) have to stay *on* for longer time due to high congestion/preempted tasks with reserved channels, others can be turned off faster – fulfilling higher power savings. On average the network power savings using our approach is 82.7% compared with No-PG. To be fair, we demonstrate as well the power overhead of the introduced control layer (PATM, additional links, and clients). The experimental results, using 65nm, indicate that the power overhead of PATM induces only 0.84% increase of the basic NoC power consumption. Moreover, the links (6-bit wide each) and clients induce 0.44% and 0.048% increase of the

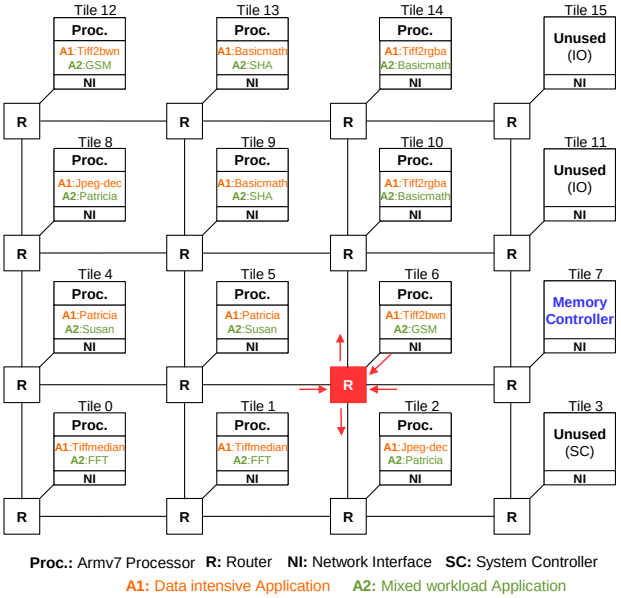


Figure 8: MiBench applications mapping on 4 × 4 NoC

basic NoC power, respectively.

After accounting for the power overhead of the control layer, the average NoC power savings using our approach become 81.4% compared with No-PG.

Regarding performance, the simulation results indicate that the absolute increase of the average packet latency employing our approach compared with No-PG is 8 cycles. Hence, the percentage increase of the average packet latency is trivial in case of big message sizes, which already demand huge number of cycles to be transmitted via NoC.

7.2. Benchmark workloads

In a second set of experiments we employ another realistic workload derived from benchmark applications to evaluate the effectiveness of the proposed approach under different aspects. For this we use benchmarks from the MiBench suite [17] as real-time applications generating traffic in the NoC. In our design, we employed two different simulation scenarios to generate two different workloads, detailed in Table 4. Moreover, we consider the technique of replicated execution, which provides reliable execution of real-time applications on unreliable hardware. To do this, each application is mapped twice to our platform (cf. Figure 8) in a Dual Modular Redundancy (DMR) configuration [3]. The first scenario is the *data intensive*, where, on average, each core performs one access (per packet) to the memory every 162 cycles. Since the packet's size is 5 flits and the NoC transmission time per link is 1 flit every 4 clock cycles, the NoC load with the first scenario is 14.1%. The second scenario is the *mixed workload*, where, on average, each core performs one access to the memory every 502 cycles. Consequently, the NoC load is 5.3%.

The benchmarks of each workload are run on the Gem5 simulator separately. An ARMv7 operating at 500MHz, with 32kB split L1 cashes and an external DDR3 memory

Table 4: MiBench applications for *Data intensive* and *Mixed workload* scenarios

Data intensive (avg. 0,00615 access/cycle) 1 access every 162 cycles		**Mixed workload** (avg. 0,00199 access/cycle) 1 access every 502 cycles	
Application	**Program**	**Application**	**Program**
Tiffmedian	[consumer]	FFT	[telecomm]
Jpeg-dec	[consumer]	Patricia	[network]
Patricia	[consumer]	Susan	[automotive]
Tiff2bwn	[network]	GSM	[telecomm]
Basicmath	[consumer]	SHA	[security]
Tiff2rgba	[automotive]	Basicmath	[automotive]

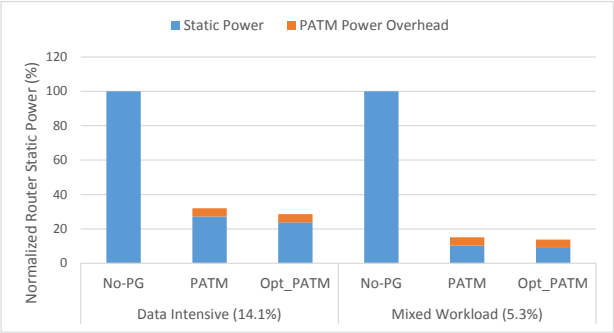


Figure 9: Router static power normalized to the basic NoC (No-PG) using MiBench Benchmark, employing *Data intensive* and *Mixed workload* scenarios

are employed. A trace with memory accesses of each application is captured and fed into OMNeT++, where delay from accesses to shared resources, such as the NoC and the DDR3, are appropriately introduced. The resulting topology and mapping are shown in Figure 8. Note that, in the experiments, we use 4 VCs with the tracked router R_6 (the closest router to the memory which is fully connected).

Regarding power savings, we assured first safe applicability of our approach employing the formal analysis framework introduced in Section (5). Next, Figure 9 details the power consumption of the tracked router under $65nm$ process technology. The Figure depicts a comparison between non-power-gating (No-PG) scheme, and PATM along with its optimization. Recall, we refer to the static power as the total of clock-tree, leakage, and even power cycling power overhead. To be fair, we also added the PATM power overhead to the router power consumption. Using *data intensive* scenario, 14.1% NoC rate, PATM achieves up to 67.96% savings of the router static power, and up to 71.41% employing the Opt-PATM. Furthermore, using the *Mixed workload*, 5.3% NoC rate, the router static power savings are 84.94% and 86.24% under PATM and Opt-PATM, respectively. As it is anticipated, the Opt-PATM has higher impact on power savings compared with basic PATM. That is mainly attributed to the capabilities of Opt-PATM, which decreases the number of router transitions between the on/off states, and thus decreases the power gating power overhead.

Regarding performance, the simulation results indicate that the relative increase of the average packet latency has a small performance penalty. Using *Data intensive*, the average latency increase employing PATM compared with No-PG is 6.3%. Additionally, the average packet latency under *Mixed workload* is only 2.8%.

7.3. Synthetic workloads

To better understand the relative aspects of PATM along with its optimization, we conduct simulations with synthetic traffic across the full range of network loads until saturation, using OMNeT++ simulator and $65nm$ technology. In these experiments, we use uniform random traffic with a 5-flit packet size. We compare PATM with the following four schemes:

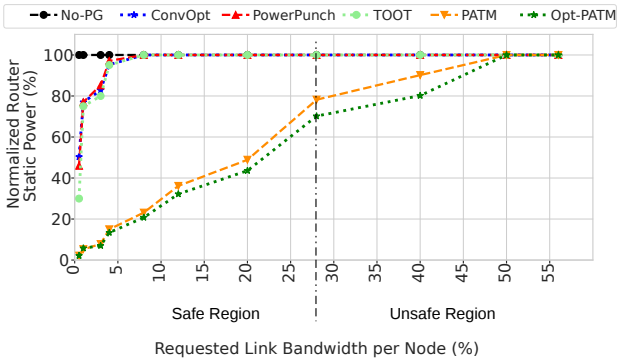
- **No-PG:** the baseline NoC.
- **ConvOpt:** like conventional power-gating schemes but it uses the early wake-up signal [26] to partially hide WU latency.
- **Power Punch:** which efficiently utilizes early power punch signal in order to completely hide routers wake-up latency.
- **TOOT:** which augments routers with TOOT component in order to increase the router sleep period.

Figure 10a compares the router static power results of PATM along with the aforementioned schemes (as reported in [9, 15]), normalized to the No-PG case.

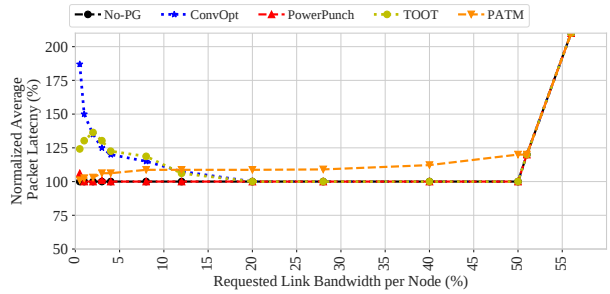
Note that the router R_5 (highlighted in red in Figure 5) power values include power-gating overhead and the control layer power consumption. Moreover, in order to illustrate the applicability of PATM under real-time systems and best-effort ones (no deadline constraints), Figure 10a plots two regions:

- **Safe region:** PATM is safely applied on the system.
- **Unsafe region:** corresponds to the best-effort case.

Using the aforementioned formal analysis (cf. Section 5) under the uniform random setup, the results indicate that the worst-case response time R_i of some interfering tasks in No-PG scheme violates the deadline after 28% requested link bandwidth. The safety standards are already violated in the basic NoC, leading obviously to stop applying any power-gating methods that in turn increase the timing violation. As it is depicted in Figure 10a, PATM considerably outperforms other schemes in static power savings. Moreover, when data rate gradually increases, we observe that PATM better saves power than others, which stop saving too early (12%). That mainly comes from the fact that other schemes do not efficiently overcome the *BET* violation, i.e., routers are turned on without accounting for *BET* which in turn, at higher rates, dramatically deceases the power savings. In contrast, PATM, which keeps



(a) Router static power normalized to No-PG case



(b) Average packet latency normalized to No-PG case

Figure 10: Router static power (a) and average packet latency (b) normalized to No-PG case using uniform random traffic

routers turned-off for an adequate time ($>BET$), decreases considerably the effect of violating BET. In other words, PATM power curve slightly increases over the full data range comparing with other curves' sharply increase. On the other hand, in unsafe region, we also observe that PATM has far better impact on static power savings than others, especially at rates under which other schemes do not achieve any savings. Moreover, as Figure 10a shows, PATM keeps saving power until NoC's saturation, and thus plays a crucial role in power savings even in best-effort systems at high requested link bandwidth. Note that once PATM stops saving power (e.g. at saturation case), and thus keeps routers *on* all the time, the control layer must be shut down, bringing the NoC back to its normal operation (only data layer). That is mainly in order not to increase the overall power induced by the useless control messages (cf. Table 1).

Furthermore, let us focus on Opt-PATM. As it can be seen, Opt-PATM in these experiments has considerable impact on power savings compared with normal PATM. That is attributed to the fact that optimized PATM not only efficiently overcomes the BET challenge but also decreases the number of routers transitions by grouping the task activations. That way, at higher rates with small message sizes, power-gating overhead, comes from power cycling a router, is significantly decreased. Note that the NoC saturates at 56% requested link bandwidth because of the congestion at output port of the tracked router in the random setup.

Regarding performance, the power savings advantage of PATM exhibits low performance penalty. To derive the PATM impact on average packet latency, and thus make it comparable with other power-gating schemes, we conduct simulations under the uniform random traffic. Figure 10b indicates that the increase of the packet average latency under PATM and at low rates is negligible. However, at high requested link bandwidth (e.g. 50%), the average latency increases by 20% compared with No-PG case. The latter increase of packet latency using PATM comes from aspects like PATM-Sender synchronization protocol and router states. As PATM is able to save power even at high data rates, it has to pay the corresponding performance penalty.

On the other hand, Figure 10b compares PATM latency overhead with other schemes, as reported in [9, 15]. ConvOpt, which optimizes conventional power-gating methods to early wake-up technique, cannot completely hide the wake-up latency. At low requested link bandwidth (e.g. 1%) where many routers are supposed to be *off*, it increases the packet average latency by 50.3% compared with No-PG. Power punch, which efficiently applies early power punch signal, is able to leave only 6.1% increase in packet latency at low rate (0.5%) as reported in [15]). Moreover, Power Punch has almost no impact on performance at higher data rates (e.g. 20%), which is achieved at the cost of less power savings opportunity comparing to PATM (cf. Figure 10a). While the previous schemes target low performance penalty at higher data rates, TOOT, on the other hand, induces higher performance penalty at higher rates because of congestion on TooT's bypass latch. When employing uniform random traffic (2.5% data rate), it increases the packet average latency by 36.4% compared with No-PG, as opposed to our approach which induces negligible latency overhead at this rate (cf. Figure 10b).

Implementation overhead: PATM implementation is highly flexible. In principle PATM could be implemented as software running on a specialized core as well as independent hardware module. In case of a software implementation, PATM performance will be lower. Thus, it increases the latency overhead of the approach. However, at the same time, PATM's design is highly flexible allowing easy updates and complex schedules. In case of a hardware implementation, PATM's design introduces additional hardware overhead and low flexibility (hard updates) but simultaneously offers high performance (thus allowing high granularity of the protocol).

In this paper, PATM is implemented as an independent hardware module. As we mentioned in the evaluation methodology, the evaluation of the area and power overhead of our approach is fulfilled using the IDAMC platform [38] for ASICs, employing $65nm$ technology. The implementation results indicate that the size of all on-core hypervisors (clients), required to synchronize the task's NoC access with PATM, is very small, only 0.037% of the basic NoC area. Furthermore, the area overhead of the additional links in the control layer is negligible compared with the

NoC area. Regarding PATM, the experimental results indicate that the area overhead of PATM increases the NoC area by 0.28% in case of 4×4 NoC (supervised by one PATM). Overall, the control layer increases the NoC size by 0.318%, which is way lower than the area overhead of other schemes that require 3.12% and 2.4% increase of the conventional scheme area for TOOT [15] and Power punch [9], respectively.

In the Sequence, we demonstrate the architecture of the proposed control layer for large NoCs.

8. PATM scalability

In this Section, the extension of the control layer is used to evaluate the proposed mechanism with much larger NoCs. To this end, we first split a NoC into subsets called regions, then assign one PATM to a dedicated region. The PATMs represent local controllers, and the number of required PATMs is basically equal to the number of the NoC regions. Recall, in case of disjoint real-time applications, we could have independent NoC regions [1], and therefore multiple independent PATMs where each of them takes care of one NoC region. However, in case of applications comprise inter-region communications, PATMs must communicate with each other using an interconnect in order to exchange each other regions' power states. Thus, we illustrate the architecture of the interconnection between PATMs using, e.g., 8×8 NoC then extend it targeting larger NoCs.

8.1. The control layer under baseline NoCs

Figure 11 depicts the control layer architecture with multiple PATMs using baseline NoC, e.g. 8×8. The NoC is split as an example into 4 different regions, each of them is controlled by one PATM (4 PATMs together in the NoC). PATMs are connected using an additional control NoC, which is only transferring control messages between PATMs. The control NoC, in case of 8×8 NoC, is composed of only one switch.

The communication policy between PATMs is presented in Figure 11. That is, R_{15} in region$_3$, connected by example to a monitor, may sporadically send data to R_{11} in region$_4$, connected by example to a memory. Overall, once PATM$_3$ receives a request message from a task in its local region, it checks whether the active task is a local or remote based on its number (considering that the task includes its number in the request message). In other words, it checks in its database whether the task is labelled as local (same region destination), or remote (remote region destination). In case of remote access, PATM$_3$ forwards the request to a remote PATM (PATM$_4$ in our example) located in the corresponding region (region 4). The request contains the corresponding source and destination of the packet in region 4. PATM$_4$ in turn looks for the current request in its database and reveals the corresponding path. Then, it checks the power state of the routers in its region, wakes them up in case of sleep mode (cf. Figure 2), and sends an *Ack_Msg* back to PATM$_3$. Once PATM$_3$ tackles the local routers, and receives an *Ack_Msg* from PATM$_4$, it acknowledges the request.

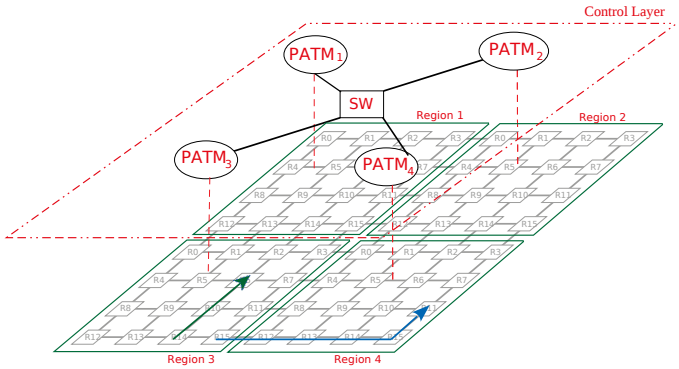


Figure 11: The control layer architecture employing 8×8 NoC

In order to better clarify the request's content (forwarded from PATM$_3$ to PATM$_4$), Table 5 addresses a local and remote tasks. While, in case of local task, PATM$_3$ acknowledges the request only based on the local routers' states, it acknowledges the request based on the local and remote routers in different regions in case of remote task. For instance, $task_{15}$ adopts the path from R_{15} in region 3 to R_{11} in region 4, and it is labelled as remote. Thus, PATM$_3$ tackles the routers in its region (R_{15} in our example), and sends a request to PATM$_4$ containing the corresponding source and destination in region$_4$ (R_{12}, R_{11}). Recall, PATM$_3$ acknowledges $task_{15}$ only after it receives an *Ack_Msg* from PATM$_4$.

Moreover, regarding the control layer, the link width of the control NoC (which connects PATMs) highly depends on the content of the control message forwarded from one PATM to another. The control message contains the Req/Rel request (required 1 bit), the source and destination of the path in the remote region. As mentioned before, PATM could run on top of different NoC architectures. However, we assume in this paper 2D mesh NoC and XY routing algorithm. Each router demands 4 bits to be addressed in, e.g., 4×4 NoC, comprising 9-bit wide link for addressing the remote path (source and destination) and the request type (Req/Rel). Moreover, the request also involves the destination address, i.e., the PATM number, which demands 2 bits in case of 4 PATMs in 8×8 NoC. Thus, the link width of the control NoC in 8×8 NoC is 11 bits. Regarding the switch in the control NoC, it is a very simple switch model. It is composed of 4 ports connecting 4 PATMs, input buffered with 1VC, and a round robin scheduling policy.

Table 5: Local and Remote tasks example

Task_Number	Corresponding Path
$task_2$ $(local)$	R_{14}, R_{10}, R_6
$task_{15}$ $(remote)$	$Reg_3 : R_{15}$, $Reg_4 : R_{12}, R_{11}$

8.2. Extending the control layer for larger NoCs

Figure 12 depicts the extension of the control layer architecture targeting, e.g., 16×16 NoC. In order to formulate analytical expressions for the proposed extension, we define the following denotations:

- N_{region}: denotes the number of NoC regions;
- N_{PATM}: denotes the number of required PATMs;
- NoC_{size}: denotes the size of the control NoC in the control layer;
- N_{SW}: denotes the number of switches in the control NoC;
- L_W: denotes the link width of the control NoC.

First, we derive the number of regions in larger NoC as follows:

$$N_{region} = NoC_{S\times S} \cdot N_{region:S\times S}, \tag{22}$$

where $NoC_{S\times S}$ denotes the number of smaller NoCs comprises the larger one; and $N_{region:S\times S}$ denotes the number of regions of each smaller NoC. We assume in our analysis symmetric square NoCs, and thus the larger NoC comprises 4 times the right smaller one. For instance, the number of regions of $NoC_{16\times16}$ is 4 times the number of regions of $NoC_{8\times8}$. That is, the number of regions of 16×16 NoC is:

$$\begin{aligned} N_{region:16\times16} &= 4 \cdot N_{region:8\times8} \\ &= 4 \cdot 4 \\ &= 16 \end{aligned} \tag{23}$$

Moreover, as we dedicate one PATM to one NoC region, the number of PATMs is always equal to the number of regions:

$$N_{PATM} = N_{region}. \tag{24}$$

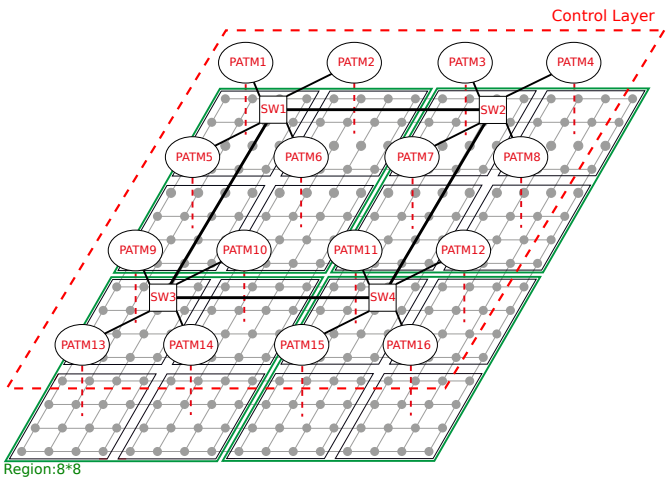


Figure 12: The control layer architecture employing 16×16 NoC

Regarding control NoC, the size of the NoC is determined as follows:

$$NoC_{size} = \frac{N_{SW}}{2} \times \frac{N_{SW}}{2}, \tag{25}$$

where the number of switches (N_{SW}) is subject to the PATMs' number, and the number of switches in the smaller NoC ($N_{SW:S\times S}$) as follows:

$$N_{SW} = \frac{N_{PATM}}{N_{PATM:S\times S}} \cdot N_{SW:S\times S}, \tag{26}$$

where $N_{PATM:S\times S}$ denotes the number of PATMs in smaller NoC. Thus, employing the aforementioned symmetry, the number of switches in 16×16 NoC is:

$$\begin{aligned} N_{SW} &= \frac{4 \cdot N_{PATM:8\times8}}{N_{PATM:8\times8}} \cdot N_{SW:8\times8} \\ &= 4 \cdot N_{SW:8\times8} \\ &= 4 \cdot 1 \\ &= 4 \end{aligned} \tag{27}$$

Regarding the link width of the control layer, it can be calculated using the following equation:

$$L_W = L_{Path} + L_D \tag{28}$$

$$L_D = \lceil log_2(N_{PATM}) \rceil, \tag{29}$$

where L_{Path} denotes the number of bits required to encode the remote path in the request type; and L_D denotes the number of bits required to encode the destination (the PATM). Recall, L_{Path} is 9-bit wide, and thus, in case of 16×16, the link width is:

$$L_W = 9 + \lceil log_2(16) \rceil = 13 \; bits. \tag{30}$$

Based on the aforementioned analysis, Table 6 demonstrates the control layer infrastructure of our approach under different NoC sizes.

Table 6: The control layer infrastructure for different NoC sizes

Basic NoC	N_{PATM}	N_{SW}	NoC_{size}	L_W (bits)
8 × 8	4	1	1 × 1	11
16 × 16	16	4	2 × 2	13
32 × 32	64	16	4 × 4	15

Moreover, Figure 13 depicts the PATMs and switches increase under different basic NoC sizes. As it can be seen, PATMs increase exponentially to the size of the basic NoC. However, under larger NoC sizes, e.g. 32×32, the increase becomes linear. That is mainly attributed to the relative increase of the NoC regions. In other words, the absolute increase of the NoC regions from 16×16 to 32×32 is 48 regions (Inc=48), which, obviously, demands larger number of PATMs. That way, the larger the NoC is, the conservative power savings is achieved employing the corresponding PATMs.

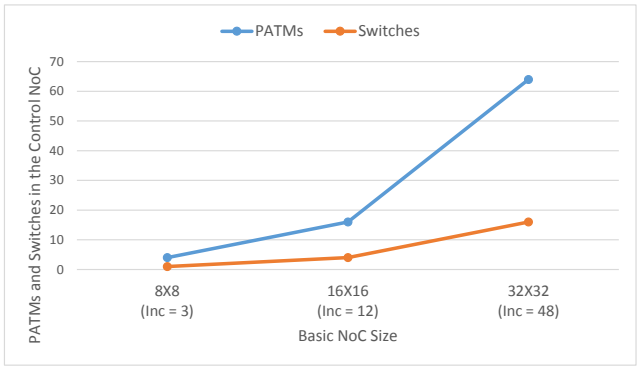


Figure 13: PATMs scalability under different basic NoC sizes

8.3. Timing analysis extension under scalability

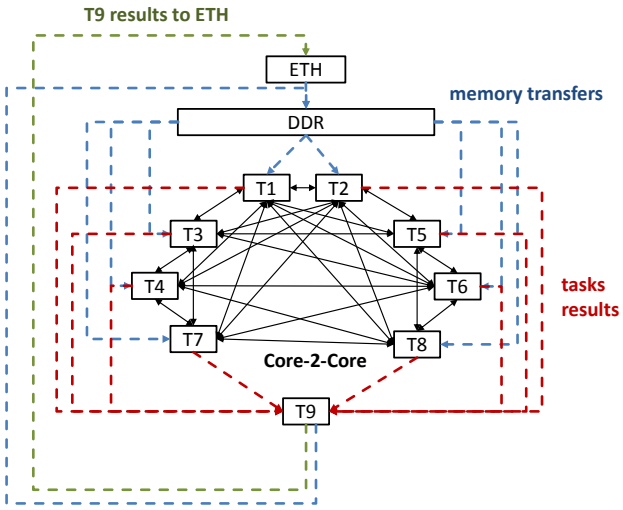
Regarding the additional latency overhead induced by the communication between PATMs through the control NoC, the worst-case latency overhead of our approach demonstrated in Equation 13 is modified as follows:

$$\begin{aligned} MLO^{Del}_{PATMs} = \; & R_i^{Ctrl} + R_i^{LPATM} + R_i^{LPATMDel} \\ & + 2 \cdot R_i^{Trans} \cdot NbrRPATMs \\ & + \max_{RPATM \in \mathcal{S}_{PATMs}} (R_i^{RPATM}) \\ & + \max_{RPATMDel \in \mathcal{S}_{PATMs}} (R_i^{RPATMDel}), \end{aligned} \tag{31}$$

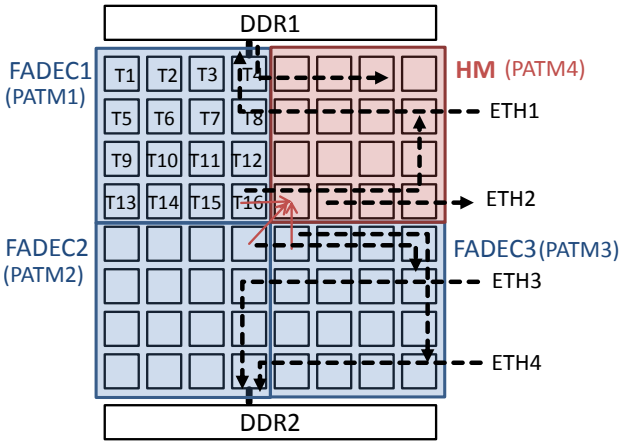
where R_i^{LPATM}: denotes the worst-case processing time required by local PATM ($LPATM$) to process a request; $max(R_i^{RPATM})$: denotes the maximum worst-case processing time from the whole set of remote PATMs ($\mathcal{S}_{PATMs}$) to process a request forwarded from the local PATM; R_i^{Trans}: denotes the worst-case transmission time required by the control NoC to transfer a request from/to the local PATM to/from the remote one; $R_i^{LPATMDel}$ denotes the worst-case delay a task i may experience by a Local PATM, induced by power-gating approach in case the local routers are turned off; $max(R_i^{RPATMDel})$: denotes the maximum worst-case delay from the whole set of remote PATMs, induced by power-gating approach in case the remote routers are turned off; R_i^{Ctrl} is the same in Equation 14.

As discussed earlier, both R_i^{LPATM} and R_i^{RPATM} correspond to the processing time required by PATM to process a request (cf. Equation 15). Both $R_i^{LPATMDel}$ and $R_i^{RPATMDel}$ correspond to the worst-case delay induced by power-gating in case the local/remote routers are turned off (cf. Equation 19).

Furthermore, in multiple regions case, PATM must create a request and send it through the control NoC to remote PATMs. Thus, R_i^{Trans} is needed in order to analyse the control NoC delay. We can use any standard NoC analysis frameworks e.g. [43, 24, 25] in order to derive the worst-case transmission latency required by the control NoC. We employed in this paper the holistic analysis [36] introduced in Section 5. The R_i^{Trans} factor is multiplied by two, one corresponds to the request transmission from the local PATM to the remote one, and the second one corresponds to the acknowledgement transmission from the remote PATM to the local one. Moreover, to account for multiple remote PATMs, the transmission factor is also multiplied by the number of remote PATMs ($NbrRPATMs$).



(a) Task graph of FADEC application



(b) Mapping of three FADEC applications and one HM application

Figure 14: Task graph of avionic usecase - FADEC application (a), and mapping of three FADEC applications and one HM application (b) using 8×8 NoC

8.4. Simulation results

We consider the usecase from [1], which is from safety critical domain. The critical applications are modelled based on the Full Authority Digital Engine Control (FADEC) application, see Figure 14a. For achieving Triple Modular Redundancy (TMR), FADEC application is executed in three regions in parallel. We consider as well the Health Monitoring (HM) application, which is gathering the results from FADEC instances in order to monitor their functionality in the MPSoC. In case of erratic behaviour i.e. receiving different results from FADEC instances, HM outputs an error message via ETH2. Thus, based on our usecase where

4 applications are employed, we use 4 PATMs (one PATM per application).

The considered workflow of FADEC is described in the scope of actions conducted by tasks. First, it receives from an Ethernet interface sensors data from an engine 90 kB. These data are stored into the DDR memory and later distributed to n tasks, noted T0 to Tn. During their parallel execution these tasks, except the last one Tn, exchange data in direct transmissions. Task periods vary between 0.5 ms to 1.2 ms, and communication volumes vary between 2.2 kB and 60 kB. Finally, they provide results to the last task Tn, which stores 30 kB in the DDR memory, and additionally sends back 15 kB through the same Ethernet interface. Figure 14a presents the graph of the communication in the FADEC application, assuming 9 tasks. In our example, presented in Figure 14b, we scale the communication to reach n = 16 tasks. The fourth region (red) in Figure 14b considers HM application. Note that to avoid high inter-region sporadic traffic due to dependencies between applications, we mapped tasks belong to different regions right next to each other (e.g. the orange arrows in Figure 14b).

Moreover, we assume a MPSoC with two independent single port memory modules (DDR1 and DDR2) as well as four Ethernet ports (ETH1-4). Therefore, FADEC1 uses ETH1 and DDR1, HM uses ETH2 and DDR1, FADEC2 uses ETH3 and DDR2, and finally FADEC3 uses ETH4 and DDR2.

Regarding power savings, we assured first safe applicability of PATMs employing the formal analysis framework introduced in Sections (5, 8.3). Next, we run the simulation, using $65nm$ technology, and the experimental results indicate that PATMs save up to 79.13% of the static power compared with No-PG NoC after accounting for the power overhead of the control layer.

9. Conclusion

NoCs designed for hosting real-time applications require both efficient real-time guarantees and tight power limitations. In this paper, PATMs have been proposed in order to safely save static power on NoC routers. PATMs, by applying their knowledge of the system state, turn effectively idle routers on/off after accounting for *BET* rule, resulting in high efficient power-gating approach. Two existing slacks *Hop-Count* slack and *task's deadline slack* have been exploited in order not to violate *BET*. Moreover, PATM has been optimized for periodic traffic, fulfilling higher power savings. Furthermore, a formal worst-case timing analysis has been provided in order to safely apply our approach with complying to timing guarantees.

Experimental results, with a realistic application, have clearly showed that compared with No-PG NoCs our approach saves, on average, up to 81.4% and 79.13% of the static power of 4×4 and 8×8 NoCs, respectively. Moreover, using synthetic workloads, our approach considerably saves power even at high NoCs utilizations (cf. Figure 10a), inducing very small area penalty.

Acknowledgements

This work was partially funded within the DFG projects ER168/24 and ER168/32.

References

[1] Abdallah, L., Jan, M., Ermont, J., Fraboul, C., 2016. Reducing the contention experienced by real-time core-to-i/o flows over a tilera-like network on chip. In: Real-Time Systems (ECRTS), 2016 28th Euromicro Conference on. IEEE, pp. 86–96.

[2] Audsley, N., Burns, A., Richardson, M., Tindell, K., Wellings, A. J., 1993. Applying new scheduling theory to static priority pre-emptive scheduling. Software Engineering Journal 8 (5), 284–292.

[3] Axer, P., Sebastian, M., Ernst, R., 2011. Reliability analysis for MPSoCs with mixed-critical, hard real-time constraints. In: Proceedings of the international conference on Hardware/software codesign and system synthesis (CODES+ISSS). pp. 149–158.

[4] Banerjee, A., Wolkotte, P. T., Mullins, R. D., Moore, S. W., Smit, G. J., 2009. An energy and performance exploration of network-on-chip architectures. IEEE Transactions on Very Large Scale Integration (VLSI) Systems 17 (3), 319–329.

[5] Ben-Itzhak, Y., Zahavi, E., Cidon, I., Kolodny, A., 2012. Hnocs: modular open-source simulator for heterogeneous nocs. In: Embedded Computer Systems (SAMOS), International Conference on. IEEE, pp. 51–57.

[6] Casu, M., Yadav, M., Zamboni, M., 2013. Power-gating technique for network-on-chip buffers. Electronics Letters 49 (23), 1438–1440.

[7] Chattopadhyay, S., Kundu, S., 2014. Network-on-Chip: The Next Generation of System-on-Chip Integration. CRC press.

[8] Chen, L., Pinkston, T. M., 2012. Nord: Node-router decoupling for effective power-gating of on-chip routers. In: Microarchitecture (MICRO), 45th Annual IEEE/ACM International Symposium on. IEEE, pp. 270–281.

[9] Chen, L., Zhu, D., Pedram, M., Pinkston, T. M., 2015. Power punch: Towards non-blocking power-gating of noc routers. In: High Performance Computer Architecture (HPCA), IEEE 21st International Symposium on. pp. 378–389.

[10] Das, R., Narayanasamy, S., Satpathy, S. K., Dreslinski, R. G., 2013. Catnap: Energy proportional multiple network-on-chip. ACM SIGARCH Computer Architecture News 41 (3), 320–331.

[11] Daya, B. K., Chen, C.-H. O., Subramanian, S., Kwon, W.-C., Park, S., Krishna, T., Holt, J., Chandrakasan, A. P., Peh, L.-S., 2014. Scorpio: a 36-core research chip demonstrating snoopy coherence on a scalable mesh noc with in-network ordering. In: ACM SIGARCH Computer Architecture News. Vol. 42. IEEE Press, pp. 25–36.

[12] Durrieu, G., Faugere, M., Girbal, S., Pérez, D. G., Pagetti, C., Puffitsch, W., 2014. Predictable flight management system implementation on a multicore processor. In: Embedded Real Time Software (ERTS'14).

[13] Escamilla, J. V., Flich, J., Casu, M. R., 2017. Icaro-papm: Congestion management with selective queue power-gating. In: High Performance Computing & Simulation (HPCS), 2017 International Conference on. IEEE, pp. 259–266.

[14] Fallin, C., Craik, C., Mutlu, O., 2011. Chipper: A low-complexity bufferless deflection router. In: High Performance Computer Architecture (HPCA), IEEE 17th International Symposium on. IEEE, pp. 144–155.

[15] Farrokhbakht, H., Taram, M., Khaleghi, B., Hessabi, S., 2016. Toot: an efficient and scalable power-gating method for noc routers. In: Networks-on-Chip (NOCS), Tenth IEEE/ACM International Symposium on. pp. 1–8.

[16] Ghaderi, Z., Alqahtani, A., Bagherzadeh, N., 2017. Online monitoring and adaptive routing for aging mitigation in nocs. In: Proceedings of the Conference on Design, Automation & Test in Europe. European Design and Automation Association, pp. 67–72.

[17] Guthaus, M. R., et al., 2001. MiBench: A free, commercially representative embedded benchmark suite. In: IEEE International Workshop on Workload Characterization, WWC-4.

[18] Hesse, R., Jerger, N. E., 2015. Improving dvfs in nocs with coherence prediction. In: Proceedings of the 9th International Symposium on Networks-on-Chip. ACM, p. 24.

[19] Hoskote, Y., Vangal, S., Singh, A., Borkar, N., Borkar, S., 2007. A 5-ghz mesh interconnect for a teraflops processor. IEEE Micro 27 (5), 51–61.

[20] Hu, Z., Buyuktosunoglu, A., Srinivasan, V., Zyuban, V., Jacobson, H., Bose, P., 2004. Microarchitectural techniques for power gating of execution units. In: Proceedings of the international symposium on Low power electronics and design. ACM, pp. 32–37.

[21] Kadeed, T., Rambo, E. A., Ernst, R., 2017. Power and area evaluation of a fault-tolerant network-on-chip. In: System-on-Chip Conference (SOCC), 30th IEEE International. IEEE, pp. 190–195.

[22] Kostrzewa, A., Saidi, S., Ecco, L., Ernst, R., 2016. Dynamic admission control for real-time networks-on-chips. In: Design Automation Conference (ASP-DAC), 21st Asia and South Pacific. IEEE, pp. 719–724.

[23] Liu, F., Narayanan, A., Bai, Q., 2000. Real-time systems.

[24] Liu, M., Becker, M., Behnam, M., Nolte, T., 2017. A tighter recursive calculus to compute the worst case traversal time of real-time traffic over nocs. In: Design Automation Conference (ASP-DAC), 2017 22nd Asia and South Pacific. IEEE, pp. 275–282.

[25] Long, Y., Lu, Z., Shen, H., 2018. Composable worst-case delay bound analysis using network calculus. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 37 (3), 705–709.

[26] Matsutani, H., Koibuchi, M., Wang, D., Amano, H., 2008. Runtime power gating of on-chip routers using look-ahead routing. In: Proceedings of the Asia and South Pacific Design Automation Conference. IEEE Computer Society Press, pp. 55–60.

[27] Mellanox, 2004. Technologies performance, price, power, volume metric (pppv). http://www.mellanox.com/products/shared/PPPV.pdf.

[28] Mullins, Robert, 2006. Minimising dynamic power consumption in on-chip networks. In: System-on-Chip, International Symposium on. IEEE. pp. 1–4.

[29] Nasirian, N., Soosahabi, R., Bayoumi, M., 2016. Traffic-aware power-gating scheme for network-on-chip routers. In: Circuits and Systems Conference (DCAS), IEEE Dallas. pp. 1–4.

[30] Parikh, R., Das, R., Bertacco, V., 2014. Power-aware nocs through routing and topology reconfiguration. In: Design Automation Conference (DAC), 51st ACM/EDAC/IEEE. pp. 1–6.

[31] Rambo, E. A., Seitz, C., Saidi, S., Ernst, R., 2017. Bridging the gap between resilient networks-on-chip and real-time systems. IEEE Transactions on Emerging Topics in Computing.

[32] Ripoll, I., Ballester-Ripoll, R., 2013. Period selection for minimal hyperperiod in periodic task systems. IEEE Transactions on Computers 62 (9), 1813–1822.

[33] Shang, L., Peh, L.-S., Jha, N. K., 2003. Dynamic voltage scaling with links for power optimization of interconnection networks. In: High-Performance Computer Architecture, HPCA-9 Proceedings. The Ninth International Symposium on. IEEE, pp. 91–102.

[34] Shi, Z., Burns, A., 2008. Real-time communication analysis for on-chip networks with wormhole switching. In: Networks-on-Chip, NoCS Second ACM/IEEE International Symposium on. IEEE, pp. 161–170.

[35] Shi, Z., Burns, A., Indrusiak, L. S., 2012. Schedulability analysis for real time on-chip communication with wormhole switching. Innovations in embedded and real-time systems engineering for communication, 198.

[36] Soares Indrusiak, L., Burns, A., Nikolic, B., 2018. Buffer-aware bounds to multi-point progressive blocking in priority-preemptive nocs. In: Proceedings of the Design, Automation & Test in Europe Conference (DATE). York.

[37] Stankovic, J. A., Ramamritham, K., Spuri, M., 1998. Deadline Scheduling for Real-Time Systems: Edf and Related Algorithms. Kluwer Academic Publishers, Norwell, MA, USA.

[38] Tobuschat, S., Axer, P., Ernst, R., Diemer, J., 2013. Idamc: A noc for mixed criticality systems. In: Embedded and Real-Time Computing Systems and Applications (RTCSA),19th International Conference on. IEEE, pp. 149–156.

[39] Tobuschat, S., Ernst, R., 2017. Real-time communication analysis for networks-on-chip with backpressure. In: Proceedings of the Conference on Design, Automation & Test in Europe. European Design and Automation Association, pp. 590–595.

[40] Wang, H., Peh, L.-S., Malik, S., 2003. Power-driven design of router microarchitectures in on-chip networks. In: Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture. IEEE Computer Society, p. 105.

[41] Wang, H.-S., Peh, L.-S., Malik, S., 2003. A power model for routers: Modeling alpha 21364 and infiniband routers. IEEE Micro 23 (1), 26–35.

[42] Zhan, J., Stoimenov, N., Ouyang, J., Thiele, L., Narayanan, V., Xie, Y., 2013. Designing energy-efficient noc for real-time embedded systems through slack optimization. In: Proceedings of the 50th Annual Design Automation Conference. ACM, p. 37.

[43] Zhan, J., Stoimenov, N., Ouyang, J., Thiele, L., Narayanan, V., Xie, Y., 2014. Optimizing the noc slack through voltage and frequency scaling in hard real-time embedded systems. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 33 (11), 1632–1643.