Time-triggered Network Interface Extension for the Versal Network-on-Chip

Daniel Onwuchekwa, Josepaul Paulachan, Rakotojaona Nambinina, Roman Obermaisser
Chair for Embedded Systems
University of Siegen
57076 Siegen, Germany

***Abstract*—In semiconductor technology, the ability to scale up the microchip made it possible to integrate more IP blocks, microprocessors, and other components into a single die known as a System-on-Chip. Network-on-Chips are introduced to enable efficient communication between the components integrated into the System-on-Chip. However, they start to suffer from data loss and a higher jitter as the number of communicating entities increase, and communicate simultaneously. Hence establishing temporal partitions between various subsystems has become a crucial requirement. The recent advancements in Network-on-Chips have led to the development of the Versal NoC, by Xilinx. The Versal Network-on-Chip provides different Quality of Service for efficient communication; However, temporal partitioning is not sufficiently covered for messages injected into the NoC.**

This work develops a Time-Triggered Extension Layer for the Versal Network-on-Chip to provide temporal guarantees for messages that traverse across the NoC. The outcome applies temporal partitioning to avoid the collision of messages, thereby providing determinism for the Network-on-Chip. The results of the evaluation show reduced jitter when the developed time-triggered extension layer is incorporated into the Versal Network-on-Chip. It also safeguards the Network-on-Chip against data loss occurring when two or more PEs (Processing Elements) attempt to access the Network-on-Chip simultaneously. The implication of this work is that the Time-Triggered Extension Layer provides determinism for messages injected into the Network-on-Chip.

***Index Terms*—Network-on-Chip, Time-Triggered Systems, Latency, Jitter, Versal NoC**

I. INTRODUCTION

The application scope of embedded devices has continued to grow over the past decades. We utilize these embedded devices daily, which have become an integral part of our lives. However, the complexity of these devices is increasing due to recent developments in chip fabrication technology. One such embedded platform whose complexity is growing daily as additional features have been installed is the System on Chip (SoC) [1]. Due to this increase in complexity from integrating various components in an SoCs, the standard bus communication started to face issues with power dissipation and arbitration. Therefore, Network-on-Chips (NoCs) were introduced to solve these communication issues as they provide scalability and power efficiency.

The Xilinx Adaptive Compute Acceleration Platform (ACAP) has a NoC [2]. The NoC provides features that can be configured at run-time, which many of the previous NoCs lacked. However, the Versal NoC does not provide temporal partitioning of messages. Since Versal NoC is employed in MCSs (Mixed Criticality Systems), it is essential to have temporal predictability for safety-critical applications.

The lack of temporal guarantees in NoCs can result in scenarios where data is lost, especially when two processing elements (PE) attempt to simultaneously access an egress port of the NoC switches. Even with mechanisms to avoid packet loss could also result in increased jitter as one or more messages must be delayed in the event of collision.

Current Versal ACAP devices such as the Xilinx Versal series support three classes of messages: low latency, isochronous, and best-effort traffic. The low latency is used for high-priority transactions usually associated with cache fill and replacement. The isochronous traffic is used for applications that can tolerate a longer latency such as video class transactions but it is still capped at a maximum latency that will not cause system degradation. The best-effort traffic supports high-throughput and best-effort transactions that are allowed with long latency but need high throughput to achieve performance goals.

At the moment, there is no traffic class which provides determinism for the message transmission in Versal NoC; time-triggered (TT) traffic is not yet supported. TT traffic uses a dynamic and static scheduler to send out messages on a conflict-free periods. The scheduler ensures that no conflict occurs between the TT traffic. This work adapts a soft Intellectual Property (IP) called the Time-Triggered Extension Layer (TTEL) to the Versal NoC to provide temporal predictability. The added layer supports safety through temporal partitioning and supports Mixed-criticality systems (MCSs). Temporal partitioning is accomplished by triggering the transmission of safety-critical messages at pre-defined instants specified by a schedule following the time-triggered paradigm. This work contributes as follows.

- It Provides temporal predictability property to the Versal NoC by adapting the TTEL in [3] to the Versal NoC.
- It shows the impact of the temporal predictability property by performing experiments that compare the Versal NoC without the TTEL to the Versal NoC with the TTEL.

The paper is structured as follows: Section II discusses the related works. Section III discusses an overall Network-on-Chip concept followed by the Versal NoC. The proposed architecture along with different building blocks of TTEL is discussed in section IV. Section V put forward the experimental setup to evaluate the architecture. Section VI covers the

results of the experiments. This paper is then concluded in section VII.

II. RELATED WORK

The related work on NoC is discussed in this part, along with the contribution of the suggested design in comparison to earlier approaches.

An open-sourced NoC known as the LISNoC is introduced in [4]. It supports strict ordering, wormhole routing, virtual channels, configurable router setup, round-robin arbitration for link multiplexing, and virtual channels. Three flow control signals—flit, valid, and ready are the basis of the LISNoC's operation. However, it lacks a standardized interface such as AXI interface and making it difficult to interface with modern processing elements in hardware.

The LISNoC [4] is extended in [5] with source based routing and AXI interfaces. However, it does not support deterministic communication.

The AEthereal [6] NoC offers guaranteed and best-effort services. With the help of time-division multiplexed circuit switching, it allots a specified amount of bandwidth to the guaranteed services while leaving the rest open for the best-effort services. However, the transmission of rate-constrained messages is not supported by this design. The distributed routing and the assignment of specific slots for the guaranteed services also require a complex router architecture.

The QNoC is introduced in [7]. Wormhole packet routing is supported by the QoS network architecture. Due to the significant expense of creating and maintaining circuit connections, circuit switching is also avoided in the architecture. The use of store-and-forward routing techniques is also discouraged since they could result in significant buffer needs and, as a result, silicon area penalties. The routers in the architecture are connected by point-to-point links. The QNoC uses an irregular mesh topology and each modules in the NoC is connected using a standard interface.

Virtual circuits are discussed in Nostrum NoC [8], which also provides bandwidth guarantees and latency limits. Prioritized, looping containers are used on a path that is predetermined at design time to obtain guaranteed bandwidth. However, Nostrum employs deflective routing, which cannot ensure the necessary deterministic behaviour. Additionally, it uses a mesh architecture and routing decisions are determined locally in the switches.

The AxNoC [9] architecture is a dual-voltage NoC. A wormhole router with five physical channels that supports the X-Y routing method is provided by AxNoC. The router must begin its pre-packet actions one clock cycle before the packet arrives at the router because it employs the look-ahead wake-up mechanism to determine which input and output channels in the router are used for the incoming packets. Although the AxNoC offers the NoC high power efficiency, it offers no recommendations for time-triggered message transfer.

The TTNoC architecture is described in [10]. The TTNoC is an extended NoC that is used to achieve Time-triggered communication specifically, as a unique system architecture for SoCs. Furthermore, it provides built-in fault isolation to simplify the smooth integration of separately produced components, possibly with varying degrees of criticality. The implementation also offers fault-handling through reconfiguration, fault tolerance, a power-aware system behaviour, and techniques for integrated resource management supporting dynamically changing resource requirements. Although the TTNoC is appropriate for safety-critical applications, the rate-constrained and best-effort message transmissions are not supported.

The Network-on-Chip (NoC) using Xilinx's newest Versal^{TM} architecture is described in research papers [2] [11]. The next-generation 7nm architecture chips from Xilinx contain a NoC. These devices are part of the Adaptable Computing Acceleration Platform (ACAP) devices. The NoC unifies communication between the accelerator functions, FPGA fabric, memory subsystem, and processor system.

The Versal NoC uses an irregular topology. Versal NoC topologies are built from reusable building components that can link in various ways for various devices. Versal NoC distinguishes itself from other NoCs by allowing the user to select the QoS at the design stage. The Versal NoC communicates with devices via the AXI standard. Additionally, AXI stream support is also included but it does not support time-triggered communication.

The TTEL [3] supports all the traffic class Time-Triggered(TT), Rate Constrained (RC), and Best-Effort(BE). The Versal NoC lacks the TT traffic class. Thus, Our proposed architecture adds TTEL to the underlying Versal NoC to have this feature. The design is tested on Versal NoC. The proposed design also allows end users to select the QoS according to their needs. The recommended approach combines the advantages of TTNoC with the Versal NoC on Versal ACAP devices to offer determinism for the MCSs.

III. NETWORK ON CHIP

A. Network On Chip

Network-on-Chip (NoC) is an on-chip communication system that is built in an integrated circuit such as a System-on-Chip. It is made up of several components such as Processing Elements (PEs), routers and links [12]. The Network Interfaces (NI) is an interface between PE and the routers that allow the communication between PEs (cores, gateways, memory, I/O). These PEs send and receive packets over a network of routers that are linked together by the links or channels. Links are typically made up of two unidirectional, point to point buses that run in opposite directions.

In NoC, data packet transmission takes place between sender PEs and receiver (PE). Depending on the router's decision, data packet transmission is forwarded on the network in a sequence of steps. The router may have a buffer, or it may be buffer-less, and packets are first received and stored in the input buffer in case of buffered routers. The packets are subdivided into flits. Flits are temporarily stored in input/output buffers. The output port allocator selects the output port for each flit. The physical connection is made by

the router via a link from the input port to the output port and the control logic is in charge of overall synchronization.

B. Versal Network on Chip

The Xilinx Versal programmable NoC is an AXI-based interconnect used for data exchange between IP e.g., processors, DDR, AI Engine, and Custom IPs. This infrastructure is a high-speed, integrated data path with dedicated switching. The Versal NoC [2] is comprised of a series of interconnected Horizontal (HNoC) and Vertical NoC (VNoC) paths and is supported by a set of configurable, hardware-implemented components that can be configured in various ways to meet design timing, speed, and logic utilization requirements. The HNoC and VNoC are dedicated, high-bandwidth paths that connect integrated blocks between the processing system and the programmable logic (PL).

The NoC components comprise NoC master units (NMU), NoC slave units (NSU), NoC packet switches (NPS), and the NoC Inter-Die-Bridge (NIDB). The NoC master unit (NMU) is the ingress point to the NoC and it is used to connect a master to the NoC. The egress point to the NoC is the NoC slave unit (NSU) which is used to connect a slave to the NoC. The NIDB connects two super logic regions (SLRs) together, providing high bandwidth between dies. The NPS is the crossbar switch, used to route the packets within the NoC. It connects the NMU and NSU interfaces.

The NoC compiler, a component of the Vivado Design Suite, determines the routing paths between ingress and egress points. It is also responsible for the assignment of NoC ingress and egress points to particular NMUs and NSUs at design time.

IV. TIME-TRIGGERED EXTENSION LAYER

The Time-Triggered Extension Layer (TTEL) [3] is a temporal and spatial partitioning layer that is established on top of the NoC. The TT extension layer allows the underlying NoC to support several types of communication (e.g., Time-Triggered (TT), Rate-Constrained (RC), and Best-Effort (BE)). The architecture of TTEL is composed of three building blocks; ports, scheduler and dispatcher. The ports achieve the spatial partitioning while the scheduler establishes the temporal partitioning and efficient resource utilization. Figure 1 show the building blocks of the TTEL. The arrows in the figure shows the direction of data movement within the TTEL.

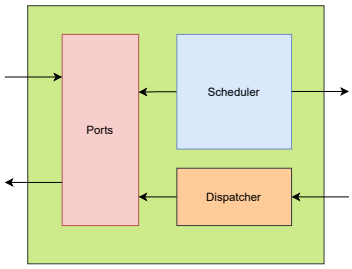

Fig. 1. Building Blocks of TTEL

A. Ports

The Ports serve as a link between the cores and the extension layer's building blocks. Ports are separated from one another by spatially separated memory areas, resulting in spatial partitioning at the transport level. In case of input ports, the core can write the message to the data area, and the scheduler then dequeues it based on the temporal conditions. The dispatcher enqueues the message to be fetched by the corresponding core once it is received at the destination.

B. Scheduler

The scheduler is responsible for most of the TTEL's functionality. It ensures that messages sent by safety-critical subsystem are not interfered with. By enabling communication forms in conjunction with priorities, the scheduler establishes safety and efficient resource management. The scheduler enforces the allocated time-slots for the different traffic classes TT, RC, and BE.

The scheduler's overall functionality can be described as a series of dequeue-process-enqueue steps. The detailed specification of each action is determined based on the preset parameters accessible at the dequeued port. The operations listed above can be classified into the following categories:

1) Periodic Transmission of the TT Messages: The time-triggered communication paradigm establishes a deterministic communication system for safety-critical subsystems by providing a priori knowledge about the instant at which the message arrives at routers and the destination. In both temporal and spatial aspects, the actions are carried out according to a preset configuration. The behaviour of the schedulers of different extension layers must be harmonized in order to achieve chip-wide collision-free communication of time-triggered messages. In this paper the TT traffic class is only considered as the Versal NoC doesnt have this feature.

2) Traffic Shaping of the RC Messages: The traffic shaping function specified in ARINC Specification 664P7 [13] influenced the capabilities of the scheduler for rate-constrained communications. The traffic shaper's goal is to control bandwidth based on the MINT specified in the configuration. If the Minimum Inter-Arrival Times (MINT) has not elapsed, the traffic shaper cannot inject consecutive rate-constrained messages into the connection. Furthermore, enough bandwidth must be supplied to ensure that delays and temporal irregularities (jitter) are kept within acceptable limits.

3) Relaying of BE Messages: Within the system, best-effort messages have the lowest priority. Only unused bandwidth by rate-constrained messages is used by the scheduler to inject such messages into the connection. Individual best-effort messages can have their own destination address, unlike time-triggered and rate-constrained messages, where the destination of all messages of each port is determined at design time by the parameters. As a result, in the case of best-effort messages, the scheduler uses the application layer to decode the inserted destination and sends the routing information to the NI.

C. Dispatcher

The dispatcher extracts the message's destination port and queues it in the appropriate port. The processor then reads the message from the respective ports.

D. Architecture of TTEL with the Versal NoC

The proposed NoC architecture is illustrated using figure 2. It consists of several tiles that are linked to the Versal NoC. These tiles can serve as a source or sink in message-based communication. A bare-metal application or an operating system can be run on each tile, which may have one or more processor cores.

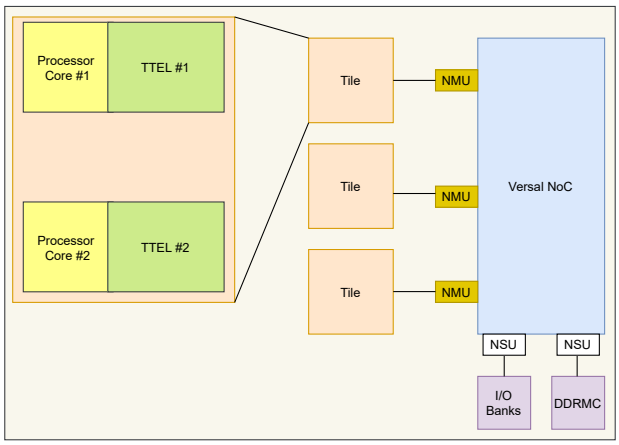

Fig. 2. Structure of the Architecture

The position of the TTEL with respect to the other NoC components is depicted in Figure 2. The TTEL handles the scheduling of messages sent by the cores and, routes messages received from the NoC to the appropriate ports so that the processor can read them.

V. EXPERIMENTS AND RESULTS

The aim of the experiments is to evaluate the impact of the TTEL on the Versal NoC. We compare the jitter of the Versal NoC without the extension (TTEL) and with a setup that includes the TTEL. We explored four different scenarios, two scenarios for isochronous traffic, and two scenarios for best-effort traffic. We investigated for each scenario, the behavior when the read-write bandwidth is set to 100MB/s and 50MB/s. Table I summarizes the configurations. It also shows different burst sizes that were observed.

The metric used to analyze the addition of the TTEL to the Versal NoC is jitter. We first compute the latency of all messages transmitted over the NoC. The jitter is the minimum latency subtracted from the maximum latency.

Figure 3, shows the experimental setup, and it is obtained after running the validation of the experiment design from the Vivado design suite. The figure is imposed with the corresponding labels Core 0, Core 1 for cores, R1-R4 for NPS and DDRMC for Double Data Rate Memory Controller for clarity. The justification of the setup is to evaluate the impact of the TTEL in a resource sharing scenario. From figure 3, it can be seen that the data path from both cores shares the egress port of NPS (R2).

TABLE I
EXPERIMENTAL SETUP

Exp No.	Sub No.	Traffic Class	Read Write Bandwidth (MB/s)	Burst Size
1	1.1	Isochronous	100	2
	1.2	Isochronous	100	4
	1.3	Isochronous	100	8
	1.4	Isochronous	100	16
2	2.1	Best-Effort	100	2
	2.2	Best-Effort	100	4
	2.3	Best-Effort	100	8
	2.4	Best-Effort	100	16
3	3.1	Isochronous	50	2
	3.2	Isochronous	50	4
	3.3	Isochronous	50	8
	3.4	Isochronous	50	16
4	4.1	Best-Effort	50	2
	4.2	Best-Effort	50	4
	4.3	Best-Effort	50	8
	4.4	Best-Effort	50	16

In the case of TTEL with Versal NoC the jitter observed is smaller than the jitter observed without the TTEL. The graphs 4 - 11 are plotted for different processing elements against burst length for different QoS (isochronous traffic 100MB/s and 50MB/s, best-effort 100MB/s and 50MB/s), indicates that the architecture with TTEL out-performs the architecture without TTEL.

A. Isochronous Traffic with 100MB/s

Table II and table III indicate the performance analysis of traffic class isochronous with 100MB/s for Core 0 and Core 1 with TTEL and without TTEL, respectively. From table II it can be seen that for Core 0, maximum latency for the design with TTEL is 216.456 μs with increasing burst length. In contrast, the maximum latency increased with a higher burst length in case of the design without TTEL. When observing the performance parameter jitter, for both the design and for the Core 0, the jitter increases with a higher burst length. The maximum jitter recorded is 139.063 μs in the case of the design with TTEL, while it is 2282.813 μs in the design without TTEL.

TABLE II
PERFORMANCE ANALYSIS OF CORE 0; ISOCHRONOUS (100MB/S); WITHOUT TTEL AND WITH TTEL

Core 0						
Without TTEL				With TTEL		
Burst Length	Max Latency (μs)	Min Latency (μs)	Jitter (μs)	Max Latency (μs)	Min Latency (μs)	Jitter (μs)
2	369.581	1.456	368.125	216.456	205.518	10.938
4	407.081	1.143	405.935	216.456	185.831	30.625
8	433.956	0.206	433.750	216.456	146.456	70
16	2286.456	3.643	2282.813	216.456	77.393	139.063

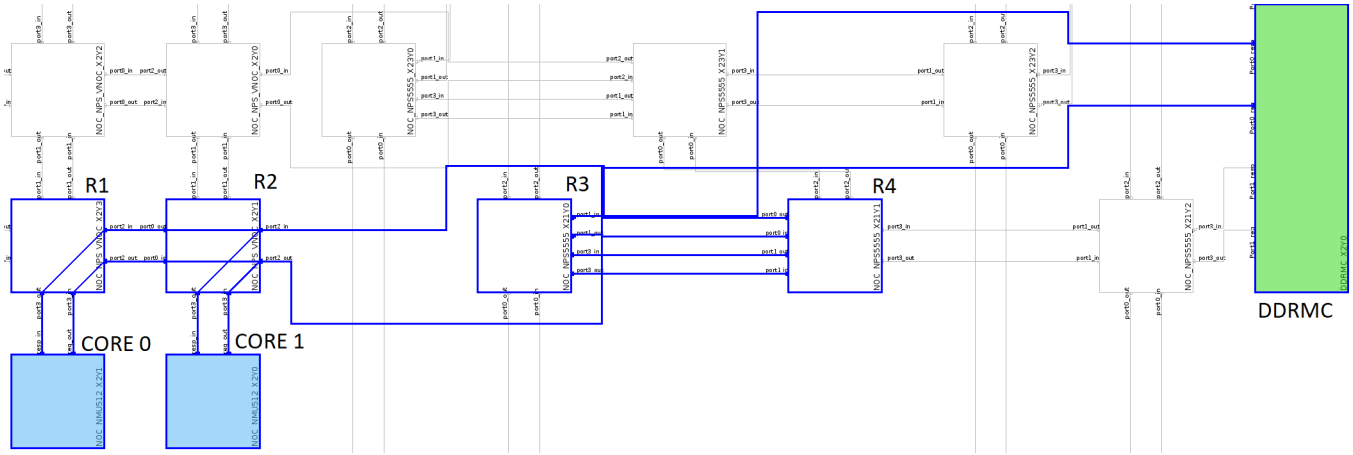

Fig. 3. NoC QoS path for the Experimental Setup

The performance analysis of Core 1 shows a similar trend as Core 0. The design without TTEL has an increasing maximum latency depending on the burst length. For the design with TTEL, the maximum latency was maintained at 232.706 μs, irrespective of the increase in burst length. For both designs, the jitter increased with an increase in burst length. The maximum jitter for the design with TTEL was 171.563 μs, and 2093.750 μs for the design without TTEL.

TABLE III
PERFORMANCE ANALYSIS OF CORE 1; ISOCHRONOUS (100MB/S); WITHOUT TTEL AND WITH TTEL

Core 1						
Without TTEL				With TTEL		
Burst Length	Max Latency (μs)	Min Latency (μs)	Jitter (μs)	Max Latency (μs)	Min Latency (μs)	Jitter (μs)
2	48.018	6.456	41.562	232.706	189.268	43.438
4	67.081	6.456	60.625	232.706	169.581	63.125
8	428.956	0.206	428.750	232.706	130.206	102.5
16	2094.893	1.143	2093.75	232.706	61.143	171.563

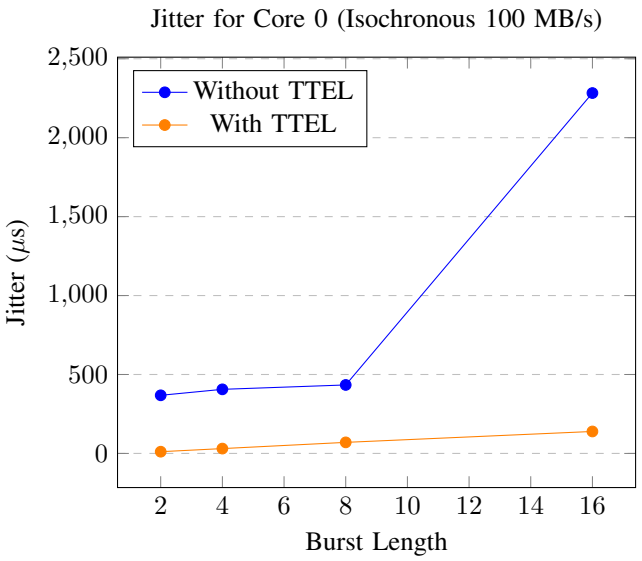

Fig. 4. Plot of Jitter against Burst Length

The graphs for jitter in each processor with and without TTEL are plotted in Figure 4 and 5 respectively. The graphs illustrates the jitter in Core 0 and Core 1 for the traffic class isochronous with the read-write bandwidth of 100MB/s.

From the two graphs, one could infer that the jitter increases for both designs as the burst length increases. However, an increase in the jitter value against burst length is more prominent in the case of the design without TTEL than the design with TTEL. In the case of the design with TTEL, the curve appears to have an approximately straight line nature for both the cores, while in the design without TTEL, the jitter results in a sudden increase as the burst length changes from a size of 8 to 16 in the case of Core 0 and from 4 to 16 in case of Core 1. This sudden increase in jitter in the design without TTEL is because of the back-pressure [14] from the NoC as two PEs try to send data at the same instant. Back-pressure is a method that controls the information flow through a communication channel while making sure that no packet is lost.

B. Best-effort Traffic with 100MB/s

The performance analysis of traffic class best-effort with 100MB/s for Core 0 and Core 1 with TTEL and without TTEL is illustrated in the tables IV and V. The Core 0 with TTEL had a maximum latency of 213.956 μs as the burst length increased. Similarly, for Core 1 with TTEL, the maximum latency was maintained at 232.706μs. However, the maximum latency showed an increasing trend with increasing burst length in the case of the design without TTEL for both the Cores 0 and 1. The maximum jitter in Core 0 is 137.813 μs in the case of the design with TTEL, while it is 2093.750 μs in the design without TTEL.

For both design scenarios in Core 1, the maximum jitter is 169.063 μs for the design with TTEL while it is 2282.813 μs

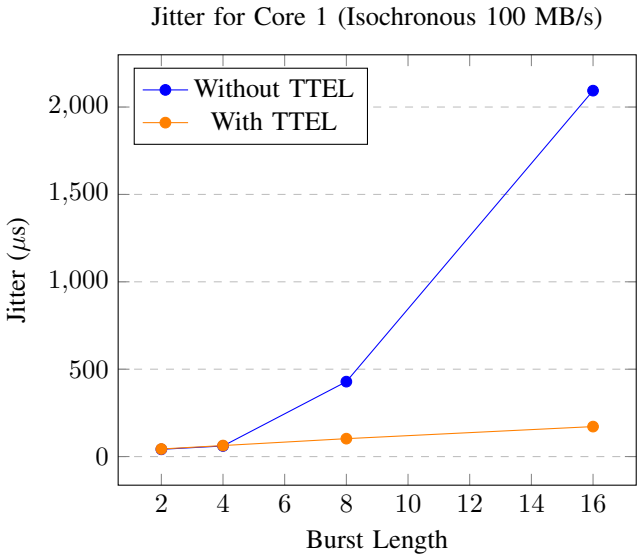

Fig. 5. Plot of Jitter against Burst Length

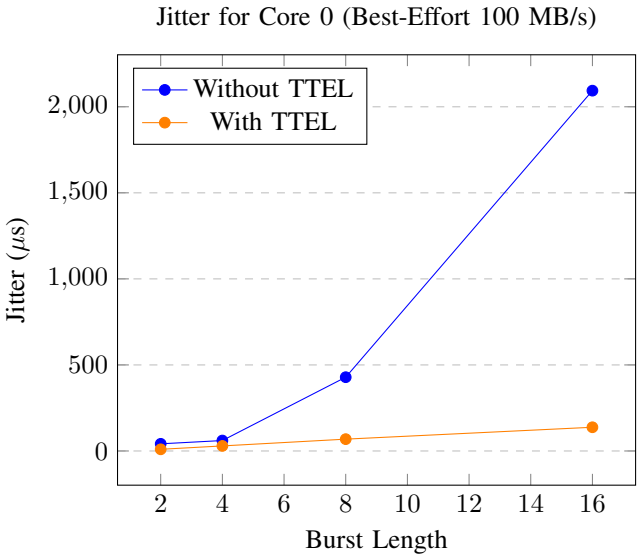

Fig. 6. Plot of Jitter against Burst Length

TABLE IV
PERFORMANCE ANALYSIS OF CORE 0; BEST-EFFORT (100MB/S); WITHOUT TTEL AND WITH TTEL

Core 0						
Without TTEL				With TTEL		
Burst Length	Max Latency (μs)	Min Latency (μs)	Jitter (μs)	Max Latency (μs)	Min Latency (μs)	Jitter (μs)
2	48.018	6.456	41.562	213.956	204.268	9.688
4	67.081	6.456	60.625	213.956	184.581	29.375
8	428.956	0.206	428.750	213.956	145.206	68.75
16	2094.893	1.143	2093.75	216.456	76.143	137.813

for the design without TTEL.

TABLE V
PERFORMANCE ANALYSIS OF CORE 1; BEST-EFFORT (100MB/S); WITHOUT TTEL AND WITH TTEL

Core 1						
Without TTEL				With TTEL		
Burst Length	Max Latency (μs)	Min Latency (μs)	Jitter (μs)	Max Latency (μs)	Min Latency (μs)	Jitter (μs)
2	369.581	1.456	368.125	232.706	191.768	40.938
4	442.081	1.143	440.938	232.706	172.081	60.625
8	433.956	0.206	433.750	232.706	132.706	100
16	2286.456	3.643	2282.813	232.706	63.643	169.063

Figures 6 and 7 illustrates the graphs of jitter against the burst length for the traffic class best-effort with 100MB/s. Graph 6 presents the variation of jitter in Core 0. Similarly, graph 7 illustrates the variation of jitter in Core 1. The curve between jitter and burst length for design with TTEL is straight-line shaped for both cores; however, in case of the design without TTEL, the jitter increases abruptly as the burst length increases from 2 to 16 in both the cores.

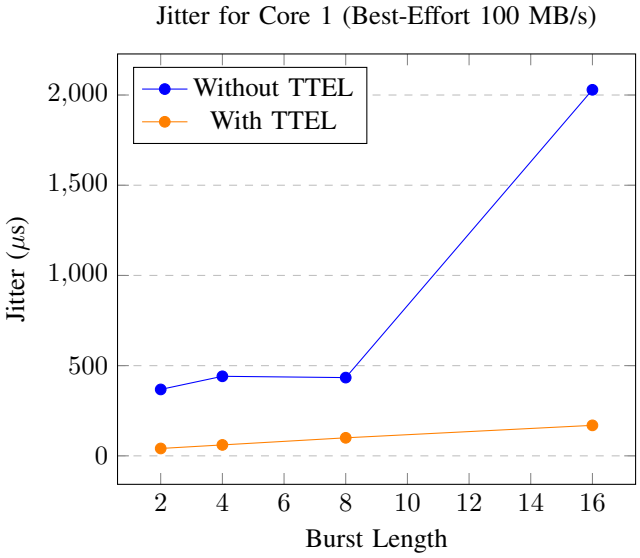

Fig. 7. Plot of Jitter against Burst Length

C. Isochronous Traffic with 50MB/s

Similarly, tables VI and VII shows the results for the 50MB/s configuration for traffic class isochronous. The graphs 8 and 9 show the jitter for the traffic class isochronous with a read-write bandwidth of 50MB/s in Cores 0 and 1. One could deduce from the two plots that as the burst length grows, jitter gets worse for both architectures. However, the design without TTEL exhibits a greater increase in the jitter value against burst length than the design with TTEL. For the design without TTEL, the jitter abruptly increases as the burst length goes from 2 to 16. The sudden increase in jitter is from the back-pressure exerted by the NoC.

TABLE VI
PERFORMANCE ANALYSIS OF CORE 0; ISOCHRONOUS (50MB/S); WITHOUT TTEL AND WITH TTEL

Core 0						
Without TTEL				With TTEL		
Burst Length	Max Latency (μs)	Min Latency (μs)	Jitter (μs)	Max Latency (μs)	Min Latency (μs)	Jitter (μs)
2	369.581	1.456	368.125	216.456	205.518	10.938
4	407.081	1.143	405.935	216.456	185.831	30.625
8	433.956	0.206	433.750	216.456	146.456	70
16	2286.456	3.643	2282.813	216.456	77.393	139.063

TABLE VII
PERFORMANCE ANALYSIS OF CORE 1; ISOCHRONOUS (50MB/S); WITHOUT TTEL AND WITH TTEL

Core 1						
Without TTEL				With TTEL		
Burst Length	Max Latency (μs)	Min Latency (μs)	Jitter (μs)	Max Latency (μs)	Min Latency (μs)	Jitter (μs)
2	48.018	6.456	41.562	232.706	189.268	43.438
4	67.081	6.456	60.625	232.706	169.581	63.125
8	428.956	0.206	428.750	232.706	130.206	102.5
16	2094.893	1.143	2093.75	232.706	61.143	171.563

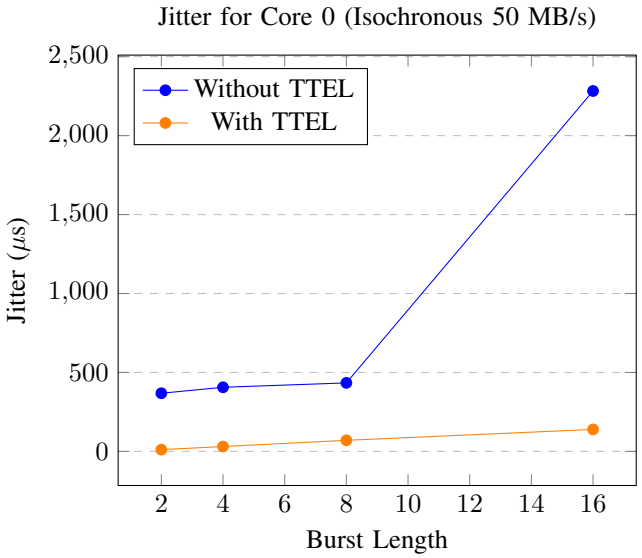

Fig. 8. Plot of Jitter against Burst Length

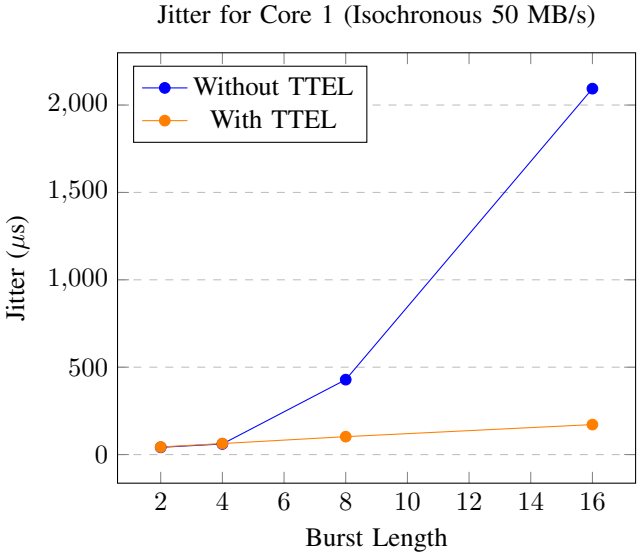

Fig. 9. Plot of Jitter against Burst Length

D. Best-Effort Traffic with 50MB/s

TABLE VIII
PERFORMANCE ANALYSIS OF CORE 0; BEST-EFFORT (50MB/S); WITHOUT TTEL AND WITH TTEL

Core 0						
Without TTEL				With TTEL		
Burst Length	Max Latency (μs)	Min Latency (μs)	Jitter (μs)	Max Latency (μs)	Min Latency (μs)	Jitter (μs)
2	48.018	6.456	41.562	213.956	204.268	9.688
4	67.081	6.456	60.625	213.956	184.581	29.375
8	428.956	0.206	428.750	213.956	145.206	68.75
16	2094.893	1.143	2093.75	213.956	77.393	136.563

TABLE IX
PERFORMANCE ANALYSIS OF CORE 1; BEST-EFFORT (50MB/S); WITHOUT TTEL AND WITH TTEL

Core 1						
Without TTEL				With TTEL		
Burst Length	Max Latency (μs)	Min Latency (μs)	Jitter (μs)	Max Latency (μs)	Min Latency (μs)	Jitter (μs)
2	369.581	1.456	368.125	232.706	181.768	50.938
4	407.081	1.143	405.938	232.706	172.081	60.625
8	433.956	0.206	433.750	232.706	132.706	100
16	2286.456	3.643	2282.813	232.706	61.143	171.563

Table VIII and IX represent the obtained experimental data for the traffic class best-effort for 50MB/s. The plotted graphs are shown in the figures 10 and 11. Graph 10 presents the jitter fluctuation in Core 0. Similarly, graph 11 shows how jitter varies in Core 1. For designs with TTEL, the relationship between jitter and burst length appears to have a constant

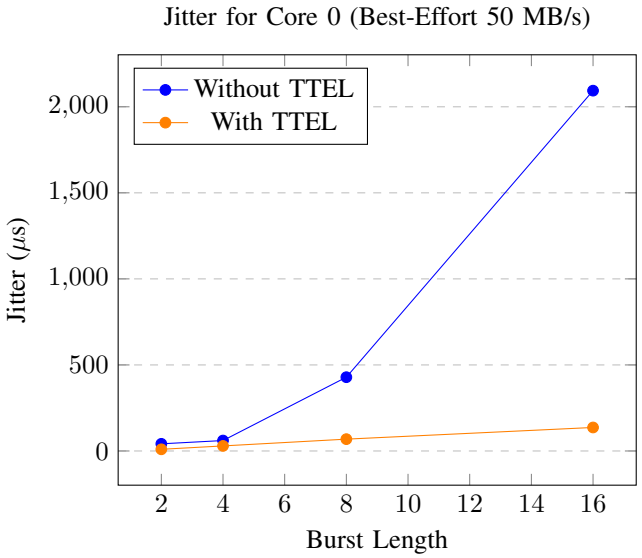

Fig. 10. Plot of Jitter against Burst Length

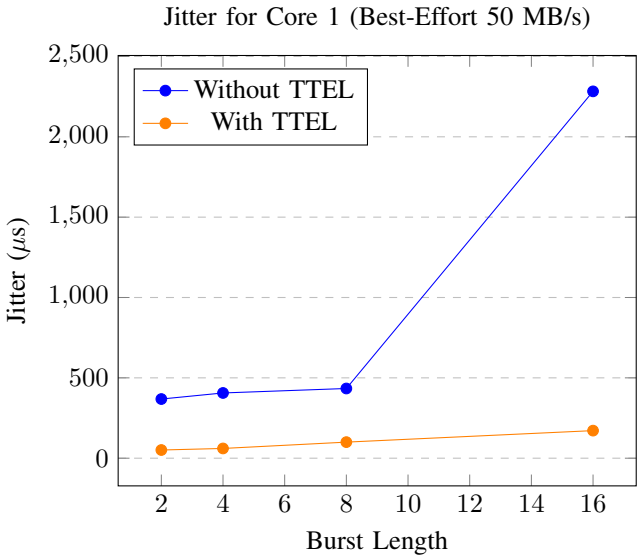

Fig. 11. Plot of Jitter against Burst Length

gradient for both cores. However, for designs without TTEL, the jitter rapidly rises as the burst length grows from 2 to 16 in both cores.

In summary, the TTEL shapes the traffic for the Versal NoC, thereby providing temporal guarantees for message transmission over the NoC.

VI. CONCLUSION

The goal was to design a Time-Triggered Network Interface Extension Layer for Versal Network-on-Chip to improve the performance of the Versal NoC. We extended the TTEL implementation in [3] with modifications to fit the Versal NoC.

Following observations were made. The TTEL provides temporal partitioning for the Versal NoC. Additionally, the jitter occurring in the NoC when two or more PEs trying to send the data over the shared routing path is reduced when using a TTEL. Furthermore the TTEL keeps the maximum latency of transferring messages constant for a particular traffic c lass i rrespective o f t he b urst l ength. T hus t he TTEL integration improves the Versal NoC by regulating the traffic to ensure low jitter.

ACKNOWLEDGEMENT

This work has been supported by the research project FRACTAL in part by the EC under grant number 877056 and the BMBF under grant number 16MEE015K.

REFERENCES

[1] G. Martin and H. Chang. System-on-chip design. In *ASICON 2001. 2001 4th International Conference on ASIC Proceedings (Cat. No.01TH8549)*, pages 12–17, 2001.

[2] Ian Swarbrick, Dinesh Gaitonde, Sagheer Ahmad, Bala Jayadev, Jeff Cuppett, Abbas Morshed, Brian Gaide, and Ygal Arbel. Versal network-on-chip (noc). In *2019 IEEE Symposium on High-Performance Interconnects (HOTI)*, pages 13–17, 2019.

[3] Hamidreza Ahmadian and Roman Obermaisser. Time-triggered extension layer for on-chip network interfaces in mixed-criticality systems. In *2015 Euromicro Conference on Digital System Design*, pages 693–699, 2015.

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

[5] 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.

[6] K. Goossens, J. Dielissen, and A. Radulescu. Aethereal network on chip: concepts, architectures, and implementations, 2005.

[7] Evgeny Bolotin, Israel Cidon, and Avinoam Kolodny. Qnoc: Qos architecture and design process for network on chip. *Journal of Systems Architecture*, 50:105–128, 02 2004.

[8] M. Millberg, E. Nilsson, R. Thid, S. Kumar, and A. Jantsch. The nostrum backbone-a communication protocol stack for networks on chip. In *17th International Conference on VLSI Design. Proceedings.*, pages 693–696, 2004.

[9] Akram Ben Ahmed, Daichi Fujiki, Hiroki Matsutani, Michihiro Koibuchi, and Hideharu Amano. Axnoc: Low-power approximate network-on-chips using critical-path isolation. In *2018 Twelfth IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, pages 1–8. IEEE, 2018

[10] Roman Obermaisser, Christian El Salloum, Bernhard Huber, and Hermann Kopetz. The time-triggered system-on-a-chip architecture. In *2008 IEEE International Symposium on Industrial Electronics*, pages 1941–1947, 2008.

[11] Ian Swarbrick, Dinesh Gaitonde, Sagheer Ahmad, Brian Gaide, and Ygal Arbel. Network-on-chip programmable platform in versaltm acap architecture. In *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '19, page 212–221, New York, NY, USA, 2019. Association for Computing Machinery.

[12] Dimitrios Soudris Konstantinos Tatas, Kostas Siozios and Axel Jantsch. Designing 2d and 3d network-on-chip architectures. In *Springer, 2014*. Springer, 2014.

[13] Airlines Electronic Engineering Committee et al. Arinc specification 664p7: Aircraft data network, part 7–avionics full duplex switched ethernet (afdx) network. *Aeronautical Radio Inc*, 2005.

[14] Sebastian Tobuschat and Rolf Ernst. Real-time communication analysis for networks-on-chip with backpressure, 2017.

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung