Toward Predictable Deflection Routing in Routerless NoCs for Real-Time Systems

M Norazizi Sham Mohd Sayuti
Faculty of Engineering and Built Environment
Universiti Sains Islam Malaysia
Negeri Sembilan, Malaysia
azizi@usim.edu.my

Leandro Soares Indrusiak
Distributed Systems and Services Group
School of Computer Science, University of Leeds
Leeds, United Kingdom
L.SoaresIndrusiak@leeds.ac.uk

***Abstract*—Contention between packets on shared resources, such as ejection links, causes high latency on routerless Networks-On-Chip. Missing packet timing guarantees due to high latency render a real-time application unschedulable. Deflection routing is a mechanism that addresses contention when multiple packets try to access the same links simultaneously. Despite its benefits, packets may be deflected more than what their timing guarantees allow in the worst-case scenario. To achieve an upper bound of deflections in conjunction with mapping, we proposed two methods: first, a method that increases the deflections of all communication flows, and second, a method that increases the deflections based on selective communication flows. Our findings, based on running both methods in an evolutionary optimisation, show that the latter outperformed the former, with a statistically significant improvement in the maximum deflection.**

***Index Terms*—networks-on-chip, routerless, deflection routing, evolutionary optimisation, design space exploration**

I. Introduction

Routerless Networks-On-Chip (NoCs) require limited buffering and flow control circuitry, providing benefits in terms of energy consumption. On a ring topology, multiple ring connections exist to transfer packets over the NoCs. A packet is injected flit-by-flit from the injection queue to an output port of the switch and travels to the next switch on its ring. Each switch has a single-flit buffer enough to store a flit on each ring. An output port does not have a buffer. Instead, each ring has a packet buffer to temporarily store a packet's flits if a newly injected packet occupies the intended output port to the next switch.

Despite such architecture, the ejection links of the switch are still shared among packets on multiple rings. If the shared links are currently occupied, the other packets are forced to perform several loops on their rings until the links are available for ejection to their destination core. This contention resolution mechanism is known as deflection routing.

Every deflection imposes a time delay on every packet, increasing its latency. Still, all packets of a real-time application must meet their timing guarantees; otherwise, the application will be deemed as unschedulable. This condition restricts the number of deflections permissible to every packet on its ring. An upper bound defines the maximum deflection of a packet to prevent it from looping beyond its timing guarantee. Having this upper bound is the first step; for packets with strict latency requirements, the situation worsens if more rings are connected to the processing core, with only small deflections permitted to mitigate the contention. Therefore, it is essential during the design space exploration to determine the upper bound in conjunction with task mapping, as it has the potential to dissipate packets under those requirements to other cores with less contention.

To address these challenges, we propose two optimisation methods to explore the deflection upper bound for the routerless NoCs. From the results, we demonstrate that achieving the upper bound in conjunction with task mapping is possible, and there is potential for further increases in the maximum deflections.

We organise the following sections as follows. Section II provides a background overview of the related works. Section III introduces the proposed optimisation methods to explore the maximum deflections bound. Section IV discusses the findings from the evaluation we performed on the proposed methods. Section V of the article concludes our findings and proposes future work.

This work was supported by a Universiti Sains Islam Malaysia research grant (Grant numbers: PPPI/USIM/FKAB/USIM/110523).

II. Background

Wormhole switching [1] is a popular network switching protocol in router-based Networks-on-Chip because it strikes a good balance between performance and buffer usage. Routers do not need to store entire packets in buffers—just a fixed, small size of data called flits is required to perform their routing functions [2]. This has its drawbacks, as small buffers mean there is not enough space to accommodate stalled packets. Packets can become stagnant at multiple routers and occupy several links simultaneously, making it challenging to predict network travel time. Furthermore, despite their superior scalability, they also incur significant power and area overhead due to the complexity of the router structures.

A deflection switch in a routerless NoC can be much smaller and faster than a wormhole or virtual cut-through switch. Liu et al. have proposed the Isolated multi-Ring (IMR) architecture [3], demonstrating improvements in throughput and average latency over several router-based NoC architectures, including Mesh and Torus networks. Following that work, another improvement has been suggested by Alazemi et al. [4]. Their routerless NoC eliminate routers and optimises on-chip wiring to achieve comparable hop count and scalability while reducing resource requirements. The proposed design shows significant power and area reduction while improving latency and throughput compared to IMR and some low-cost router-based NoC designs. Further study by Kapre et al. [5] has shown that the benefit of deflection routings extends to FPGA (Field Programmable Gate Array) overlay by eliminating FIFO (First-in First-out) buffers, and adopting a torus topology maximises resource efficiency by reducing crossbar complexity.

The role of mapping in the early design stage is essential, as it determines not only the response times of tasks running on processing cores but also the communication delays of packets originating from the tasks. The problem with deflection-routed NoCs is balancing the deflection of packets while optimising other parameters, such as task mapping. For a comprehensive review of dynamic mapping—also referred to as run-time mapping—and static mapping (design-time mapping), readers are directed to Saleem et al. [6] and Sahu et al. [7], respectively. Given the substantial influence of static task mapping on NoC with wormhole switching, particularly in the context of real-time applications, numerous prior studies [8], [9] have investigated its impact on time guarantees by employing the schedulability analysis [10].

It is common to see average performance metrics in most NoC design space explorations. A design space exploration tool, such as SUNMAP, automatically selects the best topology for a given application, then maps cores onto that topology to minimise average communication delay and other performance metrics [11], but does not support deflection router models. Simulators, such as those proposed in [12], support custom network topologies, detailed deflection router models, and various simulation modes, with a focus on average latency and power performance analysis. The use of machine learning-based methods in design space exploration has become a trend. This includes the work in [13], but, similar to the former works, it is challenging to effectively evaluate the NoCs under worst-case scenarios.

Recently, a new schedulability analysis [14] has been proposed to facilitate the worst-case analysis of packet latency in deflection-routed NoCs. Using this analytical approach, a thorough exploration of the upper bound on deflection can be performed at the early stage of design.

III. Deflection Routing Optimisation

The proposed deflection optimisation methods facilitate routerless NoCs design space exploration by optimising task mapping and determining the upper bound of deflections. To realise both optimisation methods for the worst-case scenario, the schedulability analysis in section III-B together with the related functions in section III-D and III-E were integrated with an evolutionary algorithm in section III-C. In the following sections, we explain each of the components.

A. Routerless NoCs Model

Let us elucidate the routerless NoC model used in our methods. As depicted in Fig. 1 the routerless NoC model contains a set of rings $O = \{o_1, o_2, ..., o_k\}$, switches $\xi = \{\xi_1, \xi_2, ..., \xi_m\}$, and a set of processing cores $\Pi = \{\pi_1, \pi_2, ..., \pi_m\}$. Each switch ξ_i has a processor counterpart π_i, connected with two unidirectional links to and from the switch. Each ring o_i passes through an ordered set of switches $\Xi_i^o \in \xi$, and each switch could serve multiple rings. The switch must have a buffer b^o with sufficient size to accommodate even the largest packet to that ring o_i. A set of buffers in each ring is defined as a set of ordered buffers $B = \{b_1, b_2, ..., b_n\}$.

Each processing core π_i runs a set of mapped tasks denoted as $\Psi_i = \{\lambda_1, \lambda_2, ..., \lambda_r\}$. Each Ψ_i is a subset of a real-time application $\Lambda = \{\Psi_i, \Psi_i, ..., \Psi_r\}$.

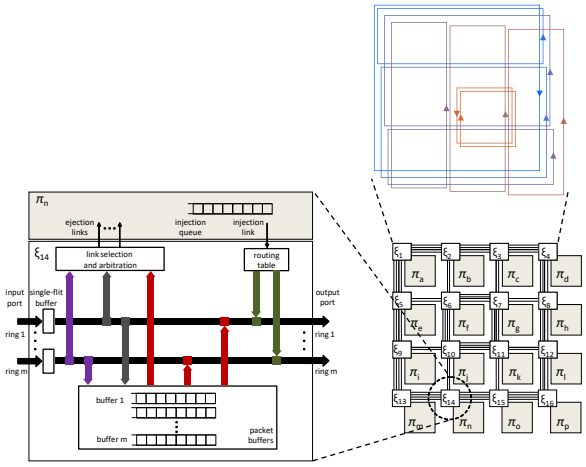

Fig. 1. Routerless network switch

All task sets of a real-time application are mutually exclusive; that is, once a set of tasks is mapped on a processing core, it remains on it until the application terminates.

In this system model, switches communicate with each other through one of their connected rings by sending communication packets, flit by flit, through each link on the ring until all of them arrive at the destination core. Packet transmission in this model is also known as traffic flow. Traffic flows of a real-time application belong to a set $\Gamma = \{\tau_1, \tau_2, ...\tau_r\}$. A real-time traffic flow is characterised based on tuple $\tau_i = \{T_i, D_i, L_i, J_i, \pi_i^s, \pi_i^d\}$. The source π_i^s is the processing core that releases the flow's packets to its corresponding switch's output port, and we assume that each packet's release is either periodic or sporadic. The minimum interval between two packet releases is defined by the attribute T_i, while the maximum delay of the flow's arrival at the destination core π_i^d is defined by the attribute D_i. A packet contains a load L_i delivered as flits. As a ring forwards one flit per cycle, L_i is also the time it takes for the flits of the packet to travel a link. We assume that the $D_i \leq T_i$ for simplicity, because every packet from the same flow always carries the same amount of load. The deviation from the successive release is also taken into consideration, denoted as J_i.

B. Schedulability Analysis

The schedulability analysis must account for the maximum deflections to calculate the worst-case latency. First, we assume that each ring in the routerless NoC has independent injection links. Thus, the whole packet from a traffic flow could be directed to the ring without experiencing worst-case interference from other traffic flows. Once the output link becomes available, it is immediately acquired by the traffic flow to send packets to the next downstream switch.

The worst-case latency of the flow is then modelled as follows. The worst-case interference before injection I_i^{pre} is the amount of interference experienced before the flow can have access to the output link of the switch. For a detailed explanation of the equation, interested readers are referred to the reference [14].

$$I_i^{pre} = 1 + \sum_{\tau_j \in \Gamma_{in_i}} L_j + \sum_{\tau_j \in \Gamma^o_{up_i}} \left\lceil \frac{I_i^{pre} + J_j + J_j^k}{T_j} \right\rceil \cdot L_j + \sum_{\tau_j \in \Gamma^o} \sum_{1}^{\text{maxdef}_j} \left\lceil \frac{I_i^{pre} + J_j + J_j^k}{T_j} \right\rceil \cdot L_j \tag{1}$$

We explain further the fourth term of the equation, as it has a direct relation to the maximum deflection. Considering the ejection link is shared among packets, the fourth term of (1) represents the additional interference caused by the deflections of packets using that ring. This interference, causing the injection delay, is modelled by including each of the packet deflections, accounting for them as if they are replicas of the original packet flow, producing interfering packets with the same period, jitter, and upstream indirect interference.

Equation (2) provides the worst-case interference I_i^{pos} for a packet τ_i after ejection, with the additional latency as the result of $maxdef_i$ deflections. To account for the latency, links on which a packet must travel while circling the ring must be considered; thus (3) is factored with $maxdef_i$ deflections to account for that.

$$I_i^{pos} = I_i^{pos} + maxdef_i \cdot I_i^{defl} \tag{2}$$

$$I_i^{defl} = \sum_{\xi \in \Xi^o} B_\xi^o \tag{3}$$

We can determine the total worst-case latency experienced by any packet τ_i by substituting (1) and (2) into (4). C_i is added into the first term as the maximum latency when no contention exists on the ring where the flow travels from π_i^s to π_i^d. Every deflection incurs no-load latency, and to account for every deflection, it must be factored with the number

of switches r_o on that ring (or in set Ξ^o) and the number of maximum deflections $maxdef_i$. Then, the total worst-case latency experienced by flow τ_i is

$$R_i = C_i + r^o \cdot \text{maxdef}_i + I_i^{pre} + I_i^{pos}. \quad (4)$$

C. Optimisation Algorithm

A routerless NoC depends on the deflection parameter to dissipate contention at the ejection links. Increasing this parameter value to allow communication flows to run several cycles at their respective rings, however, imposes delays as indicated in (1). Analysis based on (5) and (6) will determine if a given value of the parameter will allow a communication flow to become schedulable in the system or otherwise. Therefore, we defined the maximisation of the sum of all schedulable flows in (7) as the optimisation objective, as follows.

$$S_i = \begin{cases} 1, & \text{for } R_i \leq D_i \quad (5) \\ 0, & \text{for } R_i > D_i \quad (6) \end{cases}$$

$$Obj = max \sum S_i \quad (7)$$

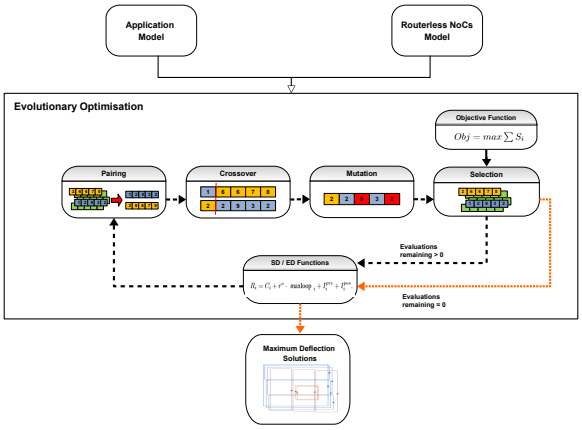

Fig. 2. Optimisation flow

Now, since task mapping allocates application tasks among processing cores, it also affects the ring paths as well as the communication flows that utilise them. Thus, contention at the ejection links of a processing core may differ from that of its counterpart. Based on this notion, and guided by (7), we explore the task mapping and the deflection parameter by optimising them both in the optimisation framework depicted in Fig. 2. Optimisation of both parameters requires two inputs from the system model: a real-time application model and a routerless NoC model. The application tasks are allocated to the processing cores based on the mapping extracted from the population chromosomes of the single-objective evolutionary algorithm (SOEA). Similarly, the deflection activation mode (refer to section III-E) for each communication flow is extracted from the population chromosomes. The maximum number of iterations that the optimisation can perform is dictated by the number of evaluations in its configuration.

In brief, SOEA's underlying principle is based on the concept of evolution in biology that describes how inherited traits in populations change over generations. This process is driven by the natural selection of individuals' traits that help them survive and thrive in their environment, as they reproduce and pass those traits on to their offspring. Interested readers are referred to [9] for a detailed explanation of the typical processes of the algorithm.

Each flow will experience a minimum amount of deflections on its ring after mapping. Assuming that the destination core of all the flows on the shared ring is the same, including the observed one. Then, the latter will experience an amount of deflections equivalent to the number of flows on the shared ring minus itself. This value becomes the initial value of the optimisation methods. Each proposed method will increment the minimum value by one until an upper bound is reached, at which point the system will become unschedulable.

D. Scaling Deflection Function

The scaling deflection (SD) function, as depicted in Algorithm 1, increments the deflection parameter value to find the upper bound for all flows. This is performed first by analysing the schedulability of the system after every mapping. If all the flows are schedulable, then the function will be called to increment the deflection value of each flow by one ($maxdef_i^{opt} \leftarrow maxdef^{init} + 1$). The system will be analysed again for its schedulability (in the if statement). The same routine continues until the schedulability test fails. Then, the previously incremented values will be returned as the upper bound of deflections in conjunction with the mapping.

E. Evolutionary Deflection Function

Evolutionary deflection (ED) performs a deflection increment based on the activation provided by the SOEA's chromosome. Two discrete values determine the activation. Value one (1) means the flow's deflection will be increased, and zero (0) means its current deflection value will remain. As shown in Fig. 3,

Algorithm 1 Scaling Deflection Function

Input: ring set O
Output: upper bound deflection of all flows $maxdef^{opt}$
$schedulable \leftarrow true$;
while *schedulable is true* **do**
  $R \leftarrow 0$ $maxdef_{init} \leftarrow 0$;
  foreach o *in* O **do**
    foreach Γ^o *in* o **do**
      $maxdef^{init} \leftarrow \sum \Gamma^o$;
      foreach τ_i *in* Γ^o **do**
        $maxdef_i^{opt} \leftarrow maxdef^{init} + 1$;
      end
    end
  end
  foreach τ_i *in* Γ **do**
    calculate R_i^{new} from equation;
    if $R_i^{new} <= D_i$ **then**
      $R \leftarrow R + 1$;
    end
  end
  if $R == \sum \tau_i \in \Gamma$ **then**
    $schedulable \leftarrow true$;
  end
  else
    $schedulable \leftarrow false$;
  end
end

in the second half of the chromosome (in orange), not every flow will be activated. This characteristic depends on the traits passed among the population. For an inactivated flow, its deflection value remains the same as the initial optimisation configuration.

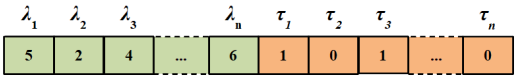

Fig. 3. Evolutionary algorithm chromosome

First, the system is analysed for schedulability. If it is schedulable, the function is called, and the deflection value of every activated flow will be increased by one (within the innermost foreach). Subsequently, the system will be re-evaluated for schedulability (in the if statement). If it is schedulable, then the same routine continues; otherwise, the last deflections of all flows will be returned as the upper bound. The function is depicted in the Algorithm 2.

Algorithm 2 Evolutionary Deflection Function

Input: ring set O
Output: upper bound deflection of all flows $maxdef^{opt}$
$schedulable \leftarrow true$;
while *schedulable is true* **do**
  $R \leftarrow 0$ $maxdef_{init} \leftarrow 0$;
  foreach o *in* O **do**
    foreach Γ^o *in* o **do**
      $maxdef^{init} \leftarrow \sum \Gamma^o$;
      foreach τ_i *in* Γ^o **do**
        if τ_i *is selected by EA* **then**
          $maxdef_i^{opt} \leftarrow maxdef^{init} + 1$;
        end
      end
    end
  end
  foreach τ_i *in* Γ **do**
    calculate R_i^{new} from equation;
    if $R_i^{new} <= D_i$ **then**
      $R \leftarrow R + 1$;
    end
  end
  if $R == \sum \tau_i \in \Gamma$ **then**
    $schedulable \leftarrow true$;
  end
  else
    $schedulable \leftarrow false$;
  end
end

IV. Results

In Section III-B, we demonstrated from the analysis that deflection is a factor that increases the worst-case latency. We will explain in the following subsections that if task mapping and the deflection parameter are sufficiently optimised, it is possible to find the upper bound deflections of the flows within their timing guarantees.

A. Evaluation Setup

For comparison, a baseline (B) was developed using the same SOEA as the proposed methods (SD and ED). Its distinguishing characteristic is that the total deflection of a flow is equivalent to the number of flows on each ring. This setting would allow a comparison in terms of schedulability before and after the deflection parameters were optimised. Our goals from this evaluation are as follows.

- To analyse the schedulability ratio of both methods.

- To analyse the maximum deflections of flows from both proposed methods.
- To analyse both methods statistically to gain confidence in the maximum deflection improvement.

The attributes of the task and flow tuples characterise the application model. By characterising these attributes, 20 synthetic applications were produced, with the smallest application containing five tasks and the largest containing one hundred tasks. Each application is a five-task increment over the previous. This would allow us to evaluate the proposed methods using various application loads on the system platform. In this work, we utilised the same number of traffic flows as the tasks. One of the attributes of traffic flow is the payload, which is the number of flits a packet delivers from a source to a destination. Using this attribute, we can further characterise the traffic flows and generate uniformly distributed flits between 1 and 126.

We modelled a single hardware platform following a regular mesh-based topology. It contains four rows and columns (4x4), with 16 processor cores interconnected through routerless NoC rings via switches, as depicted in Fig. 1.

B. Dataset Generation

All our analyses used the same datasets. Here, we provide a detailed explanation of how these datasets were created. Each proposed method and the baseline produced 20 datasets, contributing to a total of 60 datasets. To make a dataset, the proposed methods and baseline must run 25 times using the same task set to generate an output file containing 25 data samples. Every sample represents the highest incremental maximum deflection value obtained during a run. This value is determined by calculating the incremental maximum deflection for every flow using (8). Subsequently, this process was repeated for the other task sets until all 60 datasets were produced.

C. Evaluation of Schedulability Ratio

Schedulability refers to the condition under which an application can be scheduled to meet its timing guarantees. To determine if the condition exists, we refer to the schedulability analysis in section III-B. This analysis evaluates the worst-case response times of tasks and latencies of flows at two points during optimisation: first, following the completion of task mapping, and second, after every deflection increment. Suppose the executions of all tasks at the originating cores and the delivery of all packets to the destination cores are completed within the stipulated timing guarantees in the worst-case scenario. In that case, the application is deemed schedulable.

The schedulability ratio is a metric that defines the percentage of all tasks and traffic flows that meet their timing guarantees. For example, a value of 0.5 indicates that the condition is met by only 50% of the tasks and flows. In this case, the application is deemed unschedulable because its timing guarantees are not fully met. The alternative hypothesis for this comparison is that the proposed methods could meet the application timing guarantees for up to 60% of the flow sets, at least as well as the baseline achieved.

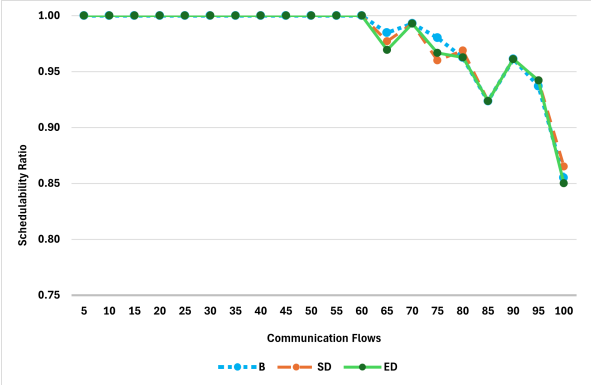

Fig. 4. Schedulability ratio comparison with various flow sets

In terms of the schedulability ratio, SD, ED, and the baseline show similar performance. Referring to Fig. 4, from the application with the lowest flows to the application with 60 flows, both methods and the baseline achieve full schedulability (100%). On the 4x4 NoC platform, however, an application with over 60 flows imposed too much contention on the NoCs; thus, meeting the condition proved very difficult. As shown in the figure, the schedulability performance further degrades as the number of flows increases.

D. Evaluation of Deflection Improvement

In this evaluation, the initial maximum deflection represents the default number of loops a flow could travel around its ring. For both methods, this value serves as the initial maximum deflection ($maxdef^{init}$), from which the deflection optimisation runs to increment as many loops as possible for every flow (in the case of SD) or activated flow (in the case of ED) within the application's timing guarantees. The final value of the maximum deflection after optimisation is known as the optimised maximum deflection value ($maxdef^{opt}$).

Our comparison in this section applies the value difference between the initial maximum deflection and the optimised maximum deflection. The difference between these parameter values is denoted as $maxdef^{dif}$. For every flow i the maximum deflection difference is defined as follows.

$$maxdef_i^{dif} = maxdef_i^{opt} - maxdef^{init} \quad (8)$$

An application contains several flows; for the maximum deflection performance comparison, only the flow that has the maximum $maxdef^{dif}$ from all 25 runs is shown in Fig. 5. It is noted here that only twelve flow sets are displayed in the graph, since flow sets with over 60 flows are not schedulable (refer to Fig. 4). The deflection optimisation is only effective if the application is schedulable.

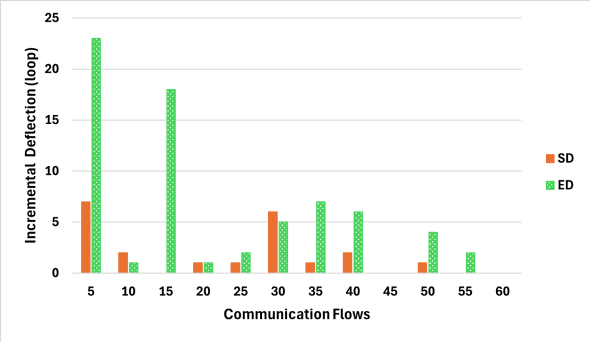

Fig. 5. Maximum deflection performance comparison with various flow sets

The alternative hypothesis of this evaluation states that the incremental deflection of the selected flow is higher than any of the flows incremented by SD, when both task mapping and deflection are optimised simultaneously. Among all the task sets, ED exhibits a higher maximum deflection than SD. Seven task sets from ED compared with only two for SD, and the remaining three task sets are equivalent in terms of performance. The highest maximum deflection shown by ED, which overcomes the highest performance shown by SD (at task set 30), is depicted at task sets 35, 15, and 5. Neither the SD nor the ED method shows any incremental maximum deflection at the two task sets of 45 and 60, thus they are considered equivalent in terms of performance.

E. Statistical Tests

The evolutionary algorithm operates using a random mechanism. Although this algorithm has demonstrated exemplary performance, its solutions are always suboptimal. Furthermore, it relies on random functions to operate; thus, there is a possibility that it could fail to find any solution.

We run the Wilcoxon signed-rank statistical test to gain confidence from the incremental maximum deflection samples that the SD and ED produced. It compares whether there is a significant shift in the median between them. The paired samples Wilcoxon test is a non-parametric test based on paired data. In total, 25 samples were collected from each SD and ED to use for this test.

The following hypotheses serve as the basis for conducting the test using paired data from both methods, with an alpha significance level set at 0.05. The following is the statistical hypothesis.

- H0: Evolutionary optimisation with either SD or ED shows no significant difference between their observations.
- H1: Evolutionary optimisation with ED produces higher incremental maximum deflection performance than SD.

We conducted the statistical tests based on three flow sets: 20, 30, and 50. The selection of these flow sets is based on the fact that they exhibited different performance profiles. In the flow set with 20 flows, they are equivalent; in the flow set with 30 flows, SD is superior; and in the flow set with 50 flows, ED is superior.

Table I summarises the results from the statistical test. For all the selected flow sets, ED shows significant shifts at the median in different flow sets. For example, the p-value of the test using the flow set with 20 flows is 0.01166, less than the significance level alpha 0.05. Here, we can conclude that the median weight of the ED is significantly different from that of the SD. Thus, we can reject the null hypothesis and concur with the alternative hypothesis that the incremental maximum deflection produced by ED is greater than that produced by SD.

TABLE I
WILCOXON SIGNED-RANK STATISTICAL TEST RESULTS

Task set	p-value
20	0.01166
30	0.04609
50	0.00196

V. CONCLUSION AND FUTURE WORK

The study highlights the importance of optimising deflection bounds to enhance real-time communication performance. To address this, two evolutionary optimisation methods were proposed to explore and

maximise deflection limits. Results show that these methods can effectively identify maximum deflection bounds, with potential for further improvements. In future work, the study on the maximum deflection bound could be extended by including other routerless NoC resources to enhance the optimisation framework further.

ACKNOWLEDGMENT

This work was supported by a Universiti Sains Islam Malaysia research grant (Grant numbers: PPPI/USIM/FKAB/USIM/110523).

REFERENCES

[1] L. Ni and P. McKinley, "A survey of wormhole routing techniques in direct networks," *Computer*, vol. 26, no. 2, pp. 62–76, Feb. 1993, ISSN: 0018-9162. DOI: 10.1109/2.191995.

[2] R. Singh, M. K. Bohra, P. Hemrajani, *et al.*, "Review, Analysis, and Implementation of Path Selection Strategies for 2D NoCs," *IEEE Access*, vol. 10, pp. 129 245–129 268, 2022, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2022.3227460.

[3] Shaoli Liu, L. Shaoli, Tianshi Chen, *et al.*, "IMR: High-Performance Low-Cost Multi-Ring NoCs," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 6, pp. 1700–1712, Jun. 2016. DOI: 10.1109/tpds.2015.2465905.

[4] Fawaz Alazemi, Fawaz M. Alazemi, F. M. Alazemi, *et al.*, "Routerless Network-on-Chip," *International Symposium on High-Performance Computer Architecture*, pp. 492–503, Feb. 2018. DOI: 10.1109/hpca.2018.00049.

[5] N. Kapre and J. Gray, "Hoplite: A Deflection-Routed Directional Torus NoC for FPGAs," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 10, no. 2, pp. 1–24, Jun. 2017, ISSN: 1936-7406, 1936-7414. DOI: 10.1145/3027486.

[6] S. Saleem, F. Hussain, W. Amin, *et al.*, "A Survey on Dynamic Application Mapping Approaches for Real-Time Network-on-Chip-Based Platforms," *IEEE Access*, vol. 11, pp. 122 694–122 721, 2023, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2023.3329233.

[7] P. K. Sahu and S. Chattopadhyay, "A survey on application mapping strategies for Network-on-Chip design," *Journal of Systems Architecture*, vol. 59, no. 1, pp. 60–76, Jan. 2013, ISSN: 13837621. DOI: 10.1016/j.sysarc.2012.10.004.

[8] M. N. S. M. Sayuti and L. S. Indrusiak, "Real-time low-power task mapping in networks-on-chip," in *VLSI (ISVLSI), 2013 IEEE Computer Society Annual Symposium On*, 2013. DOI: doi.org/10.1109/ISVLSI.2013.6654616.

[9] M. N. S. M. Sayuti, L. S. Indrusiak, and A. Garcia-Ortiz, "An optimisation algorithm for minimising energy dissipation in NoC-based hard real-time embedded systems," in *Proceedings of the 21st International Conference on Real-Time Networks and Systems*, Sophia Antipolis, France, 2013, ISBN: 978-1-4503-2058-0. DOI: 10.1145/2516821.2516844.

[10] L. S. Indrusiak, "End-to-end schedulability tests for multiprocessor embedded systems based on networks-on-chip with priority-preemptive arbitration," *Journal of Systems Architecture*, 2014. DOI: doi.org/10.1016/j.sysarc.2014.05.002.

[11] M. Srinivasan and G. De Micheli, "SUNMAP: A tool for automatic topology selection and generation for NoCs," in *Proceedings of the 41st Annual Design Automation Conference*, ser. Dac '04, New York, NY, USA: ACM, 2004, ISBN: 1-58113-828-8. DOI: 10.1145/996566.996809.

[12] G. Oxman and S. Weiss, "An NoC Simulator That Supports Deflection Routing, GPU/CPU Integration, and Co-Simulation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 10, pp. 1667–1680, Oct. 2016, ISSN: 0278-0070, 1937-4151. DOI: 10.1109/TCAD.2016.2527698.

[13] Ting-Ru Lin, T.-R. Lin, Drew Penney, *et al.*, "A Deep Reinforcement Learning Framework for Architectural Exploration: A Routerless NoC Case Study," *International Symposium on High-Performance Computer Architecture*, pp. 99–110, Feb. 2020. DOI: 10.1109/hpca47549.2020.00018.

[14] L. Soares Indrusiak and A. Burns, "Real-time guarantees in routerless networks-on-chip," *ACM Transactions on Embedded Computing Systems*, vol. 22, no. 5, pp. 1–27, 2023. DOI: doi.org/10.1145/3616539.