Latency-Aware Frequency Scaling in Time-Triggered Network-on-Chip Architecture

Rakotojaona Nambinina, Daniel Onwuchekwa, Roman Obermaisser
Chair for Embedded Systems
University of Siegen
Siegen, Germany
{Andrianoelisoa.Rakotojaona, Daniel.Onwuchekwa, Roman.Obermaisser@uni-siegen.de}

***Abstract*—Low power consumption is one of the major design requirements for Network-on-Chip (NoC) based multi-core architectures. Scaling the voltage and frequency of NoC during run-time allows to optimize power consumption within the chip. However, the associated latency increase and throughput degradation limit its use. Scaling the frequency of routers can affect the performance of the NoC, as the frequency of routers may be scaled down while they are still active. On the other hand, using time-triggered communication in the NoC ensures predictability and deterministic communication and facilitates frequency scaling at the router level according to a predefined schedule. Consequently, the time-triggered scaling of router frequency optimizes the power consumption of the NoC. It also preserves the performance of the NoC by adjusting the frequency of routers at a specific time defined by the schedule and guaranteeing that each router's frequency is clock-gated only when it is idle. In this paper, we discuss the architecture of a time-triggered NoC equipped with power-saving techniques that enable frequency scaling of the NoC routers while preserving the system's performance and predictability.**

***Index Terms*—NoC, time-triggered, frequency scaling, multi-core architecture, power-saving.**

I. Introduction

In 1965, Moore predicted that the number of transistors per integrated circuit would double every two years. To this day, the semiconductor industry has successfully adhered to Moore's prediction, leading to the current state of the art in system-on-chip architectures. With today's advanced technology, up to a billion transistors can be integrated into a single chip, thus optimizing the performance of embedded devices and opening up the possibility of using a multi-core architecture on a single chip. The standard shared bus is often used in multi-core architectures to communicate within the SoC. However, using a shared bus is unsuitable when the number of cores within the SoC increases since it does not support simultaneous communication between multiple cores, which increases the delay when multiple cores need to exchange data. Network-on-Chips (NoCs) were introduced to solve this communication problem of the shared bus. NoCs consist of routers, links, and network interfaces connected to processing elements. Using an NoC to connect multiple cores consumes more power than a standard shared bus. Therefore, power optimization techniques in NoC-based multi-core systems play an important role. There are several techniques for improving power consumption in multi-core architectures. These techniques include clock gating, power gating, and dynamic voltage and frequency scaling (DVFS), which are used at the core or NoC level. However, using power-saving techniques can impact NoC performance because scaling the voltage or frequency of cores or routers during run-time increases message latency. It can result in messages not arriving on time, which is not desirable in real-time applications. However, using deterministic and predictable communication such as Time-Triggered NoC (TTNoC) for communication between multiple cores facilitates frequency scaling in the NoC routers since the router's active time and idle time can be predicted. Thus, allow the power-saving techniques built in the TTNoC to clock-gated the idle router, while adjusted the frequency of active routers based on different deadlines and thus slack. In addition, different routers might serve multiple messages at the same time, needing a higher frequency (e.g., North to South and West to East). In this work, we propose latency-aware frequency scaling using a time-triggered network-on-chip (TTNoC) architecture. This work is an extension of previous work presented in [7], [8] with a distributed frequency controller deployed in each router. The frequency controller is responsible for adjusting the frequency of each router over time depending on the configuration of the schedule rather than using a central controller. It also supports multiple frequency levels to scale the frequency of NoC routers for energy optimization. The schedule used for scaling the frequency of each router is computed offline, and this is an optimization problem for a scheduler, which is not addressed in this paper. In this work, we evaluate the resources used by the proposed architecture in an FPGA and perform latency measurements to evaluate the performance analysis of the proposed architecture compared to its baseline LISNoC introduced in [8]. In addition, we assess the power consumption of the proposed techniques compared to the baseline without frequency scaling using ORION 3 [4], which is an NoC power estimation tool. The rest of this paper is organized as follows. Section II discusses related work. Section III discusses the system model. Section IV discusses the experimental results, and section V concludes the paper.

II. Related Work

NoC power consumption in multi-core architectures accounts for a significant portion of the total chip. Many research results [3], [5], [14] have confirmed that NoC power consumption is around 10-36%. Therefore, reducing NoC power consumption is important for developing a multi-core platform. One way to solve the power problem in NoC-based multi-core systems is to use power-saving techniques. Several studies have proposed DVFS, clock gating, and power gating to minimize the power consumption of NoC-based multi-core architecture [16]. However, these results focus on power saving in the processor or cache. Recently, DVFS has been proposed for NoCs to optimize power dissipation. Some previous works propose similar techniques for NoCs [1], [16], [7], [6], [17] by scaling the voltage/frequency of each router. Meanwhile, there are studies [2], [9] on static voltage and frequency allocation for NoC components by partitioning and optimizing voltage islands. However, these results are still for non-real-time systems. In [7], time-triggered frequency scaling, called TTFS, was introduced to enable frequency scaling in NoC routers according to a predefined schedule. However, the proposed architecture uses a centralized controller to scale the frequency of NoC routers. This centralized controller is sensitive to failure. Therefore, we extended the TTFS using a distributed controller that scales the frequency of routers according to a schedule. We also use two clock domains (full and half frequency mode) to scale the frequency of the active routers. However, the concept is more generic to support multiple clock domains.

III. System Model

In time-triggered NoC (TTNoC), starting tasks or injecting messages follows an offline precomputed schedule that ensures each message or task accesses resources at a specific time to avoid resource conflicts [10]. In addition, the TTNoC is equipped with power-saving techniques to optimize the NoC energy consumption. The power-saving techniques enable frequency scaling of the NoC routers according to a schedule. The TTNoC architecture supports different topologies such as meshes and connects heterogeneous resources such as hard and soft processors, memory subsystems, etc. Figure 1 provides an overview of the TTNoC architecture. The architecture consists of multiple cores interconnected by an NoC. The NoC consists of routers, and each router is connected to other routers and the tiles through communication links. The NI enables a resource to access the NoC. The TTNoC has a frequency controller that controls the operating frequency of the routers according to the configuration in the NI. This configuration is based on information from the TTNoC schedule.

A. LISNoC Overview

LISNoC served as the basis for implementing the TTNoC with power saving-techniques. It is an open-source NoC implemented in Verilog, mainly used for academic purposes [15]. The main features of LISNoC include virtual channel

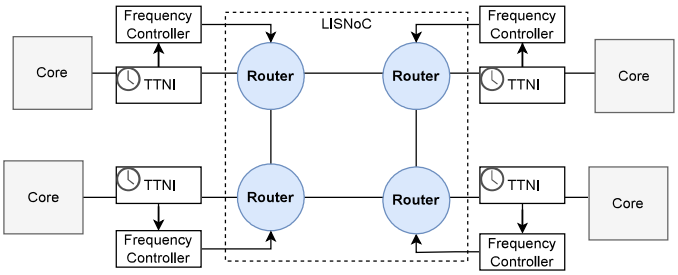

Fig. 1. Architecture of TTNoC with power saving techniques

support, flexible configuration, wormhole routing, and round-robin arbitration for link multiplexing [15]. LISNoC uses a packet format for data transmission, where each packet is divided into flits with a header containing all information about the destination processing element. The LISNoC presented in [15] has been extended to include source-based routing and a TTNI that supports time-triggered communication to enable deterministic communication [8]. In addition, the LISNoC router has been extended to support frequency scaling so that the frequency of the routers can be adjusted during run-time.

B. Power model

In digital circuits, the power can be modeled in two phases: active and standby. In the active phase, an input data is pass through the module and produces an output. In the standby phase, the module is idle. Dynamic power consists of switching and short-circuit power, while static power consists of leakage current, which is the current that flows through the transistor when there is no activity. The total power consumption in a CMOS-based transistor circuit has been described in [13], [7]. It can be expressed by the following equation:

$$P_{total} = P_{Dyn} + P_{Stat} \tag{1}$$

Where P_{Dyn} is the dynamic power, and P_{Stat} is static power. The dynamic power is described in equation (2):

$$P_{Dyn} = P_{SW} + P_{SC} \tag{2}$$

P_{SW} is the switching power, and P_{SC} is the short circuit power. The switching power and the short circuit power are further described in equation (3) and (4).

$$P_{SW} = \alpha \cdot F \cdot C_L \cdot (V_{dd})^2 \tag{3}$$

Where α is the activity factor, F is the clock frequency, C_L is the load capacitance, and V_{dd} is the power voltage.

$$P_{SC} = 10\% \cdot P_{dyn} \tag{4}$$

The static power is due to the leakage current in the individual logic blocks, which is proportional to the supply voltage and can be expressed by the following relation:

$$P_{Stat} \approx \cdot V_{dd} \tag{5}$$

$$P_{Stat} \approx \beta \cdot V_{dd} \cdot e^{-V_{th}/\gamma \cdot V_T} \quad (6)$$

where β and γ are experimentally derived constants , and V_{dd} is the power voltage, and V_{TH} is the threshold voltage and V_T is the Boltzmann thermal voltage that is linearly proportional to the temperature [13].

C. Frequency Scaling in Time-Triggered NoC

In TTNoC, communication and computations are controlled by a scheduler [12]. The scheduler is used to schedule the injection time of messages into the NoC. It ensures that messages are transmitted between the sender and receiver according to a fixed message transmission schedule established during design time and is guaranteed to be collision-free. The NI uses a global time base to ensure a consistent timing view. Since TTNoC communication is deterministic and predictable, it is possible to predict the time in which the router received flits within the period, which facilitates frequency scaling of the router and allows it to adjust its frequency during run-time according to a schedule. The scheduler clock gates the idle routers and operates the active routers at a frequency determined by the schedule. Moreover, the scheduler ensures that the router's frequency is clock-gating only when no message passes through to maintain the NoC's performance and ensure that critical messages meet the deadline. Figure 2 showed how the frequency of different routers is scaling according to a schedule.

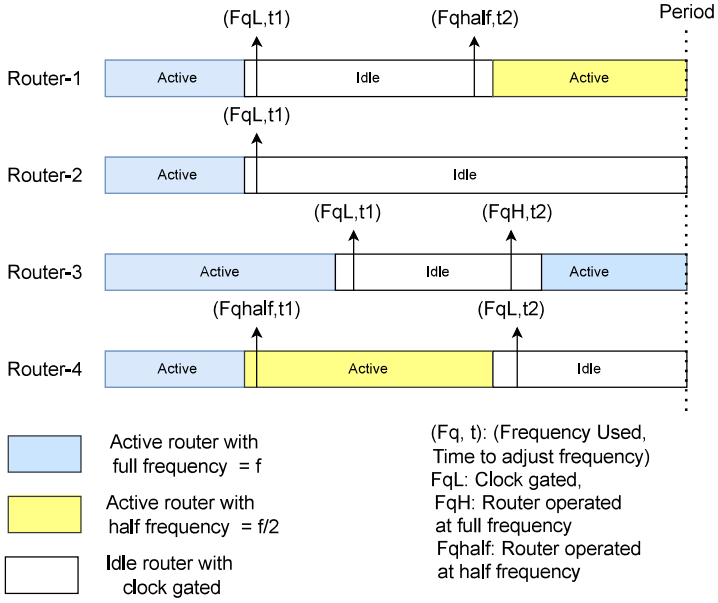

Fig. 2. Example of frequency Scaling in 4 routers according to a schedule

As shown in Figure 2, each router can be operated with multiple clocks domain such as full frequency, half frequency, and clock gating. The scheduler activates the frequency scaling at a specific time defined by the schedule. During idle time, the router frequency is clock gated. However, for active routers, the operating frequency may be in full or half-frequency mode, depending on the value specified in the schedule. Figure 3 shows the structure of the TTNoC schedule entries, which are stored as a circularly linked list. The schedule format of the TTNoC contains five values, namely IstMsg, IstFreq, PortID, FrqMode, and Next, as described below.

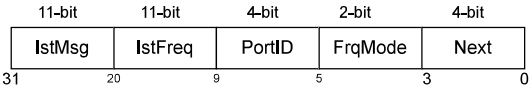

Fig. 3. Structure of schedule entries for messages transmission in TTNoC

- IstMsg: specifies when the messages in a particular PortID in the core interface are fed into the NoC.
- IstFreq: represents when the frequency of the router is adjusted.
- PortID: Each sub-memory in the NoC core interface is assigned a PortID from which messages are fed into the NoC. Each PortID is associated with the path of the packets.
- FreqMode represents the frequency used in the router. For active routers the frequency should be operated with a full or half frequency mode. The idle routers are clock gated.
- Next is a reference to the next entry in the schedule.

Figure 4 shows an example of a schedule entry in the TTNoC. Two entries are distinguished by color: Message entries are white, and frequency entries are gray. The number in the schedule entry shown in Figure 4 represents the address of the entry in the schedule file. Selecting the correct schedule starts by tracking the TTNoC schedule entries by address and next pointers. The scheduler is responsible for triggering message injection or frequency scaling at the time specified in the schedule entries of TTNoC. As shown in Figure 4, entry 0 is a message entry, which means that the message with the corresponding PortID will be transmitted at the time specified by the entry. For message-type entries, the next value refers to the address of the next entry. Here, entry 1 is a frequency entry, meaning that the router's frequency needs to be adjusted with the frequency mode specified by the schedule entry. The same operation is applied to the other entries in the TTNoC schedule. After the operations are performed for all entries, the next pointer of the last entry follows, pointing back to the first entry of the period [10].

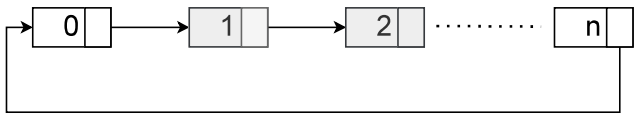

Fig. 4. Linked-list TTNoC schedule with n-number of entries

D. Proposed Network Interface Architecture

The network interface is the interface that connects the core to a routers. The time-triggered NI (TTNI) is a temporal

and spatial partitioning layer built on top of the NoC. Figure 5 shows the building blocks of the TTNI. It consists of five building blocks, namely the core interface, scheduler, packetization, depacketization, and a memory to store the path for source-based routing. The components of the TTNI are

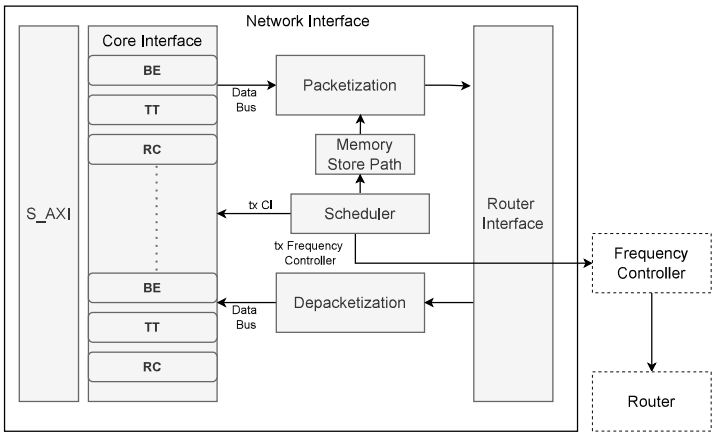

Fig. 5. Proposed Network Interface Architecture

described below:

- Core interface and AXI wrapper: The main function of the core interface is to provide the required communication service between the core and the NI. The core interface has two ports, an input port and an output port. The output port is used to queue messages from the core, while the input ports forward messages from the NoC side to the core. The core interface is equipped with an AXI wrapper for efficient communication with the core. The core interface sends the message in its specific port according to the time specified in the schedule.
- Scheduler: This module is responsible for controlling communication within the NoC by feeding messages at a predefined time. In addition, the scheduler is also responsible for triggering frequency scaling in the router according to a predefined time defined in the schedule entries. As shown in Figure 6, the scheduler consists of a

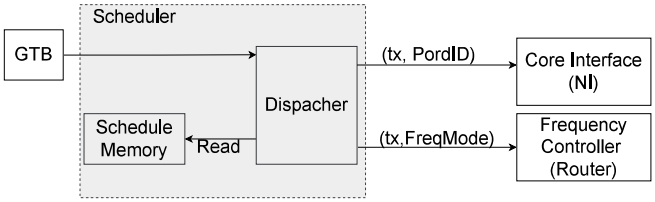

Fig. 6. Scheduler Architecture

schedule memory and a dispatcher. The schedule memory is responsible for storing the schedule entries of the TTNoC. The dispatcher reads the schedule information from the schedule memory and compares it with the current value of the global time base (GTB). When the progression of the GTB reaches a certain point of the entry from the schedule memory, the core interface or frequency controller is triggered by the scheduler. The core interface is triggered when the schedule entry is for messages, and the frequency controller is triggered when the schedule entry is for frequency. Figure 7 illustrates the state machine of a dispatcher.

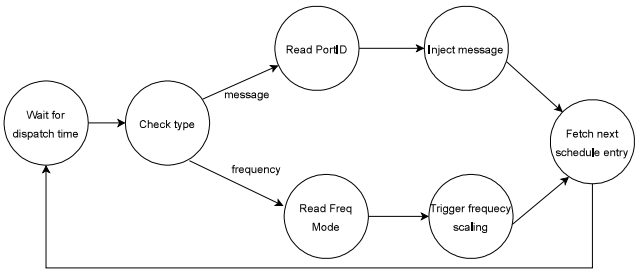

Fig. 7. Scheduler state machine

The state machine transits to the next state at the time defined by the TTNoC schedule entries. At this time the type of the TTNoC schedule entries is checked. For message type entries, the state machine reads the PortID and injects the message of the specified port at the instant message defined by the schedule entries. After injection, the new entry is fetched based on the next pointer in the TTNoC schedule entries. In the case of a frequency schedule entry, the state machine reads the frequency mode (FreqMode) and causes the frequency controller to adjust the frequency of the router based on the frequency mode extracted from the schedule entries, after which the new entry is fetched based on the next pointer in the TTNoC schedule entries. After selecting the next TTNoC schedule entries, the state machine waits again for the dispatch time.

- Packetization and depacketization: Packetization ensures that messages from the core interface are encoded before they are sent to the router. Depacketization receives the messages from the router of the NoC and decodes the messages before they are written to the core interface.
- The router interface is an interface that connects both the router and NI as well as router to router.

E. Proposed Router Architecture

The proposed router architecture used in TTNoC is described in this section. The router consists of a set of input and output buffers, a connection matrix, and a control unit such as virtual channel (VC) allocator and SW allocator. The buffers at the inputs and outputs are used to queue the data transmitted over the channels. Buffers allow local storage of data that cannot be forwarded immediately. Each router has five channels corresponding to South, East, North, and Local. The South, East, North, and West channels are responsible for interconnection between neighboring routers, while the Local channel is responsible for communication with the core. The proposed router uses the source-based routing algorithm to route data from the input port to the output port of the router.

This means that the routing decision for the output port is made in the router, at least depending on the routing opcode, and the paths of each packet are defined in the first flit or header. Wormhole flow control is used in TTNoC because it supports low latency, high speed, and guaranteed delivery of packets, making it suitable for real-time communication [11]. The generic router architecture is extended to support frequency scaling with a frequency controller that re-configures the operating frequency of the router during run-time. This frequency controller receives configuration data from the neighboring NI. Figure 8 shows the architecture of the proposed router.

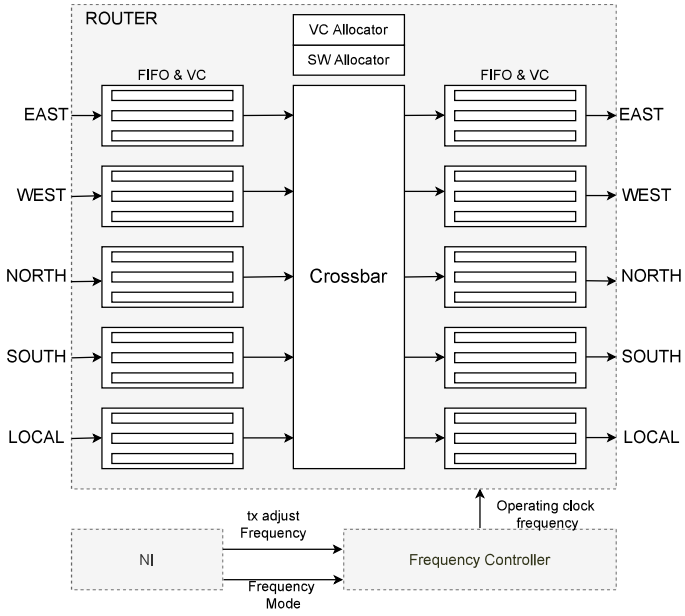

Fig. 8. Proposed router architecture

As shown in Figure 8, the proposed router architecture consists of three main components, which are explained below:

- First-In-First-Out buffer: This memory is used to buffer incoming and outgoing data in the router. The buffers of the FIFO are replicated in the input and output ports of the router for communication over virtual channels.
- Crossbar: This component is used to connect the input and output ports of the router. All possible input data lines are connected to the input ports of the crossbar multiplexers. The output data of the input data line is then controlled by the arbiter via high priority messages.
- Frequency Controller: It optimizes the power consumption of the router. Each NoC router is equipped with a frequency controller. This component is responsible for configuring the frequency used by each router. The router's frequency is clock gated when the router is idle and the scheduler assigns a frequency to routers when the router is in active mode. The operating frequency of the router is extracted from the schedule. The frequency controller is a state machine that reads the configuration from the NI, such as the trigger frequency and the frequency mode (full or half frequency mode or down-clocking). Figure 9 shows the states and transitions of the state machine.

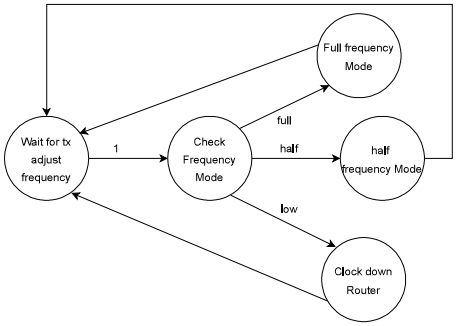

Fig. 9. State machine of Frequency Controller

The state machine transits to the next state at the rising edge of the trigger signal from the NI and the frequency operation mode is checked. In full frequency mode, the state machine configures the router to operate at full frequency. In half frequency mode, the state machine configures the router to operate at half frequency mode. On the other hand, in low frequency mode, the state machine clock gated the frequency of router. After configuring the router's frequency, the state machine waits again for the rising edge of the trigger signal to adjust the router's frequency.

IV. Result and Discussion

The architecture presented in the system model was implemented on an FPGA to evaluate the functional behavior of the proposed architecture and the resource utilization on the FPGA, and to measure the packet latency of the TTNoC with frequency scaling compared to the baseline LISNoC. In addition, an ORION 3.0 [4] tool is used to estimate the power consumption of routers. This is an open-source tool that can be used to estimate the power consumption of a micro-architecture, implementation, and operational parameters, as well as multiple routers Register transfer level (RTL).

A. Zynq Prototype

The TTNoC architecture with low-power techniques was instantiated for prototyping on a Xilinx Zynq UltraScale+ ZCU102 evaluation board. The ZCU 102 consists of processing systems (PS) and programmable logic (PL). The PL is also referred to as FPGA, and the routers, TTNIs, and soft processors are deployed in the PL. The extended LISNoC described in section III-A served as the basis for implementing the TTNoC. In addition, we use three soft processors, known as MicroBlaze, implemented in the PL of the ZCU 102, and a single hard processor (Arm Cortex-M1 processor) from the PS. The resource consumption of a 2X2 TTNoC equipped with power-saving techniques is shown in Table IV-A-1.

Hardware	LUT	Register (FF)
MicroBlaze x3	3250	2767
LISNoC x4	21920	14500
TTNI x4	52328	46604
Frequency Controller x4	12	16
GTB	2	57

TABLE I
RESOURCE UTILIZATION OF 2X2 TTNOC

B. Performance Analysis of Proposed Architecture

The simulation results from ModelSim were analyzed to evaluate the performance of the proposed TTNoC architecture equipped with power-saving techniques and the baseline LISNoC. The performance metric used to analyze the NoC is latency. We are transferring two messages from core 0 and core 1 that share resources as depicted in Figure 10 and evaluate the average latency of both messages from core 0 and core 1. Table IV-B-2 indicates the average latency result of the LISNoC, operating with 200 MHz, and table IV-B-3 indicates the average latency of TTNoC with frequency scaling. In this experiment, the TTNoC is assumed to be operated at a full frequency (200 MHz) when the messages pass through the router, and clock gating is used when no messages pass through the router.

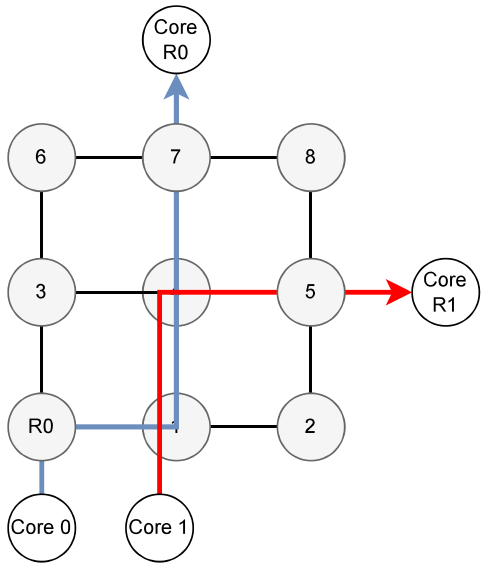

Fig. 10. 3x3 LISNoC and TTNoC paths

Packet Size	LISNoC Average Latency (ns)	TTNoC Average Latency (ns)
2	785	550
4	1850	1100
8	3600	1870
16	10605	4210

TABLE II
PACKET SIZE VS LATENCY OF CORE 0

Packet Size	LISNoC Average Latency (ns)	TTNoC Average Latency (ns)
2	180	125
4	340	249
8	840	450
16	2100	875

TABLE III
PACKET SIZE VS LATENCY OF CORE 1

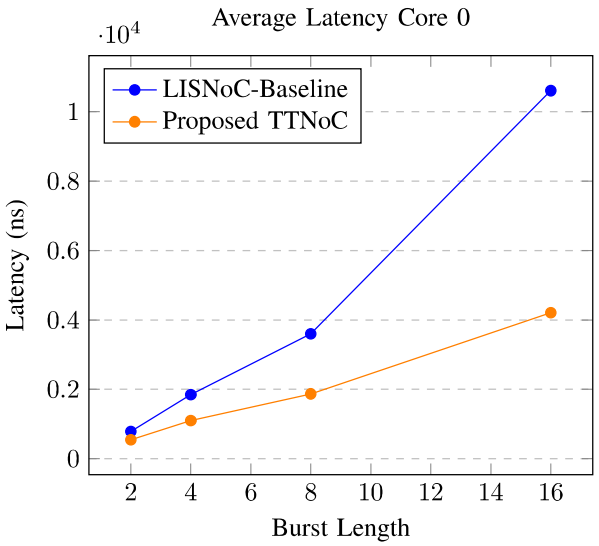

Fig. 11. Plot of Average Latency against Burst Lenght

Graphs were plotted for both cores based on the values given in the table IV-B-2 and IV-B-3. Figure 11 represents the average latency of core 0 with different packet sizes. Similarly, graph 12 illustrates the average latency of core 1 with different packet sizes. The curve between latency and packet size for TTNoC looks roughly straight-line shaped for both cores; however, in the case of LISNoC, the latency increases abruptly as the packet size increases from 2 to 16 in both cores. As a result, the proposed TTNoC performs better than the baseline LISNoC when both are operated with the same clock speed.

C. Power estimation of TTNoC with frequency scaling vs LISNoC

As described in the system model, dynamic frequency scaling is used at the router level of the NoC to optimize the power consumption of the multi-core on the chip. We consider three cases to evaluate the power-saving techniques used at the router level of TTNoC. The first case is a TTNoC without frequency scaling (see Figure 13). The second case is a TTNoC with power-saving techniques, where the frequency of active routers is always operated in full-frequency mode, and the inactive routers are clock gated (see Figure 14). The third case (see Figure 15) is a TTNoC with multiple clock domains, such as full-frequency and half-frequency modes. The scheduler operates the active router in a full-frequency mode for communication that requires high bandwidth and in a half-frequency mode for communication that requires lower

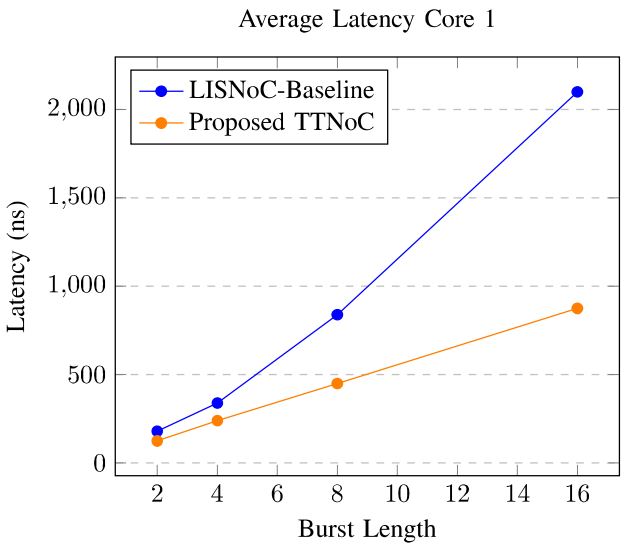

Fig. 12. Plot of Average Latency against Burst Length

bandwidth. Frequency selection is made by the frequency controller depending on the schedule. Half-frequency mode is typically used to optimize power consumption in the router that needs to transmit a low-criticality message. Also, inactive routers are always clock gated to save more power.

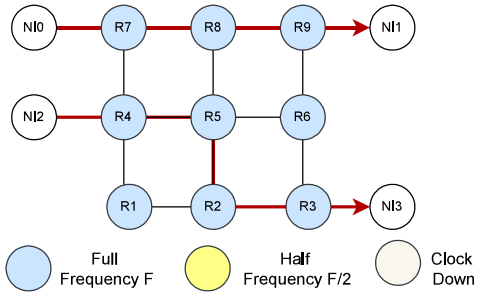

Fig. 13. Example of TTNoC without frequency scaling in the router

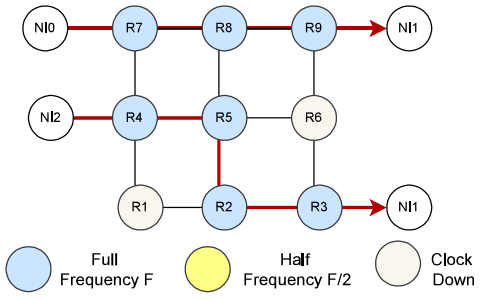

Fig. 14. Example of TTNoC with frequency scaling

Both cases, 2 and 3, can optimize the power consumption of the NoC router by clock gating the inactive routers. When the second case is used, the frequency of the active routers is always operated at full frequency, and the slack in the low critical messages may be large. However, by using multiple clock domains during the active time of the router, the frequency of the routers can be operated at half frequency instead of full frequency, which may reduce the slack time of messages and optimizes the power consumption of routers, as illustrated in Figure 16 and 17.

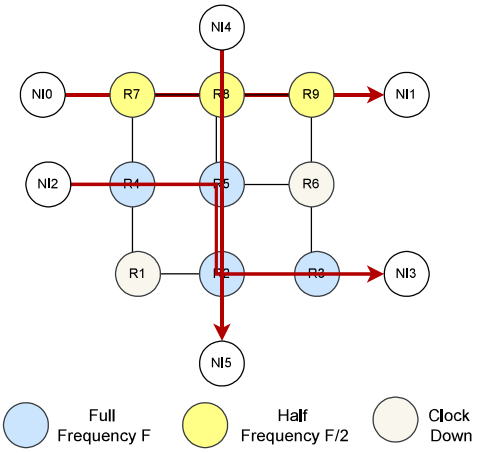

Fig. 15. Example of TTNoC with frequency scaling with multiple clock domain

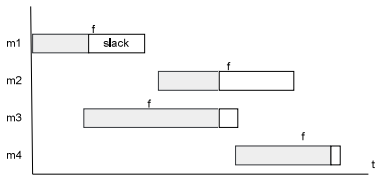

Fig. 16. Example of messages within the NoC: Message 1, and 2 have large slack time

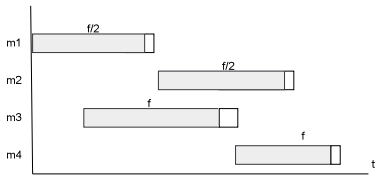

Fig. 17. Example of messages within the NoC: We can see a four messages . The scheduler adjust the frequency of m1, and m2 with half frequency mode

Table IV-C-4, IV-C-5, and IV-C-6 summarize the energy consumption of a 3x3 TTNoC with frequency scaling and without frequency scaling with different active times and different messages. Table IV-C-4 depicts the energy consumption of the TTNoC without frequency scaling, which means that the operating frequencies of routers are always operated at full frequency mode for idle and active routers.

Table IV-C-5 shows the energy consumption of TTNoC with frequency scaling. In this case, the frequency of the active

Message ID	Active Router	Idle Router	Frequency Used	Deadlines (micro s)	Energy Active (nJ)	Energy Idle (nJ)
1	6	3	full	520	77380	38690
2	5	4	full	600	59523	74404
3	6	3	full	100	14880	7440
4	2	7	full	120	5952	20833

TABLE IV
CASE-1 TTNOC WITHOUT FREQUENCY SCALING

routers is always operated at full frequency and the idle routers are clocked down.

Message ID	Active Router	Idle Router	Frequency Used	Deadlines (micro s)	Energy Active (nJ)	Energy Idle (nJ)
1	6	3	full	520	77380	920
2	5	4	full	600	74404	1416
3	6	3	full	100	14880	177
4	2	7	full	120	5952	495

TABLE V
CASE-2 TTNOC WITH FREQUENCY SCALING

Table IV-C-6 shows the energy consumption of TTNoC with frequency scaling. In this case, the active routers can operate at full or half frequency depending on the schedule, and the idle routers are clocked down. The following figure shows

Message ID	Active Router	Idle Router	Freq Used	Deadline (micro s)	Energy Active (nJ)	Energy Idle (nJ)
1	6	3	half	520	76413	920
2	5	4	half	600	73474	1416
3	6	3	full	100	14880	177
4	2	7	full	120	5952	6447

TABLE VI
CASE-3 TTNOC WITH FREQUENCY SCALING USING FULL AND HALF FREQUENCY MODE

the comparison of the power consumption of the 3x3 TTNoC based on the three cases define before.

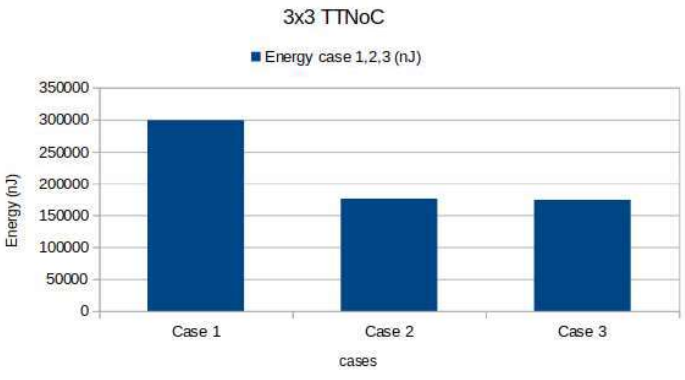

Fig. 18. Average Energy consumed by TTNoC as defined in 3 cases

As seen in Figure 18, the energy consumption of a 3x3 TTNoC without frequency scaling is 299107 nJ. However, scaling the frequency of TTNoC, as described in Case 2, optimizes the energy consumption of the TTNoC router as the energy consumption decreases to 175628 nJ. The scaling of TTNoC frequency with multiple clock domains described in Case-3 reduces the energy consumption of TTNoC to about 173730 nJ. Therefore, Case 3 is the best method for scaling the TTNoC frequency since it has the lowest energy consumption compared to Case 1 and Case 2.

V. CONCLUSION

TTNoC-based frequency scaling supports time-triggered communication and frequency scaling at the router level according to a predefined schedule. The results showed that scaling the router's frequency in both cases 2 and 3 described in section IV-B is beneficial due to the reduction of the energy consumption in the router while maintaining the performance of NoC. However, scaling the frequency of routers using multiple clock domains is the most suitable technique to optimize the energy consumption of TTNoC since the energy saving, in this case, is the lowest. In the future, we plan to work on low-power techniques of NoC to optimize the static power of NoC, which contributes more to the overall chip power when the size of the transistor is scaled down.

ACKNOWLEDGMENT

This work has received funding from the ECSEL Joint Undertaking (JU) under Grant Agreement No. 877056. The JU receives support from the European Union's Horizon 2020 research and innovation program and from Spain, Italy, Austria, Germany, France, Finland, and Switzerland.

REFERENCES

[1] Xi Chen, Zheng Xu, Hyungjun Kim, Paul V. Gratz, Jiang Hu, Michael Kishinevsky, Umit Ogras, and Raid Ayoub. Dynamic voltage and frequency scaling for shared resources in multicore processor designs. In *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–7, 2013.

[2] Radu David, Paul Bogdan, Radu Marculescu, and Umit Ogras. Dynamic power management of voltage-frequency island partitioned networks-on-chip using intel's single-chip cloud computer. In *Proceedings of the Fifth ACM/IEEE International Symposium*, pages 257–258, 2011.

[3] Yatin Hoskote, Sriram Vangal, Arvind Singh, Nitin Borkar, and Shekhar Borkar. A 5-ghz mesh interconnect for a teraflops processor. *IEEE Micro*, 27(5):51–61, 2007.

[4] Andrew B. Kahng, Bill Lin, and Siddhartha Nath. Orion3.0: A comprehensive noc router estimation tool. *IEEE Embedded Systems Letters*, 7(2):41–45, 2015.

[5] Timothy G Mattson, Rob F Van der Wijngaart, Michael Riepen, Thomas Lehnig, Paul Brett, Werner Haas, Patrick Kennedy, Jason Howard, Sriram Vangal, Nitin Borkar, Greg Ruhl, and Saurabh Dighe. The 48-core scc processor: the programmer's view. In *SC '10: Proceedings of the 2010 ACM/IEEE International Conference for*

High Performance Computing, Networking, Storage and Analysis, pages 1–11, 2010.

[6] Asit K. Mishra, Reetuparna Das, Soumya Eachempati, Ravi Iyer, N. Vijaykrishnan, and Chita R. Das. A case for dynamic frequency tuning in on-chip networks. In *2009 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 292–303, 2009.

[7] Rakotojaona Nambinina, Daniel Onwuchekwa, Hamidreza Ahmadian, Dinesh Goyal, and Roman Obermaisser. Time-triggered frequency scaling in network-on-chip for safety-relevant embedded systems. In *2021 International Conference on Smart Generation Computing, Communication and Networking (SMART GENCON)*, pages 1–7, 2021.

[8] Rakotojaona Nambinina, Daniel Onwuchekwa, Sabikun Nahar, Darshak Sheladiya, and Roman Obermaisser. Extension of the lisnoc (network -on-chip) with an axi-based network interface. In *2022 6th International Conference on Computing Methodologies and Communication (ICCMC)*, pages 682–686, 2022.

[9] K. Niyogi and D. Marculescu. Speed and voltage selection for gals systems based on voltage/frequency islands. In *Proceedings of the ASP-DAC 2005. Asia and South Pacific Design Automation Conference, 2005.*, volume 1, pages 292–297 Vol. 1, 2005.

[10] Roman Obermaisser, Hamidreza Ahmadian, Adele Maleki, Yosab Bebawy, Alina Lenz, and Babak Sorkhpour. Adaptive time-triggered multi-core architecture. *Designs*, 3(1), 2019.

[11] Jens Rettkowski and Diana Göhringer. Wormhole computing in networks-on-chip. In *2021 31st International Conference on Field-Programmable Logic and Applications (FPL)*, pages 273–274, 2021.

[12] Martin Schoeberl. A time-triggered network-on-chip. In *2007 International Conference on Field Programmable Logic and Applications*, pages 377–382, 2007.

[13] Vitor R. G. Silva, Alex F. A. Furtunato, Kyriakos Georgiou, Carlos A. V. Sakuyama, Kerstin Eder, and Samuel Xavier-de Souza. Energy-optimal configurations for single-node hpc applications. In *2019 International Conference on High Performance Computing & Simulation (HPCS)*, pages 448–454, 2019.

[14] M.B. Taylor, J. Kim, J. Miller, D. Wentzlaff, F. Ghodrat, B. Greenwald, H. Hoffman, P. Johnson, Jae-Wook Lee, W. Lee, A. Ma, A. Saraf, M. Seneski, N. Shnidman, V. Strumpen, M. Frank, S. Amarasinghe, and A. Agarwal. The raw microprocessor: a computational fabric for software circuits and general-purpose programs. *IEEE Micro*, 22(2):25–35, 2002.

[15] TUM. Lisnoc. https://www.ce.cit.tum.de/lis/forschung/aktuelle-projekte/optimsoc/lis-noc/. Accessed 2022-14-09.

[16] Jia Zhan, Nikolay Stoimenov, Jin Ouyang, Lothar Thiele, Vijaykrishnan Narayanan, and Yuan Xie. 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, 2014.

[17] Pingqiang Zhou, Jieming Yin, Antonia Zhai, and Sachin S. Sapatnekar. Noc frequency scaling with flexible-pipeline routers. In *IEEE/ACM International Symposium on Low Power Electronics and Design*, pages 403–408, 2011.