Slack-Aware Packet Approximation for Energy-Efficient Network-on-Chips

Yuechen Chen, *Member, IEEE,* Ahmed Louri, *Fellow, IEEE,* Shanshan Liu, *Member, IEEE,* and Fabrizio Lombardi, *Fellow, IEEE*

Abstract— Network-on-Chips (NoCs) are the standard on-chip communication fabrics for connecting cores, caches, and memory controllers in multi/many-core systems. With the increase in communication load introduced by emerging parallel computing applications, on-chip communication is becoming more costly than computation in terms of energy consumption. This paper contributes to existing research on approximate communication by proposing a slack-aware packet approximation technique to reduce the energy consumed by NoCs for sustainable parallel computation. The proposed approximation technique lowers both the execution time and NoC power consumption by reducing the packet size based on slack. The slack is the number of cycles by which a packet can be delayed in the network with no effect on execution time. Thus, low-slack packets are considered critical to system performance, and prioritizing these packets during the transmission will significantly reduce execution time. The proposed technique includes a slack-aware control policy to identify low-slack packets and accelerates these packets using two packet approximation mechanisms, namely, an in-network approximation (INAP) and a network interface approximation (NIAP). INAP mechanism prioritizes low-slack packets during the arbitration phase of the router by approximating packets with high-slack. NIAP mechanism reduces the latency of the network links and switch traversals by truncating data for the low-slack packets. An approximate network interface and router are implemented to support the proposed technique with lightweight packet approximation hardware for lower power consumption and execution time. Cycle-accurate simulations using the AxBench and PARSEC benchmark suites show that the proposed approximate communication technique achieves reductions of up to 24% in execution time and 38% in energy consumption with 1.1% less accuracy loss on average compared to existing approximate communication techniques.

Index Terms— Approximate Communication, Energy Consumption, Network-on-Chips (NoCs)

1 Introduction

NETWORK-ON-CHIPS (NoCs) play a critical role in the performance of multi/many-core processors. Because of the heavy NoC communication loads of current parallel computing applications, such as big data and machine learning applications, the NoC is becoming a serious bottleneck affecting both power consumption and execution time [3], [4]. Existing research shows that the energy consumed for on-chip communication can easily exceed the energy consumed for computation in a multicore processor [5]–[10]. Thus, there is a need for innovative energy reduction techniques for future NoC designs.

Recent research on approximate communication leverages the error tolerance of applications [11], [12] to reduce network latency and dynamic power consumption [4], [13]–[15]. A typical approximate communication framework consists of two components: a quality control method and a packet approximation technique [13], [16], [17]. The quality control methods are implemented in software to ensure the result quality by analyzing each application and annotating error-resilient variables [16], [17]. The packet approximation techniques are implemented in the NoC to reduce the size of the error-resilient variables before packet transmission [4], [15], [18]–[20]. This reduction in the amount of transmitted data leads to improvements in NoC performance, including lower dynamic power consumption, reduced network latency, and increased network throughput. However, reducing the dynamic power consumption alone is not sufficient for energy reduction. Since energy is the product of power and execution time, the key to energy-efficient on-chip communication is to reduce both values using approximation methods. Most existing works [3], [4], [13]–[20] utilize lossy data compression mechanisms to achieve a smaller packet size for better NoC performance. However, software-based quality control methods have limited ability to reduce execution time. Since the software cannot predict on-chip communication before execution, the performance-critical packets, which will significantly impact the execution time if delayed during the transmission, cannot be identified and accelerated during communication. As a result, even though a large number of packets are approximated, existing techniques [4], [13]–[17], [19] achieve relatively little improvement in energy consumption. As shown in Fig. 1, an average of 95% of the data packets are approximated, but this approach reduces the energy consumption by only 8% on average.

Another widely used class of energy reduction techniques for NoCs is dynamic voltage and frequency scaling (DVFS) [21]–[24]. In these techniques, the NoC is first partitioned into several voltage/frequency domains. Then, a control policy is developed to dynamically adjust the supply voltages and frequencies of the components of the NoC according to the currently observed or predicted network status (e.g., link utilization, cache traffic, etc.). Through voltage and frequency reduction, the overall power consumption for on-chip com-

- *Y. Chen and A. Louri are with the George Washington University, Washington, DC 20052. E-mail: yuechen@gwu.edu, louri@gwu.edu.*
- *F. Lombardi is with the Northeastern University, Boston, MA 02115. E-mail: lombardi@ece.neu.edu*
- *S. Liu is with the New Mexico State University, Las Cruces, NM 88001. E-mail: ssliu@nmsu.edu*

This article has been accepted for publication in IEEE Transactions on Sustainable Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TSUSC.2022.3213469

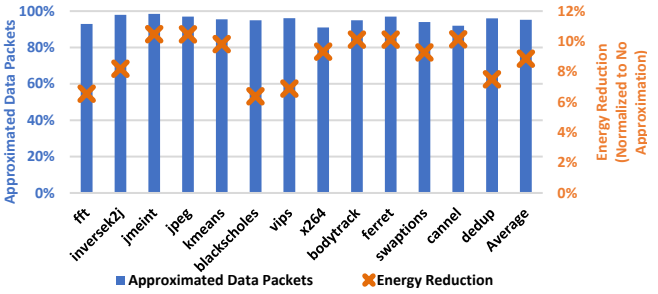


Fig. 1. Approximated data packets and energy reduction for [16] compared to a NoC with no approximation.

munication is reduced. However, these reductions in voltage and frequency simultaneously lead to greater network latency and a longer execution time, resulting in a small overall reduction in NoC energy consumption. Thus, to achieve significant savings in NoC energy consumption, reducing both power consumption and execution time is critical.

In this paper, we propose a slack-aware packet approximation technique to achieve a reduction in NoC energy consumption for sustainable parallel computing. The slack of a packet is defined as the number of cycles by which the packet can be delayed in the network with no effect on execution time [25]–[27]. Low-slack packets are considered critical to system performance, and network latency for these packets often results in processor stalls. In comparison, high-slack packets can tolerate considerable network latency without causing processor stalls. The proposed technique includes a slack-aware control policy to identify low-slack packets and accelerates these packets to reduce both the NoC power consumption and the execution time of applications using two lightweight packet approximation techniques, namely, an in-network approximation (INAP) mechanism and a network interface approximation (NIAP) mechanism. The NIAP mechanism utilizes a lightweight data approximation method (truncation) to reduce the latency of network links and switch traversals for the low-slack packets. The INAP mechanism is designed to reduce the latency of arbitration in the routers for low-slack packets by approximating high-slack packets. The slack-aware control policy is designed to identify low-slack packets and activate the appropriate packet approximation mechanism to achieve a lower execution time. To support the proposed technique, an approximate network interface and an approximate router are implemented. The proposed implementations reduce the packet size using lightweight data approximation logic for lower power consumption. The simultaneous reduction in both execution time and power consumption leads to lower NoC energy consumption. Specifically, the contributions of this work are as follows:

- A slack-aware packet approximation technique is proposed to reduce the energy consumption of NoCs for sustainable parallel computing.
- An approximate router and an approximate network interface are implemented to support the slack-aware control policy, the INAP mechanism, and the NIAP mechanism to reduce the network latency for the low-slack packets.
- A detailed performance evaluation is conducted to show that the proposed technique reduces the execution time and energy consumption by up to 24% and 38%, respectively, compared to existing approximate communication techniques with 1.1% lower accuracy degradation on average.

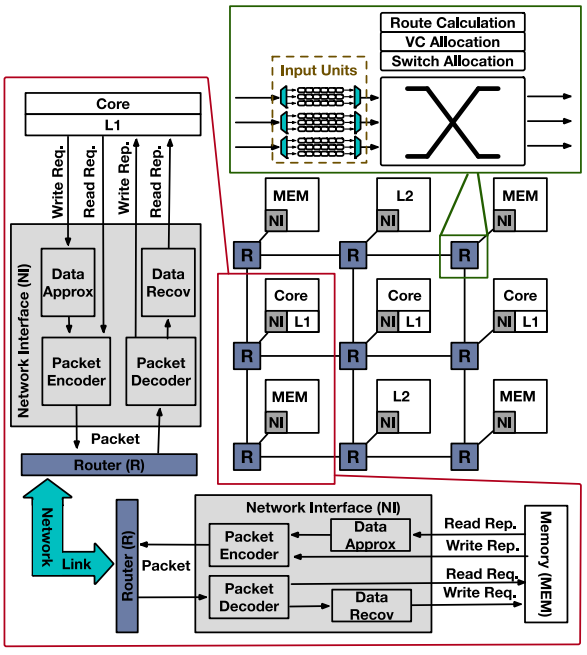


Fig. 2. Multicore architecture with an approximate communication NoC.

This paper is organized as follows. Section 2 presents the background of the proposed slack-aware packet approximation technique. Section 3 outlines the basic operational principles of the proposed technique. The implementation is described in detail in Section 4, while Section 5 presents its extensive evaluation. Section 6 concludes the manuscript.

2 Background

2.1 Network-on-Chips (NoCs)

The increasing scale of data movement in emerging parallel computing applications is resulting in heavy NoC communication loads [3], [4]. Moreover, as the numbers of processing cores, caches, and on-chip memories in multi/many-core systems increase, communication is becoming more costly than computation in terms of energy consumption [5]–[10].

Fig. 2 shows an NoC implemented in a multicore system with approximate communication capabilities. The NoC comprises network interfaces (NIs) and routers. Each NI contains a packet encoder/decoder to convert between request/reply packets. Data approximation/recovery modules process the data for approximate communication, which will be further discussed in Section 2.2. When a cache miss occurs during a memory load operation, a read request packet is sent to the memory or the shared cache through the NoC. Then, the memory or shared cache uses a read reply packet to send the required data back to the core. When a cache miss occurs during a memory store operation, the data are incorporated into a write request packet and sent to the memory or shared cache through the NoC. After the memory or shared cache receives the data, a write reply is sent back to the core to confirm a successful memory write.

The routers in Fig. 2 transmit a packet in seven steps: buffer write, route computation, virtual channel (VC)

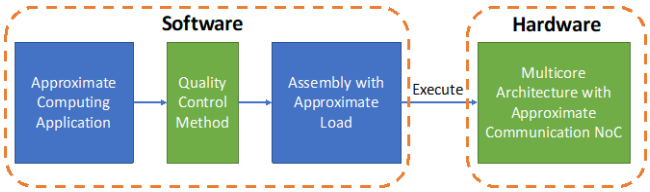


Fig. 3. High-level workflow of the current approximate communication framework.

allocation, switch allocation, buffer read, switch traversal, and link traversal. When a packet arrives at the router, the packet is first written to the buffer in the input unit during the buffer write stage. Then, the route computation logic calculates the output port for the incoming packet. For VC allocation, a round-robin arbitrator is used to choose one VC in each input unit. Afterward, for switch allocation, a round-robin arbitrator is again used to choose an input unit if more than one input unit is requesting to connect to the same output port. When a packet wins both arbitrations, that packet is removed from the buffer in the input unit (buffer read) and traverses the allocated switch and link.

2.2 Approximate Communication Techniques

Existing research shows that approximate communication techniques improve communication performance in multicore systems while ensuring result quality for parallel computing applications [4], [13]–[16], [19]. Approximate communication techniques trade result quality for lower dynamic power consumption, reduced network latency, and increased network throughput. The general working process of the current approximate communication framework is shown in Fig. 3. This framework includes a software-based quality control method and a hardware-based data approximation technique.

The quality control method ensures that the data error introduced by approximate communication can be tolerated by each approximate computing application [13], [16], [17]. Existing methods [4], [14], [15], [17], [20] allow program designers to assign error thresholds to each variable before the execution of an application. In [16], [17], software-based quality control systems are introduced to automatically determine the approximation level for each variable while ensuring result quality. These methods first identify error-resilient variables and calculate the error tolerances of these variables based on the application's requirements in terms of the result quality. Then, the load and store instructions for error-resilient variables are replaced with corresponding approximate load and store instructions for packet approximation during execution. These approximate load and store instructions contain approximation information, including the approximation level for the corresponding variable and the variable type (e.g., int or float). Finally, the application with the approximate load and store instructions is executed on a multicore architecture with an approximate communication NoC. Since the quality control method identifies the approximable variables and their approximation levels, in the existing approximate communication frameworks [4], [13]–[16], [19], the number of approximable packets is determined before the execution of the application. Because the traffic pattern cannot be analyzed and, hence, performance-critical packets cannot be identified by the software before the execution, the absence of this functionality results in only a small reduction in execution time.

The data approximation module in the NI (Fig. 2) reduces the packet size of the read reply and write request according to the approximation information during a memory load or store operation. The approximated packet carries the approximated data and the approximation information to the destination node. When the approximated packet reaches the destination node, the packet's approximated data are recovered to their original length according to the approximation information by the data recovery module. In existing research, several data approximation techniques have been implemented to reduce the packet size [4], [13]–[16], [19]. For example, [4] and [15] use lossy data compression techniques to approximate data in the write request or read reply reducing on-chip communication. Although existing techniques achieve a reduction in dynamic power consumption, a lossy data compression method requires complicated logic to process data, incurring considerable overheads in terms of on-chip area and static power for data approximation. As a result of the combined effect of the small execution time reduction and the additional static power consumed by the data approximation logic, the existing methods achieve a low reduction in energy consumption compared to a conventional NoC, even with a large number of approximated packets. Since on-chip communication can consume a large amount of energy in modern multicore processors [5]–[10], it would be of interest to explore the possibility of developing an approximate communication technique to address the energy consumption issue.

2.3 Dynamic Voltage and Frequency Scaling (DVFS)

Dynamic voltage and frequency scaling (DVFS) techniques have been extensively studied for energy-efficient on-chip communication[21]–[24]. In these techniques, the NoC is divided into several voltage/frequency domains. Then a control policy for voltage/frequency adjustment is developed. The control policy observes the NoC status in each voltage/frequency domain and makes voltage/frequency adjustments using various algorithms. For example, in [22], [24], prediction methods are presented to proactively adjust the voltage and frequency. Although DVFS can achieve a significant reduction in power consumption, these techniques simultaneously result in longer network latency and lower network throughput. The increase in network latency results in a longer execution time and, thus, higher energy consumption. Moreover, because of the heavy NoC communication loads imposed during the execution of current parallel computing applications, such as big data and machine learning applications, the performance penalty incurred by reducing the router voltage and frequency makes this approach no longer feasible in the future multi/many-core systems.

2.4 Slack

Slack is widely used in modern microprocessors to measure the performance criticality of a packet [25]–[27]. The packet latency significantly impacts the execution time for requests with low slack. On the other hand, the network latency has less effect on the execution time for requests with high slack. Fig. 4 shows the cumulative distributions of the diversity in estimated slack for different applications in PARSEC [31] and AxBench [12]. The slack is estimated using the

This article has been accepted for publication in IEEE Transactions on Sustainable Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TSUSC.2022.3213469

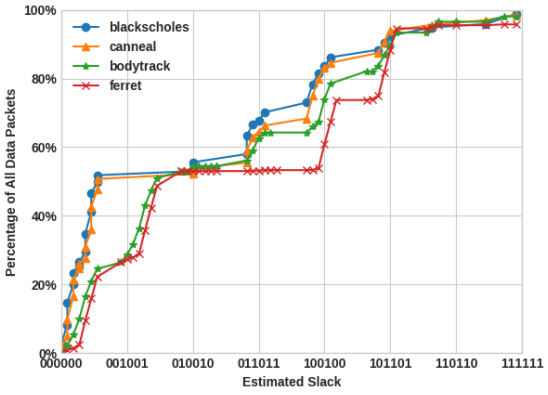


Fig. 4. Cumulative distribution of packets in terms of the estimated slack.

method proposed in [26] for a 4 × 4 2D mesh on-chip network and is represented with 6 bits in total for each packet. The two most significant bits (MSBs) represent the number of miss-predecessors, the third MSB represents the L2 cache hit/miss status, and the three least significant bits (LSBs) represent the number of hops that the packet needs to travel. For example, a slack value of 010010 indicates that: (1) the packet has one miss-predecessor, (2) the packet is predicted to be an L2 miss, and (3) it will take two hops for the packet to reach the destination. Low-slack packets (with slack values of less than 011111) account for more than 50% of the total data packets for blackscholes (73%), canneal (68%), bodytrack (64%), and ferret (53%). Regarding high-slack packets, existing works [25]–[27] indicate that although the number of high-slack packets is lower than that of low-slack packets, high-slack packets tend to be injected within a small amount of time. This traffic pattern increases the network latency for low-slack packets resulting in a longer execution time. Thus, the execution time of an approximate computing application can be further reduced if the approximate communication technique can accelerate low-slack packets.

3 Slack-Aware Packet Approximation

3.1 Overview

The slack-aware packet approximation technique is proposed to reduce the energy consumption of NoCs for sustainable parallel computation. The development of the proposed technique is based on the following observations.

Observation 1: Energy is the product of power and execution time; thus, the proposed packet approximation technique needs to reduce both values for energy-efficient on-chip communication.

Observation 2: Packet approximation is an effective technique to lower dynamic power consumption by reducing on-chip communication; however, existing approximation techniques achieve a small reduction in execution time with considerable overheads in terms of on-chip area and static power.

Observation 3: As shown in Fig. 4, low-slack packets account for most communication traffic. Since the network latency for these packets directly translates into a longer execution time, reducing the network latency for low-slack packets can shorten the execution time of an application.

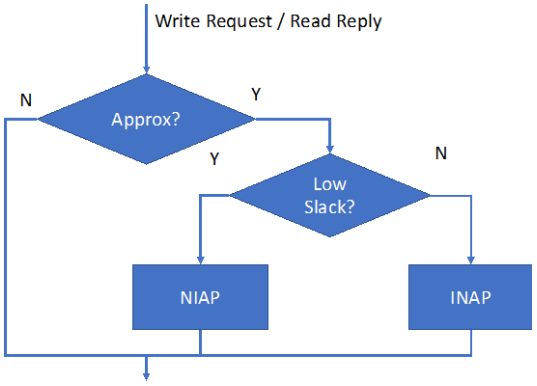


Fig. 5. Overview of the slack-aware packet approximation process.

In the proposed technique, two packet approximation mechanisms (i.e., NIAP and INAP) and a slack-aware control policy are developed to reduce both power dissipation and execution time. Specifically, the slack-aware control policy identifies low-slack packets and activates appropriate packet approximation mechanisms to save execution time. The NIAP mechanism approximates low-slack packets before the packet encoding process to reduce the latency of network links and switch traversals; the INAP mechanism approximates high-slack packets in the network to prioritize low-slack packets during the arbitration process in the router. Fig. 5 shows an overview of the proposed packet approximation process. It first checks whether the write request or read reply can be approximated according to the approximation information. Then, it chooses one packet approximation mechanism according to the slack. To identify low-slack packets, the performance characteristics of both mechanisms are studied, and a slack-aware control policy is then developed according to the performance characteristics of the two packet approximation mechanisms.

To implement the proposed packet approximation technique, an approximate NI and an approximate router are proposed. The approximate NI processes error-resilient data packets according to the approximation level and data type for both NIAP and INAP by utilizing a data truncation method with low area and static power overheads. The approximate NI also includes a slack-aware controller to choose between NIAP and INAP for effectively accelerating low-slack packets. The proposed approximate router prioritizes low-slack packets by approximating high-slack packets.

3.2 Packet Approximation Mechanisms

3.2.1 Network Interface Approximation (NIAP) Mechanism

The goal of designing the NIAP mechanism is to reduce the latency of network links and switch traversals for low-slack packets. To achieve this, a lightweight data truncation method is developed to reduce the data size for approximable low-slack packets. The NIAP mechanism truncates data according to the given approximation information, including the approximation level for the corresponding variable and the variable type (e.g., int or float). Table 1 elucidates the relationship between the data error threshold and the approximation level for both floating-point and integer data. Eleven approximation levels ranging from level 0 to level 10 are

TABLE 1
Relationship Between the Data Error Threshold and the Approximation Level.

Data Error Threshold	Approximation Level
0.125	10
0.03125	9
0.0078125	8
0.001953125	7
0.000488281	6
0.00012207	5
3.05176E-05	4
3.05176E-05	3
7.62939E-06	2
4.76837E-07	1
0	0

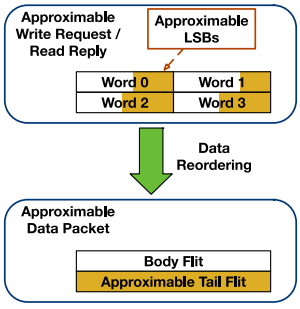


Fig. 6. Data reordering example.

supported by the proposed data approximation method. Level 0 corresponds to data that are transmitted with 100% accuracy, while level 10 corresponds to data that can tolerate a 12.5% relative error. The data error threshold is defined in Eq. 1, where $\tilde{a}$ is the approximated version of a and E_{ra} is the relative error.

$$E_{ra} = \frac{|a-\tilde{a}|}{a} \leq data\ error\ threshold \quad (1)$$

Based on the IEEE 754 standard [28], a single-precision floating-point value is defined as shown in Eqs. 2 and 3.

$$\text{float} = (-1)^{S} \times \text{mantissa} \times 2^{\text{exp}} \quad (2)$$

$$mantissa = 2^0 + \sum_{k=1}^{23} X_k 2^{-k}\ (X_k = 1\ or\ 0) \quad (3)$$

According to Eqs. 2 and 3, the mantissa always starts with a one. Thus, the first bit of the mantissa is omitted when a data point is represented in the floating-point format based on the IEEE 754 standard [28]. It is observed that when c bits (of the 23-bit mantissa) are protected, the maximum relative error on this floating-point data value will be $\sum_{k=c+1}^{23} 2^{-k}$, which is less than 2^{-c} according to the sum of the geometric sequence ($\sum_{k=1}^{n} ar^{k-1} = a(1-r^n)/1-r$, where a is the first term, n is the number of terms, and r is the common ratio in the sequence). Therefore, using Eq. 3, we can deduce the following expression (Eq. 4) for the data error threshold.

$$error\ threshold = 2^{-n} (1 \leq n \leq 23) \quad (4)$$

In Eq. 4 above, the data error threshold is a number between 0 and 1, and n is the number of most significant bits (MSBs) in the mantissa of this floating-point value. In a floating-point data value, the 1-bit sign and the 8-bit exponent (a total of 9 bits) are also critical bits that must be transmitted. Thus, when 23 – n bits are truncated, the relative error on the value will be less than 2. For example, to satisfy an approximation level of 9 for any floating-point value, we can truncate 18 LSBs, resulting in a maximum relative error of 3.12%. In this case, the NIAP mechanism reduces floating-point data size by 56% when the approximation level is 9.

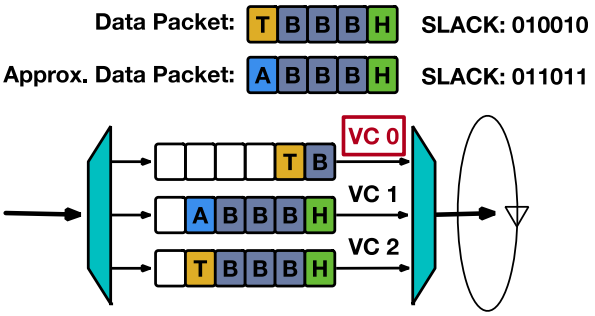


Fig. 7. An example of packet approximation during VC allocation.

The representation of a signed integer is illustrated in Eq. 5, where k is the bit position and X_k is the bit value at position k. In a signed integer, the MSB represents the sign, and the remaining 31 bits represent the value.

$$int = \sum_{k=0}^{31} X_k 2^k (X_k = 0\ or\ 1) \quad (5)$$

When n bits (of the 31 LSBs) are truncated, the maximum error caused by truncation will be $\sum_{k=0}^{n} X_k 2^k (X_k = 0\ or\ 1)$. Thus, Eq. 6 is derived to calculate the number of bits (n) to be truncated during the approximation process for a given error threshold.

$$error\ threshold = \frac{\sum_{k=0}^{n} X_k 2^k}{\sum_{k=0}^{31} X_k 2^k} (X_k = 0\ or\ 1) \quad (6)$$

For example, suppose that an integer 548320 (hexadecimal representation: 0x00085DE0) is to be approximated with approximation level = 9. According to Eq. 6, 14 LSBs are truncated and the truncated integer in hexadecimal representation is 0x00021. In this case, the NIAP mechanism reduces the integer data size by 44% when the approximation level is 9.

3.2.2 In-Network Approximation (INAP) Mechanism

The goal of designing the INAP mechanism is to reduce the arbitration latency in each router for low-slack packets by approximating high-slack packets. To achieve this, a two-step packet approximation process is developed.

1) The NI reorders the bits in an approximable packet according to the approximation information before injecting it into the network. The bit reordering process separates the data into critical bits and approximable bits for integer and floating-point values according to the approximation level. The INAP mechanism uses the same process as the NIAP mechanism to calculate the number of approximable LSBs according to the approximation level. Different from the NIAP mechanism, which discards the approximable LSBs, the INAP mechanism places the approximable bits at the end of the packet. For example, Fig. 6 shows an example of a data reordering process in the NI. The NI places the approximable LSBs in the tail flit of an approximable data packet. These approximable flits, which are filled with approximable LSBs, can be discarded by the router to prioritize the low-slack packets.

2) After the approximable packets have been injected into the network, an approximate router can discard the approximable tail flits for high-slack packets to accelerate the

This article has been accepted for publication in IEEE Transactions on Sustainable Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TSUSC.2022.3213469

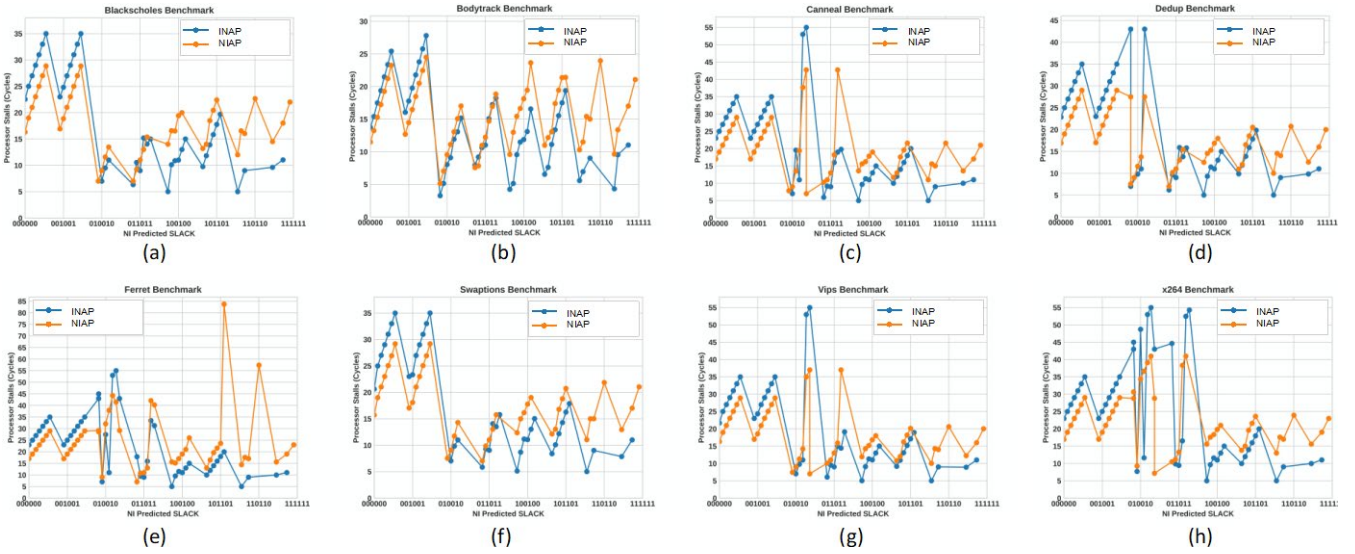


Fig. 8. Processor stalls versus NI-predicted slack.

arbitration process for low-slack packets in the router. To achieve this, the VC allocation and switch allocation processes are modified to prioritize packets with low slack values. During VC allocation, if a packet has a smaller slack value than one or more approximable packets, the VC allocator drops all approximable flits for the packets that are buffered in the VC. For illustration, Fig. 7 shows an example in which the packets in VC 1 and VC 2 are waiting for VC allocation, VC 0 is sending a data packet, and the slack values for the packets in VC 1 and VC 2 are 010010 and 011011, respectively. According to round-robin arbitration, the next packet to be sent is the data packet with an approximable tail flit in VC 1. Since the packet in VC 2 has a lower slack value than the approximable packet in VC 1, the VC allocator switches to VC 2 after the fourth flit of the approximable packet is finished. The approximable tail flit in VC 1 is discarded after the VC allocator switches to VC 2. In this way, the network latency for the low-slack packet is reduced by one cycle during VC allocation. The same design is applied for switch allocation. When two or more packets are requesting to use the same output port, and one packet has a lower slack value than other packets, the router drops all approximable flits in front of that packet.

3.2.3 Slack-Aware Control Policy

The INAP and NIAP mechanisms accelerate low-slack packets using a lightweight data approximation method to reduce the power consumed during on-chip communication and data approximation. To effectively reduce energy consumption, a control policy is needed for identifying low-slack packets in order to reduce the network latencies that impact the execution time.

The NIAP reduces the packet size at the NI by truncating approximable data. When the router forwards an NI-approximated packet, fewer cycles are needed for switch and link traversal than for the original data packet due to the reduced packet length. Since the packet needs to win two arbitrations (VC allocation and switch allocation) before traversing the switch and link, the time spent waiting for an available port cannot be reduced. On the other hand, the INAP reduces the wait time for arbitration for packets with low slack values. During the execution of a parallel computing application, packets with different levels of slack are injected simultaneously. To effectively reduce the execution time, finding a suitable slack threshold value to define low-slack packets is the key. In this case, the NIAP can be used for packets with low slack values to reduce the time for switch and link traversal. The wait time for arbitration for a packet with a low slack value can be further reduced by utilizing the INAP. To successfully reduce the wait time for arbitration, the INAP mechanism needs the other packets, which are waiting for the arbitration, to be reordered before injection. Considering that packets with high slack are injected in large numbers within a short period of time [25]–[27], applying INAP to a packet with high slack can increase the probability of a low-slack packet being prioritized. Thus, the execution time can be significantly reduced if low-slack and high-slack packets are approximated using NIAP and INAP, respectively.

To determine the slack threshold value to define low-slack packets, the two approximation mechanisms are each applied to an NoC separately, and the stall cycles experienced due to the processor waiting for packets are measured. Fig. 8 shows the relationships between the slack and the processor stall cycles observed under the two packet approximation mechanisms when the PARSEC benchmarks are executed on a multicore system. The NIAP reduces the number of cycles needed for switch and link traversal, with limited reduction in the number of cycles needed for the arbitration. Thus, for packets with high slack values requiring intensive traffic arbitration, the network latency is significantly increased compared to that using the INAP mechanism leading to processor stalls. For example, the intense traffic injected by the ferret benchmark (Fig. 8(e)) causes significant processor stalls (83 cycles) during times of high slack (101110) with the NIAP. In contrast, the INAP achieves significant improvement when the injected packets have high slack values. Since packets with high slack values are injected in large numbers within a short period of time, the INAP mechanism can effectively reduce the network latency by reducing the number of cycles needed for the arbitration. For example, the INAP reduces the processor stalls to 20 cycles, which is 76% fewer than in the case of the NIAP for the ferret benchmark (Fig. 8(e)) under

This article has been accepted for publication in IEEE Transactions on Sustainable Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TSUSC.2022.3213469

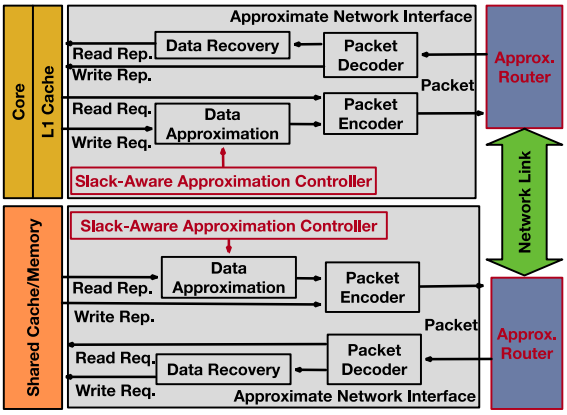


Fig. 9. Slack-aware approximate NI design.

high slack (101110). On the other hand, the INAP causes the network latency to increase compared to that using the NIAP mechanism for packets with low slack values. Since the INAP mechanism only reorders the packets, more cycles are needed for switch and link traversal, resulting in more processor stalls compared to the NIAP. By further analyzing the results, it can be seen that when the slack is less than 100000, the NIAP results in fewer processor stall cycles than the INAP. Thus, the slack-aware control policy should select the packet approximation mechanisms to be applied according to the slack threshold. If the slack is less than 100000 (slack threshold value), the NIAP is used; otherwise, the INAP mechanism is activated.

4. The Implementation of Slack-Aware Packet Approximation

4.1 Slack-Aware Approximate Network Interface (NI) Design

Fig. 9 shows the proposed approximate NI design. The approximate NI includes a slack-aware approximation controller, the data approximation logic consisting of data truncation and data reordering mechanisms, and the data recovery logic.

4.1.1 Data Approximation Logic Design

The data approximation logic supports the data truncation applied in NIAP and the data reordering required by INAP. Fig. 10. illustrates the implementation of the proposed design.

Fig. 10(a) illustrates the data truncation process when the NI approximates a floating-point value (NIAP). First, the demultiplexer separates the sign bit and exponent bits from the floating-point value. Then, the approximation logic truncates the LSBs of the mantissa according to the approximation level. Afterward, the sign and exponent bits are added to the MSBs of the truncated mantissa. Finally, the approximated floating-point value and slack value are sent to the packet encoder.

Fig. 10(b) illustrates the data truncation process when the NI approximates an integer value (NIAP). First, the data truncation module eliminates the LSBs of the integer value to be approximated based on the approximation level. Then, the approximated integer and slack value are sent to the packet encoder.

When the INAP mechanism is activated, the NI only reorders the bits of the request/reply packet according to the approximation level and data type if the request/reply packet can

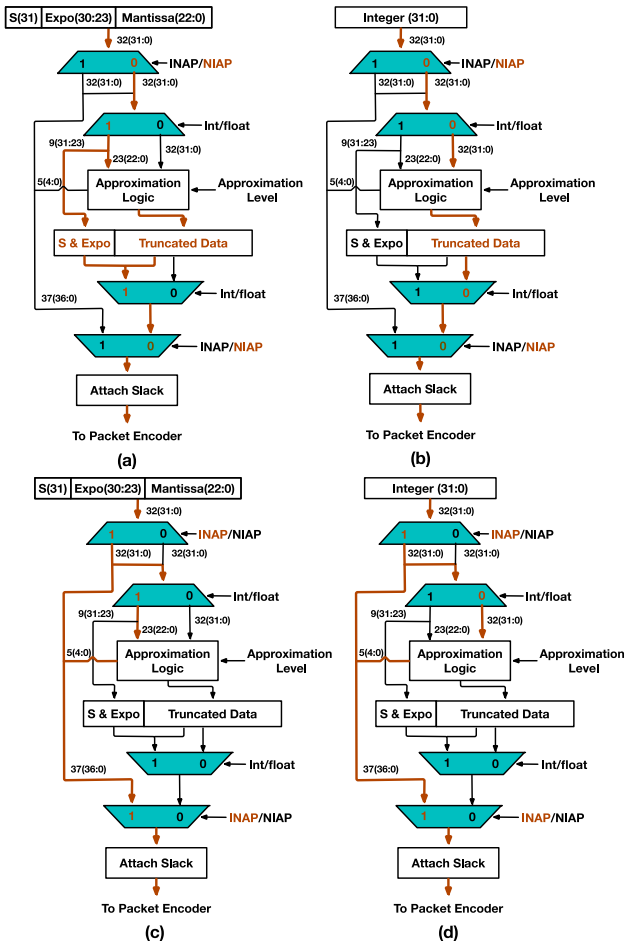


Fig. 10. Data approximation logic design.

tolerate some level of error. Fig. 10(c) and (d) illustrate the first step of the data reordering process for floating-point and integer values, respectively. The approximation logic calculates the number of approximable LSBs according to the approximation information. The original data, the number of approximable LSBs, and the slack value are sent to the packet encoder. Then, the packet encoder places the calculated number of approximable LSBs in the last flit of the packet based on the calculated number. After data reordering, the packet is the same size as the original data packet. The only difference is that the approximable LSBs are placed in the tail flit.

In contrast to the NIAP mechanism (Fig. 10(a) and (b)), the INAP mechanism (Fig. 10(c) and (d)) bypasses the registers to store the truncated data, thus eliminating a cycle needed for data approximation at the NI. Moreover, NIAP and INAP use the same approximation logic to translate approximation levels for packet approximation. Thus, the proposed implementation saves area and static power compared to lossy-compression-based packet approximation techniques [4], [15], [16], [19].

4.1.2 Data Recovery Logic Design

Fig. 11 shows the data recovery logic for approximated packets. The data recovery process consists of three steps. First, the demultiplexer separates the approximated data from accurate data based on approximation indicators. The approximation indicator for a packet is generated by checking the approximation information carried by that packet. If the

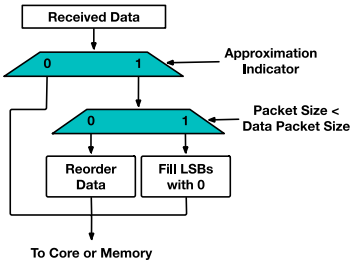


Fig. 11. Data recovery logic design.

approximation level is larger than 0, the approximation indicator is set to 1. Otherwise, the approximation indicator is set to 0. Then, the data recovery logic checks the size of each approximated data packet. If the packet was approximated using INAP or NIAP, the packet size would be less than the size of a normal data packet. Thus, the logic recovers the data by filling in the truncated LSBs with zeros. If the packet size is equal to the size of a normal data packet, this indicates that the packet was not approximated by the router using the INAP mechanism. Therefore, the logic reorders the received data, which have not been approximated in the network, according to the approximation information. Finally, the recovered data are sent to the core or memory.

4.1.3 Slack-Aware Approximation Controller Design

The approximation controller is designed to dynamically select the appropriate packet approximation mechanism based on slack. This controller has two functionalities: slack calculation and approximation mechanism selection. For slack calculation, the method proposed in [26] is used. In this paper, the slack is represented by three sets of bits for each packet. The two highest bits represent the number of miss-predecessors, the third-highest bit represents the L2 cache hit/miss status, and the rest of the LSBs represent the number of hops that the packet needs to travel, which depends on the network topology. The approximation mechanism compares the packet slack against the threshold slack value(100000) for a 4 × 4 2D mesh topology. If the packet slack is lower than the threshold, data truncation (NIAP) is selected to approximate the data. Otherwise, data reordering (INAP) is selected. For larger network topologies (e.g., 8 × 8 2D mesh), the length of slack value can be increased to accommodate larger numbers of hops. The threshold slack values for these networks can be determined using the same method proposed in Section 3.2.3.

4.2 Approximate Router Design

The approximate router prioritizes packets with low slack values by dropping approximable flits during the arbitration processes of the VC and switch allocation stages. Thus, the VC and switch allocation stages are modified as follows. First, a slack comparator is added to the router to identify packets with small slack values. Second, the buffer reading process is modified to identify and discard approximable flits to prioritize the low-slack packets.

5 Evaluation

5.1 Experimental Methodology

We evaluate the performance of the proposed slack-aware packet approximation technique using the GEM5 simulator [29]. The simulator is modified to support slack-aware packet approximation. The data approximation logic and data recovery logic are implemented in the NI, and the router design is modified to support the NIAP, INAP, and the slack-aware control policy. The NoC used for evaluation shares the same specifications as the NoC in Section 3.2.3. Table 2 shows the detailed settings used for the GEM5 simulator. DSENT [30] is used to estimate the power consumption of the NoC. The applications in the AxBench [12] and PARSEC [31] benchmark suites are used as workloads. Table 3 lists the benchmarks used and the evaluation metrics utilized to measure the result quality.

TABLE 2
Simulation Environment Setup

NoC Parameters	Network type: Garnet 2.0 Topology: 4 × 4 2D mesh Data packet size: 5 flits Link width: 128 bits Routing algorithm: X-Y routing Flow control: wormhole switching
System Parameters	16 on-chip cores 32 kB L1 instruction cache 32 kB L1 data cache 4-way associative 16-bank fully shared 16 MB L2 cache
Approximate Communication Techniques	Approximate Communication Framework(ACF) [16]; Approx-NoC [4]; AxBA [15]; Slack-Aware (proposed)
Conventional Energy-Saving Technique	DVFS [22]

The proposed technique is evaluated by comparing it against a state-of-the-art approximate communication framework(ACF) [16], Approx-NoC [4], and AxBA [15] from four perspectives: execution time, power consumption, energy consumption, and result error. The DVFS [22] and the baseline (i.e., a NoC without approximation or DVFS) are also evaluated for comparison. Since packets are transmitted with full accuracy for DVFS and the baseline NoC, these techniques suffer no result error.

The state-of-the-art ACF technique [16] includes an NI packet approximation technique based on frequent patterns and data truncation. Approx-NoC [4] includes an approximated frequent-pattern compression technique. AxBA [15] includes a base-delta data approximation technique. To fairly compare the performance of these different packet approximation techniques, all approximate computing applications are annotated using the state-of-the-art learning-based quality control method [17] with a target result quality of 95%. All approximate communication-based techniques are simulated with a 1.1 V supply voltage and a 2.0 GHz clock frequency. The DVFS technique partitions NoC into sixteen voltage/frequency domains. Each domain contains one router. The DVFS supports total of 5 voltage/frequency levels, including 0.8 V @ 1 GHz, 0.85 V @ 1.25 GHz, 0.9 V @ 1.5 GHz, 1.0V @ 1.8 GHz, and 1.1 V @ 2.0 GHz. The controller in each domain chooses one voltage/frequency setting according to the traffic intensity for each router.

This article has been accepted for publication in IEEE Transactions on Sustainable Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TSUSC.2022.3213469

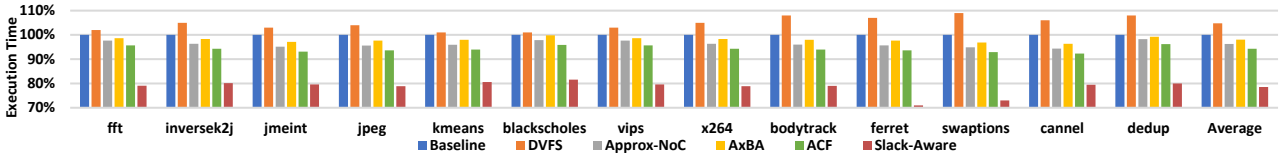


Fig. 12. Execution times of the applications under DVFS, Approx-NoC, AxBA, ACF, and Slack-Aware (proposed) compared to the baseline.

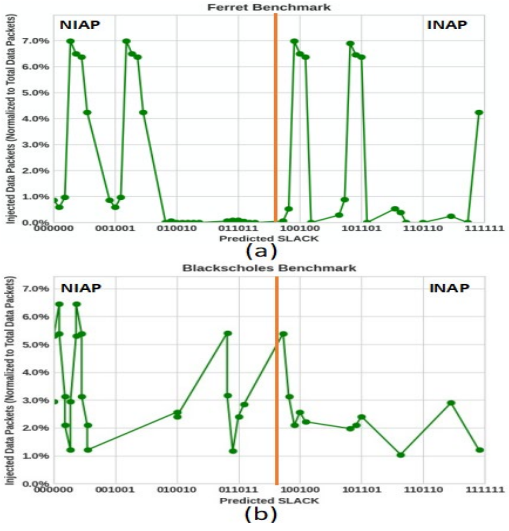


Fig. 13. Injected data packets versus predicted slack. The results are normalized with respect to the total injected data packets for the corresponding benchmark.

5.2 Execution Time

Fig. 12 shows the evaluation results for the execution time normalized with respect to the baseline. We compare the proposed slack-aware packet approximation technique with ACF, Approx-NoC, AxBA, and DVFS. In ACF, Approx-NoC, and AxBA, every annotated error-resilient variable is approximated in the NI. Through the use of a frequent-pattern compression technique, ACF and Approx-NoC achieve smaller sizes for error-resilient packets than AxBA. Thus, the ACF and Approx-NoC improve the execution time by 2% and 4%, respectively, compared to AxBA. The DVFS incurs a longer network latency, resulting in a 5% average increase in execution time compared to the baseline. In contrast, the proposed technique achieves a 20% reduction in the execution time on average compared to AxBA. Compared to the state-of-the-art technique (ACF), the proposed technique improves the execution time by 17% on average. Compared to DVFS, the proposed technique improves the execution time by 26% on average. The largest network latency reduction in the experiment is achieved for the ferret benchmark (24% reduction compared to ACF). The smallest network latency improvement is obtained for blackscholes (14% reduction compared to ACF). The significant improvement in the execution time is achieved due to the slack awareness of the proposed technique. The INAP mechanism significantly reduces the latency of arbitration for low-slack packets during transmission. At the same time, the NIAP mechanism, similar to ACF, Approx-NoC, and AxBA, reduces the latency of the link and switch traversal for low-slack packets. The combined effect of the two packet approximation techniques significantly reduces the latency for low-slack packets and the execution time.

Fig. 13 shows the distributions of the number of injected data packets relative to the slack for the ferret and blackscholes benchmarks. The ferret benchmark has more packets for which the INAP is used (47%) than the blackscholes benchmark does (27%). Compared to existing techniques, the INAP mechanism further reduces the latency for low-slack packets during packet transmission. Because more packets are approximated using INAP for the ferret benchmark, the latency for low-slack packets in ferret can be further reduced, leading to a greater execution time improvement than blackscholes.

5.3 Power Consumption

Fig. 14 shows the evaluation results for the power consumption under the approximate communication techniques and DVFS, normalized with respect to the baseline. The power consumption includes two components: static power and dynamic power. The static power is dissipated by leakage currents that flow when the NoC is powered on. The dynamic power is dissipated when activating or deactivating gates during signal transitions.

Compared to ACF, the proposed slack-aware packet approximation technique achieves an 18% reduction in power consumption on average. The largest power consumption reduction in the experiment is achieved for the ferret benchmark (25% reduction compared to ACF). The smallest power consumption improvement is obtained for blackscholes (16% reduction compared to ACF). The significant improvement in power consumption is achieved due to not only the slack awareness of the proposed technique but also the lightweight packet approximation mechanisms. Compared to ACF, Approx-NoC, and AxBA, the proposed technique uses a lightweight data truncation process to approximate data in the packets consuming less static power with a similar compression rate. In contrast, ACF, Approx-NoC, and AxBA use lossy data compression, which incurs area and static power overheads during approximation. Section 4.5 compares the area and static power overheads in detail. Compared to DVFS, the proposed method achieves a similar reduction in power consumption on average. However, due to the higher rate of packet approximation for several benchmarks, such as ferret (82% of the data packets are approximated, according to Fig. 17), the proposed method achieves 7% more power savings than DVFS. This shows that the proposed method is effective in saving power.

5.4 Energy Consumption

Fig. 15 shows the NoC energy consumption normalized with respect to the baseline. Energy consumption is defined as the product of power and execution time. Since the proposed slack-aware packet approximation technique achieves reductions in both execution time and power consumption, a significant reduction in energy consumption is expected.

This article has been accepted for publication in IEEE Transactions on Sustainable Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TSUSC.2022.3213469

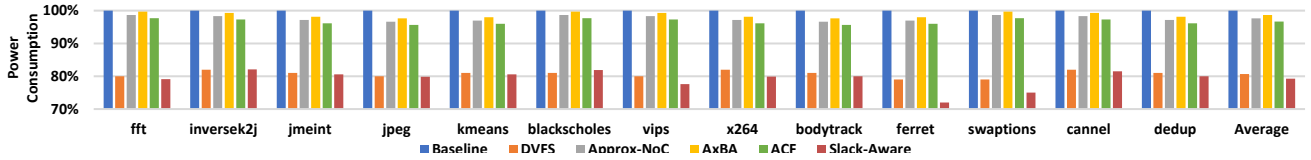


Fig. 14. NoC power consumption under DVFS, Approx-NoC, AxBA, ACF, and Slack-Aware (proposed) compared to the baseline.

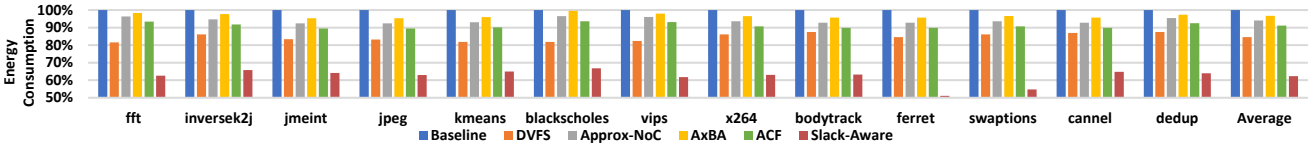


Fig. 15. NoC energy consumption under DVFS, Approx-NoC, AxBA, ACF, and Slack-Aware (proposed) compared to the baseline.

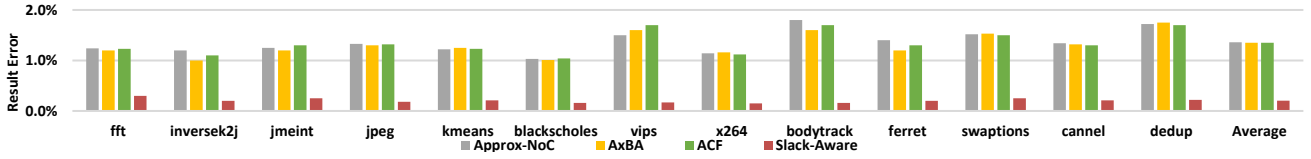


Fig. 16. Result error with a target error of 5%

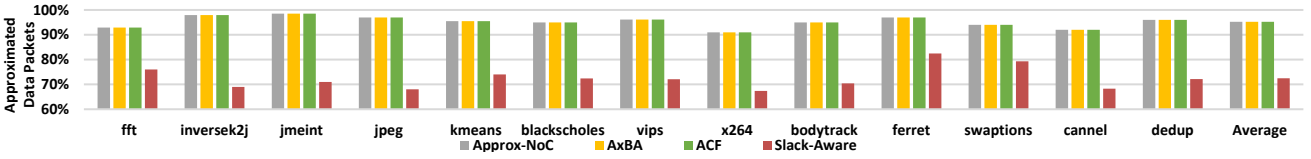


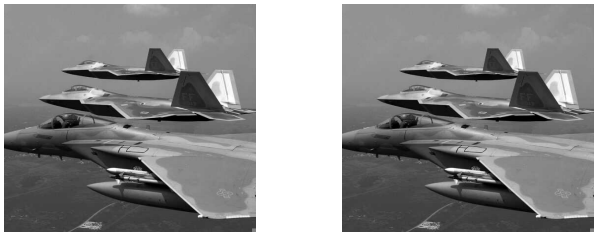
Fig. 17. Approximated data packets.

Compared to ACF, the proposed technique achieves a 28% reduction in energy consumption on average. The largest energy reduction is achieved for the ferret benchmark (38% reduction compared to ACF). The smallest power consumption improvement is obtained for blackscholes (25% reduction compared to ACF). Compared to DVFS, the proposed technique achieves a 22% reduction in energy consumption on average. This shows that the proposed method is effective in reducing NoC energy consumption.

5.4 Result Quality

Table 3 provides the metrics for measuring the applications' result quality. The result errors for all the benchmarks running on an approximate-communication-enabled NoC are shown in Fig. 16. The applications have a target result quality of 95%, which is equivalent to a tolerable result error of less than 5%. This figure shows that ACF achieves a 1.3% result error on average, whereas the proposed technique achieves a 0.2% result error on average. Fig. 17 shows the numbers of data packets approximated when different packet approximation techniques are used.

The proposed technique approximates 24% fewer data packets on average than ACF, Approx-NoC, and AxBA. Since the INAP mechanism drops flits based on the slack value, not all error-resilient variables are approximated, leading to an improvement in the result error. For example, the x264 benchmark suffers the least quality loss (0.15%) in the result and has the lowest percentage of approximated data packets (67%) among the benchmarks. For further illustration, the accurate results for the jpeg benchmark are compared against the approximate results obtained using the proposed technique in Fig. 18. The difference between the two outputs is negligible and unrecognizable to the human eye. These results indicate that the proposed approximate communication technique limits the result error within an acceptable amount.



(a) Accurate Result (b) Approximate Result

Fig. 18. Comparison of results for the jpeg benchmark.

5.5 Overhead Analysis

We implemented the proposed technique with 16 cores using Verilog to evaluate the area, static power, and latency overheads. The proposed technique was synthesized with 32 nm technology using the Synopsys Design Vision software. The synthesis results show that the technique incurs area overheads of 1.56 μm^2 and 0.4 μm^2 for each NI and router, respectively. The overall area overhead for a 4 × 4 2D mesh is less than 1% of the total NoC area. When the supply voltage is 1.1 V, the proposed technique incurs a static power overhead of 5.6 mW for the whole NoC. Compared to the existing approximate communication techniques (e.g., ACF, Approx-NoC, and AxBA), the proposed packet approximation technique requires fewer hardware components to reduce the data size and recover data in the NI. The synthesis results show that the proposed technique reduces the on-chip area by an average of 52% and the static power overhead by 64% compared to the existing techniques for data approximation.

Regarding the latency overhead, we find that data truncation at the NI requires one cycle. Thus, one cycle is added for a write request packet or a read reply packet only when the NIAP mechanism is activated.

TABLE 3
Parallel Computing Application Specifications

Benchmark Suit	Benchmark	Query Data Size	Result Quality Evaluation Metric
AxBench [12]	fft	5k random floating-point numbers	Relative error
	inversek2j	100k random (x,y) points	Relative error
	jmeint	10k pairs of 3D triangles	# of mismatches
	jpeg	512 * 512-pixel image	Average pixel difference
	kmeans	512 * 512-pixel image	Average pixel difference
PARSEC [31]	blackscholes	4096 options	Relative error
	vips	1600 * 1200 pixel image	Average pixel difference
	bodytrack	4 cameras, 1 frame, 1000 particles, 5 annealing layers	Relative error
	x264	640 * 360 pixels, 8 frames	Average pixel difference/frame
	ferret	16 image queries, database with 3544 images	Relative error
	swaptions	16 swaptions, 5000 simulations	Relative error
	canneal	2000 start temperature, 100000 netlist elements	Relative error
	dedup	10 MB data	Bit difference

5.6 Sensitivity Study

We show the sensitivity of the proposed technique to different network topologies. The number of bits for representing the slack is increased for the larger network topology due to a larger hope count. Thus, the slack threshold used by the slack-aware control policy needs to be adjusted as per different topologies. Table 4 shows the slack threshold for different network topologies when the procedure described in Section 3.2.3 is used to determine the slack threshold. By comparing the thresholds for different topologies, the three MSBs, which represent the number of miss-predecessors and L2 cache hit/miss status, are the same for different topologies. The value for hope count is increased for the larger networks. The increase in the value covers the packets with a greater hop count in large networks. More importantly, the increase in threshold value maintains the ratio between low-slack packets and high-slack packets when the proposed technique is applied to a larger network. This ensures a sufficient number of packets being accelerated by both INAP and NIAP in larger networks for reductions in execution time and power dissipation. Therefore, the proposed method is applicable to larger network topologies and can maintain substantial energy savings.

TABLE 4
Threshold Slack for Different Network Topologies

Network Topology	4 × 4 2D Mesh	5 × 5 2D Mesh	6 × 6 2D Mesh
Threshold	100000	1000010	1000100
Number of Slack Bits	6 bits	7 bits	7 bits

6 Conclusion

In this work, we propose a slack-aware packet approximation technique for NoCs to reduce the energy consumption of the NoCs for sustainable parallel computing. The proposed technique includes a slack-aware control policy and two new packet approximation mechanisms, namely, an in-network approximation (INAP) mechanism and a network interface approximation (NIAP) mechanism. The NIAP mechanism reduces the latency of switch and link traversal for low-slack packets by reducing the data size at the network interface. The INAP mechanism reduces the latency for packets with a low slack by prioritizing these packets in the arbitration phases for switch allocation and VC allocation. The slack-aware control policy identifies low-slack packets in order to achieve a shorter execution time. An approximate network interface and an approximate router are developed to support the slack-aware packet approximation technique. The proposed technique reduces the power consumption of the NoC while improving the execution time and lowering the quality loss for applications. Our detailed evaluation shows that the proposed slack-aware packet approximation technique reduces the execution time by up to 24% and the energy consumption by up to 38% compared to existing approximate communication techniques with 1.1% lower result error on average.

Acknowledgment

This research is supported by NSF grants CCF-1953961, CCF-1812467, CCF-1812495 and CCF-1953980.

References

[1] W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in Proceedings of the 38th annual Design Automation Conference, New York, NY, USA, Jun. 2001, pp. 684–689.

[2] T. Bjerregaard and S. Mahadevan, "A survey of research and practices of Network-on-chip," ACM Comput. Surv., vol. 38, no. 1, pp. 1-es, Jun. 2006.

[3] F. Betzel, K. Khatamifard, H. Suresh, D. J. Lilja, J. Sartori, and U. Karpuzcu, "Approximate Communication: Techniques for Reducing Communication Bottlenecks in Large-Scale Parallel Systems," ACM Comput. Surv., vol. 51, no. 1, p. 1:1-1:32, Jan. 2018.

[4] R. Boyapati, J. Huang, P. Majumder, K. H. Yum, and E. J. Kim, "APPROX-NoC: A Data Approximation Framework for Network-On-Chip Architectures," in Proceedings of the 44th Annual International Symposium on Computer Architecture, Toronto, ON, Canada, Jun. 2017, pp. 666–677.

[5] M. B. Taylor et al., "The Raw microprocessor: a computational fabric for software circuits and general-purpose programs," IEEE Micro, vol. 22, no. 2, pp. 25–35, Mar. 2002.

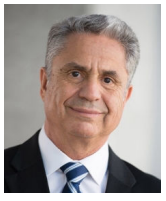
[6] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar, "A 5-GHz Mesh Interconnect for a Teraflops Processor," IEEE Micro, vol. 27, no. 5, pp. 51–61, Sep. 2007.

[7] H. Wang, L.-S. Peh, and S. Malik, "Power-driven design of router microarchitectures in on-chip networks," in Proceedings. 36th Annual IEEE/ACM International Symposium on Microarchitecture, 2003. MICRO-36., Dec. 2003, pp. 105–116.

[8] H. K. Mondal, S. H. Gade, S. Kaushik, and S. Deb, "Adaptive Multi-Voltage Scaling with Utilization Prediction for Energy-Efficient Wireless

NoC," IEEE Transactions on Sustainable Computing, vol. 2, no. 4, pp. 382–395, Oct. 2017.
[9] A. Karanth et al., "Sustainability in Network-on-Chips by Exploring Heterogeneity in Emerging Technologies," IEEE Transactions on Sustainable Computing, vol. 4, no. 3, pp. 293–307, Jul. 2019.
[10] K. Wang, H. Zheng, Y. Li, and A. Louri, "SecureNoC: A Learning-enabled, High-performance, Energy-efficient, and Secure On-chip Communication Framework Design," IEEE Transactions on Sustainable Computing, pp. 1–1, 2021.
[11] S. Mittal, "A Survey of Techniques for Approximate Computing," ACM Comput. Surv., vol. 48, no. 4, p. 62:1-62:33, Mar. 2016.
[12] A. Yazdanbakhsh, D. Mahajan, H. Esmaeilzadeh, and P. Lotfi-Kamran, "AxBench: A Multiplatform Benchmark Suite for Approximate Computing," IEEE Design Test, vol. 34, no. 2, pp. 60–68, Apr. 2017.
[13] Y. Chen and A. Louri, "An online quality management framework for approximate communication in network-on-chips," in Proceedings of the ACM International Conference on Supercomputing, Phoenix, Arizona, Jun. 2019, pp. 217–226.
[14] Y. Chen, M. F. Reza, and A. Louri, "DEC-NoC: An Approximate Framework Based on Dynamic Error Control with Applications to Energy-Efficient NoCs," in 2018 IEEE 36th International Conference on Computer Design (ICCD), Oct. 2018, pp. 480–487.
[15] J. R. Stevens, A. Ranjan, and A. Raghunathan, "AxBA: an approximate bus architecture framework," in Proceedings of the International Conference on Computer-Aided Design, San Diego California, Nov. 2018, pp. 1–8.
[16] Y. Chen and A. Louri, "An Approximate Communication Framework for Network-on-Chips," IEEE Transactions on Parallel and Distributed Systems, vol. 31, no. 6, pp. 1434–1446, Jun. 2020.
[17] Y. Chen and A. Louri, "Learning-Based Quality Management for Approximate Communication in Network-on-Chips," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 39, no. 11, pp. 3724–3735, Nov. 2020.
[18] L. Wang, X. Wang, and Y. Wang, "ABDTR: Approximation-Based Dynamic Traffic Regulation for Networks-on-Chip Systems," in 2017 IEEE International Conference on Computer Design (ICCD), Nov. 2017, pp. 153–160.
[19] V. Fernando, A. Franques, S. Abadal, S. Misailovic, and J. Torrellas, "Replica: A Wireless Manycore for Communication-Intensive and Approximate Data," in Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, New York, NY, USA, Apr. 2019, pp. 849–863.
[20] S. Xiao, X. Wang, M. Palesi, A. K. Singh, and T. Mak, "ACDC: An Accuracy- and Congestion-aware Dynamic Traffic Control Method for Networks-on-Chip," in 2019 Design, Automation Test in Europe Conference Exhibition (DATE), Mar. 2019, pp. 630–633.
[21] A. K. Mishra, R. Das, S. Eachempati, R. Iyer, N. Vijaykrishnan, and C. R. Das, "A case for dynamic frequency tuning in on-chip networks," in 2009 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), Dec. 2009, pp. 292–303.
[22] R. Hesse and N. E. Jerger, "Improving DVFS in NoCs with Coherence Prediction," in Proceedings of the 9th International Symposium on Networks-on-Chip, New York, NY, USA, Sep. 2015, pp. 1–8.
[23] L. Shang, L. Peh, and N. K. Jha, "Power-efficient Interconnection Networks: Dynamic Voltage Scaling with Links," IEEE Computer Architecture Letters, vol. 1, no. 1, pp. 6–6, Jan. 2002.
[24] Q. Fettes, M. Clark, R. Bunescu, A. Karanth, and A. Louri, "Dynamic Voltage and Frequency Scaling in NoCs with Supervised and Reinforcement Learning Techniques," IEEE Transactions on Computers, vol. 68, no. 3, pp. 375–389, Mar. 2019.
[25] R. Das, O. Mutlu, T. Moscibroda, and C. R. Das, "Application-aware prioritization mechanisms for on-chip networks," in 2009 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), Dec. 2009, pp. 280–291.
[26] R. Das, O. Mutlu, T. Moscibroda, and C. R. Das, "Aérgia: exploiting packet latency slack in on-chip networks," SIGARCH Comput. Archit. News, vol. 38, no. 3, pp. 106–116, Jun. 2010.
[27] Jia Zhan, N. Stoimenov, J. Ouyang, L. Thiele, V. Narayanan, and Yuan Xie, "Designing energy-efficient NoC for real-time embedded systems through slack optimization," in 2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC), May 2013, pp. 1–6.
[28] "IEEE Standard for Floating-Point Arithmetic," IEEE Std 754-2008, pp. 1–70, Aug. 2008.
[29] N. Binkert et al., "The gem5 simulator," SIGARCH Comput. Archit. News, vol. 39, no. 2, pp. 1–7, Aug. 2011.
[30] C. Sun et al., "DSENT - A Tool Connecting Emerging Photonics with Electronics for Opto-Electronic Networks-on-Chip Modeling," in 2012 IEEE/ACM Sixth International Symposium on Networks-on-Chip, May 2012, pp. 201–210.
[31] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC benchmark suite: characterization and architectural implications," in Proceedings of the 17th international conference on Parallel architectures and compilation techniques - PACT '08, Toronto, Ontario, Canada, 2008, p. 72.

Yuechen Chen (Member, IEEE) received his master's degree in Electrical Engineering from the George Washington University, Washington, District of Columbia, in 2016. He is currently pursuing Ph.D. with the Department of Electrical and Computer Engineering at the George Washington University. His research interests include approximate computing and NoCs.



Ahmed Louri (Fellow, IEEE) is the David and Marilyn Karlgaard Endowed Chair Professor of Electrical and Computer Engineering at the George Washington University, which he joined in August 2015. He received the Ph.D. degree in Computer Engineering from the University of Southern California, Los Angeles, California in 1988. From 1988 to 2015, he was a professor of Electrical and Computer Engineering at the University of Arizona. From 2010 to 2013, he served as a program director in the National Science Foundation's (NSF) Directorate for Computer and Information Science and Engineering. His research interests are interconnection networks and network on chips for multicores, and the use of machine learning techniques for energy-efficient, reliable, high-performance and secure many-core architectures and accelerators. He was recently selected to be the recipient of the IEEE Computer Society 2020 Edward J. McCluskey Technical Achievement Award. He is currently serving as the Editor-in-Chief of IEEE Transactions on Computers.

Shanshan Liu (Member, IEEE) received the Ph.D. degree in Microelectronics and Solid-State Electronics from Harbin Institute of Technology, Harbin, China, in 2018, and was a Post-doctoral faculty researcher with the Department of Electrical and Computer Engineering, Northeastern University, Boston, USA, from 2018 to 2021. She is currently an Associate Research Professor with the Klipsch School of Electrical and Computer Engineering, New Mexico State University, Las Cruces, USA. Her research interests include fault tolerance design in high performance computer systems, VLSI design, dependable machine learning, stochastic computing, error correction codes.

Fabrizio Lombardi (Fellow, IEEE) received the B.Sc. degree (Hons.) in electronic engineering from the University of Essex, U.K., in 1977, the master's degree in microwaves and modern optics and the Diploma degree in microwave engineering from the Microwave Research Unit, University College London, in 1978, and the Ph.D. degree from the University of London in 1982. He is currently the International Test Conference (ITC) Endowed Chair Professorship with Northeastern University, Boston, USA. His research interests are bio-inspired and nano manufacturing/computing, VLSI design, testing, and fault/defect tolerance of digital systems. He is currently the Vice President for Publications of the IEEE Computer Society, the 2021 President-elect of the IEEE Nanotechnology Council, and a member of the IEEE Publication Services and Products Board.