Approximate Communication Strategies for Energy-Efficient and High Performance NoC: Opportunities and Challenges

Md Farhadur Reza
farhadurreza@vt.edu
Department of Electrical & Computer Engineering
Virginia Polytechnic Institute and State University
Blacksburg, VA

Paul Ampadu
ampadu@vt.edu
Department of Electrical & Computer Engineering
Virginia Polytechnic Institute and State University
Blacksburg, VA

ABSTRACT

With the advancement and miniaturization of transistor technology, hundreds of cores can be integrated on a single chip. Network-on-Chips (NoCs) are the *de facto* on-chip communication fabrics for multi/many core systems because of their benefits over the traditional bus in terms of scalability, parallelism, and power efficiency. However, relative power consumption of NoC has been increasing with the increase in the number of cores on a chip, as communication costs much more time and energy than that of computation. Approximating computing concept can be applied in the NoC to reduce power consumption by approximating the communication data of emerging data-intensive applications, such as machine learning and big data analytics, that can tolerate errors. In this paper, we present an architecture for NoC approximation, and also provide preliminary results (which shows significant improvement) to support the importance of approximate communication solution for energy-efficient and high-performance NoC. Then we present various approximate communication solutions for reducing data movement in NoC. Finally, we discuss about the challenges and software and hardware overheads to adapt approximate technique in NoC, and also provide tentative solutions to address those challenges.

CCS CONCEPTS

• **Computer systems organization → Multicore architectures**; • **Networks → Network on chip**; • **Hardware → Network on chip**.

KEYWORDS

Approximate Communication; Error Tolerance/Threshold; Accuracy; Energy Consumption; Networks-on-Chip (NoCs)

ACM Reference Format:
Md Farhadur Reza and Paul Ampadu. 2019. Approximate Communication Strategies for Energy-Efficient and High Performance NoC: Opportunities and Challenges. In *Great Lakes Symposium on VLSI 2019 (GLSVLSI '19), May 9–11, 2019, Tysons Corner, VA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3299874.3319455

GLSVLSI '19, May 9–11, 2019, Tysons Corner, VA, USA

ACM ISBN 978-1-4503-6252-8/19/05…$15.00
https://doi.org/10.1145/3299874.3319455

1 INTRODUCTION

Multi-core systems are moving towards the integration of hundreds of cores on a single chip to accomplish emerging applications through low-power highly parallel computing. Network-on-Chips (NoCs) have been adopted as a viable solution to manage complex interconnection and provide energy efficient performance in multi/many core systems. For example, a multi-core architecture on NoC is illustrated in Figure 1 where heterogeneous processor tiles are placed in a 2D grid. Each tile contains a processor (with distinct computation power), its local cache, a slice of the shared last-level cache, and a router (configured with different communication capacity) for data and control transmission between the tiles via varying bandwidth links. In multi-core NoC architecture, data flows from L1 and L2-caches of one core to another core, core to memory controller for memory traffic, etc.

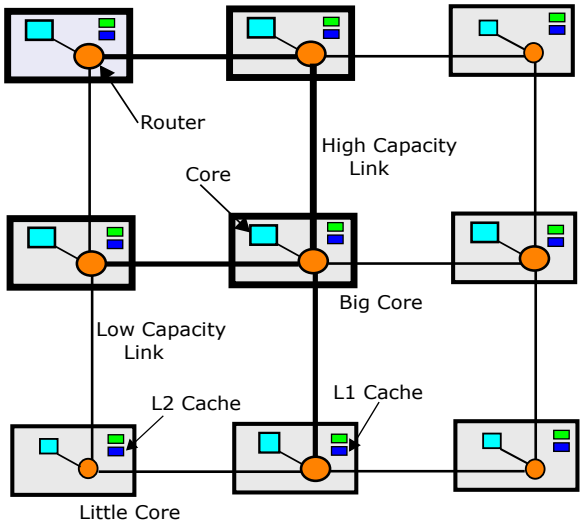

Figure 1: Multi-Core NoC Architecture

Computation has become more energy-efficient with the continued scaling of transistors. Billions of transistors can now be integrated on a single chip. However, interconnection energy efficiency/delay has not been scaled in the same way as that of transistor. For example, from 90nm to 7nm transistor technology, interconnection energy efficiency has been scaled down by only 1.6X compared to 6X scaling down of transistors (computation) [7]. Because of this disparity, energy consumption in the on-chip systems is increasingly being dominated by the interconnection delay [3].

NoC power consumption has been steadily increasing with the increase in the number of cores that are integrated on the chip. Research has shown that NoC power consumption can reach up to 30% of the overall chip power [14, 23, 24]. This high NoC power consumption challenge has led to many development in NoC architecture, for example, router architecture enhancement, adaptive routing algorithms, power/thermal-aware designs, etc.

Approximation, the focus of this paper, can reduce the communication workload to get low-power and high-throughput NoC. As the destination node does not need fully accurate data for computation, it presents an opportunity to reduce the data movement through NoC. There have been few works that focus on approximate communication (AComm) techniques [5, 8, 9, 25] for improving the communication efficiency in multi/many core systems. [5] defines the AComm as the application of approximate computing techniques to reduce the amount of communication between processing elements in large-scale parallel systems like multi/many core architectures. They explore the potential benefits of AComm for tackling the communication bottleneck issues in parallel multi-core systems by surveying three promising techniques : compression, relaxed synchronization, and value prediction.

The contributions of this paper are outlined below.

- We first present the motivation of AComm in NoC. Then we present NoC design challenges, and propose AComm technique to address those NoC challenges.
- We are presenting AComm technique to approximate the data dynamically based on the error tolerance (ET) of the applications, while different applications have different accuracy requirements. We present AComm architecture for the application of approximation in NoC. We also provide some empirical results to support the importance of AComm technique in NoC.
- We present several promising AComm solutions, such as packet prediction and error checking reduction, for NoC.
- We discuss the several challenges, such as error tolerance determination and software and hardware support, of AComm techniques for its application in NoC, and also propose probable solutions to address those challenges.

2 MOTIVATION OF APPROXIMATE COMMUNICATION

We discuss the communication bottleneck in multi-core NoC and error resiliency of data-intensive applications. We present the opportunity of AComm to improve the energy-efficiency and performance of NoC.

2.1 Communication Bottleneck

Communication is the major bottleneck to achieve high-parallelism in large-scale systems with many compute nodes (processing cores). Communication takes relatively more time compared to computation with the increase of the number of processing cores in the systems [3, 21]. Communication also consumes more energy compared to computation [3, 21, 28]. Furthermore, the error control techniques introduce more communication overhead and higher power consumption as were shown in [4, 12]. Therefore, it is essential to reduce the volume of data communication for energy-efficient and high-performance NoC.

2.2 Error Tolerance of Applications

Many emerging data intensive applications, such as machine learning (ML), pattern-recognition, image processing, and big data analytics, can tolerate errors with acceptable accuracies [5, 19, 27]. To extract the benefits out of this error resilient nature of these applications, approximate computing has emerged as an attractive computing technique by trading off computation accuracy for benefits in both performance and energy efficiency. We can apply the same concept of approximate computing in the communication by approximating the communication data, between compute nodes, needed for computation. For example, we can reduce the communication data volume to exploit the lower accuracy requirements of the data-intensive applications. ET is defined as the difference between actual and approximated data that is within (less than) that ET (error tolerance and error threshold used interchangeably in the paper). For example, for 10% ET, the data 32 as the approximate data of accurate data 35 will be accepted (as the difference of 3 is within the 10% ET). Approximation can reduce the communication workload to get low-power and high-throughput NoC. Some of the approximation techniques to improve the energy-efficiency and performance are: value prediction, data-protection reduction, and compressing data. As the destination node does not need fully accurate data for computation, it presents an opportunity to reduce the data movement through NoC.

3 NOC DESIGN CHALLENGES AND APPROXIMATE COMMUNICATION OPPORTUNITY

Energy-efficiency and high-performance are the major challenges in multi/many core NoC. Both of the challenges are discussed in detail in the following sections.

3.1 Energy Efficiency

Relative power consumption of NoC has been increasing with the increase in the number of cores in a chip, e.g., NoC consumes 36% of total chip power in 16-tile MIT Raw [23]. Power and temperature density, the dissipation by the per unit of area of chip, becomes important design factors in deep-submicron technologies. Power and thermal hotspots bring significant challenges in chip reliability, performance, cooling costs, and leakage power. Power consumption and temperature dissipation are directly proportional, for example, leakage power is exponentially related to temperature. Temperature of a node also affects (increases) by the neighbor active nodes. Thermal hot spots accelerate failure mechanisms such as electromigration, stress migration, and dielectric breakdown, which can cause permanent device failures [15]. High temperature also degrades performance, as the effective operating speed of devices decreases with increasing temperature. Therefore, power consumption becomes a critical issue in many-core on-chip networks.

As a significant power and thermal source, NoC resources can be configured dynamically to decrease pressure on the power and

thermal budgets. Approximation can help this configuration process by reducing data movement based on the ET of the applications.

3.2 High Performance

As multi-core can provide multithreaded performance, performance can be increased indefinitely by increasing the number of cores theoretically. However, interconnection between cores imposes a limit on performance gain from multi-core systems [5, 7]. With the increase in number of cores, communication latency between distant nodes increases. Computational parallelism is hampered in multi-core systems as computation can be halted at a core by the data-dependent communication from the remote core or memory.

By approximating the packets in NoC, communication volume and/or number of communications can be reduced, and this will improve the packet latency. Processing elements (cores) can perform their computation quickly if the communication time of the data needed for computation reduces. Therefore, AComm can improve the computation speed and so can improve the parallelism in the system.

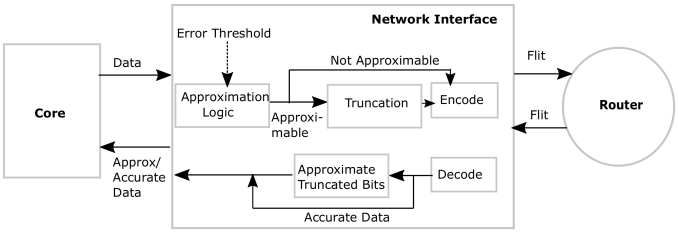

Figure 2: Approximate NoC Architecture

4 APPROXIMATE COMMUNICATION ARCHITECTURE

Figure 2 shows an approximate NoC architecture. We are presenting AComm technique to approximate the data dynamically based on the ET of the applications, while different applications have different accuracy requirements. Approximation is done in the network interface (NI) of the NoC, as NI is responsible for packetization/de-packetization of data, flit fragmentation/ assembly for flow control, transmission error control, and other functions in NoC. NI-level approximation implementation makes the NoC applicable for any kind of processing elements (soft or hard block) in the multi/many core architectures, as modification (e.g., cache architecture) is not needed in processing elements. Approximation logic in NI finds out whether data packet can be approximated or not, based on the annotation of the code sections [13, 20] of packet communication. Approximation flag (in approximation logic) is used to identify whether the data packet is approximable (can be approximated) or the data needs full accuracy. Based on the annotation and ET of an application, we can approximate the communication data by truncating the data-bits to reduce data movement among cores. Data truncation technique has been adapted in this work for approximation in the NoC. Data truncation technique removes part of packet based on the accuracy requirements of applications. At the destination, NI will approximate the truncated bits before sending the packet to the processing elements. Finally, as approximation/truncation reduces data movement and resource utilization in the NoC, we can apply dynamic voltage and frequency scaling (DVFS) to the links and routers to improve the energy-efficiency of NoC. Voltage-levels of the router and links can be reduced at run-time based on the approximation of data packets, and this reduces communication energy consumption and improves the energy-efficiency of NoC. DVFS application based on the approximation can significantly reduce the power and thermal density, which are the major challenges of multi-core architectures. The operation flow Chart of the proposed AComm technique is shown in Figure 3.

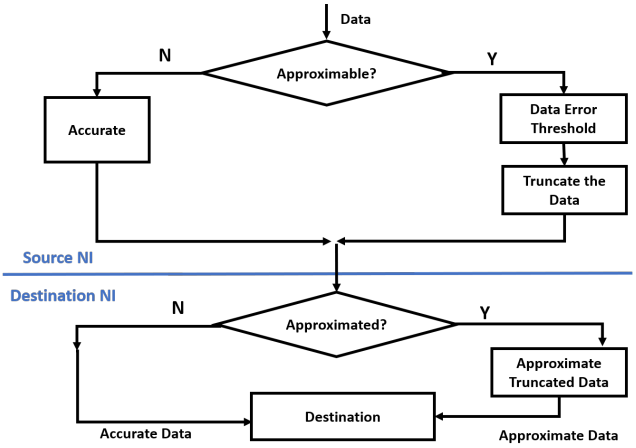

Figure 3: Flow Chart of Approximate Communication

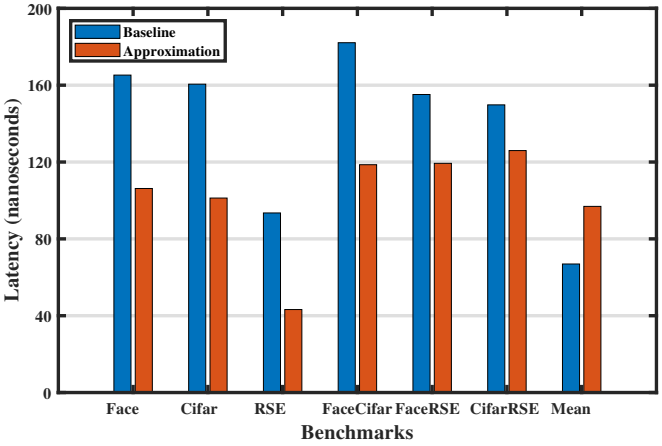

Figure 4: Latency Comparison between Approximate and Baseline Solutions for COSMIC Benchmarks

We simulate the proposed architecture using Garnet [1] on-chip network in gem5 platform [6]. Garnet/gem5 simulation configurations include: instruction set architecture = ALPHA , interconnect Frequency = 1GHz, no. of virtual networks (VN) = 3, no. of virtual channels (VC) per VN = 4, no. of buffers per VC = 4 and XY-routing [11]. DSENT [22] tool is integrated with gem5 to evaluate

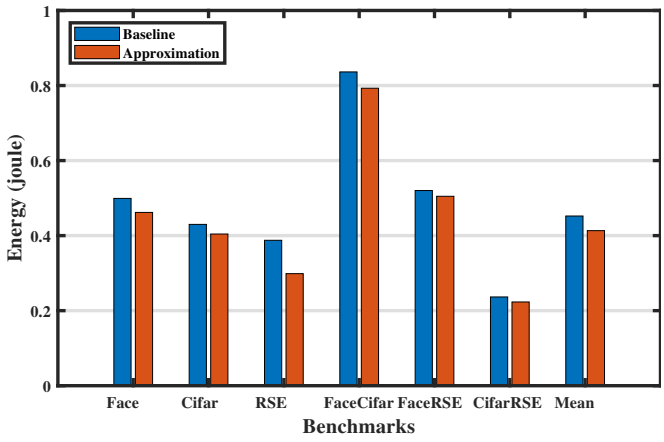

Figure 5: Energy Comparison between Approximate and Baseline Solutions for COSMIC Benchmarks

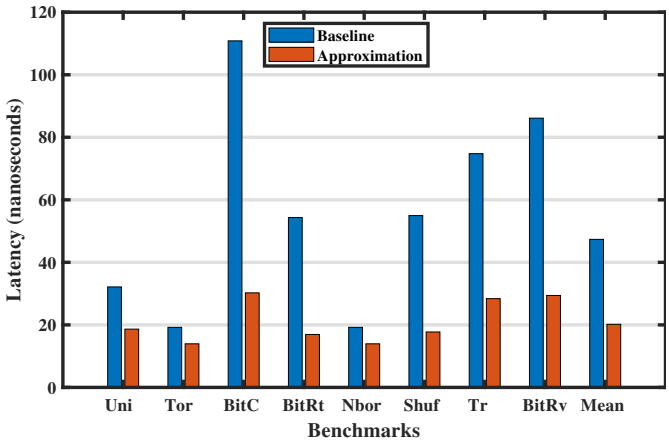

Figure 6: Latency Comparison between Approximate and Baseline Solutions for Synthetic Traffic Patterns

the energy consumption. Simulations were carried out using COSMIC benchmark suite [26] and synthetic traffic patterns. We assume all the packets can be approximated, and have applied 50% truncation for approximation to show the significance of AComm. The energy consumption and latency are used to evaluate the AComm technique. COSMIC benchmarks is simulated using 64-cores and routers in a 8×8 2D-mesh NoC. Three applications from COSMIC are mapped in our simulations: face recognition, cifar, and reed-solomon code encoder (RS-Encoder). Face recognition (face), cifar, and RS-Encoder (RSE) application have 33, 33, and 141 tasks, respectively. We also mix the applications to simulate multiple applications in NoC. As shown in Figure 4 and 5, AComm improves latency and energy by 50% and 10% compared to that of baseline, because of the reduction of the data movement through approximation. Eight synthetic traffic patterns are simulated using 16-cores and routers in a 4×4 2D-mesh NoC. Traffic patterns are: Uniform Random (Uni), Tornado (Tor), Bit-complement (BitC), Bit Rotation (BitRt), Neighbor (Nbor), Shuffle (Shuf), Transpose (Tr), and Bit

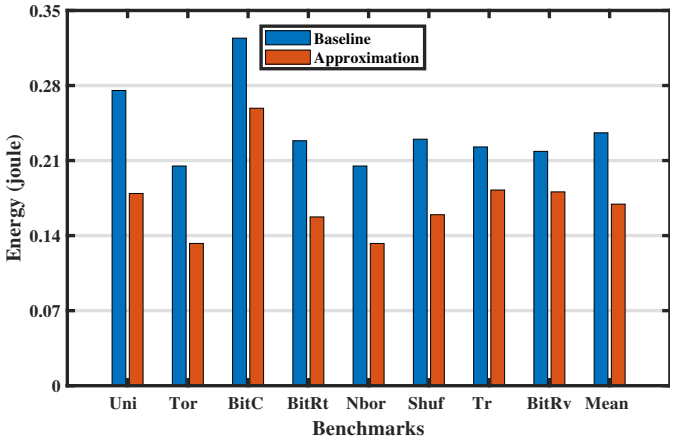

Figure 7: Energy Comparison between Approximate and Baseline Solutions for Synthetic Traffic Patterns

Reverse (BitRv). As shown in Figure 6 and 7, AComm improves latency and energy by 130% and 40% compared to that of baseline.

Therefore, we can see that AComm technique can significantly improve the energy and delay performance of the NoC, if the application is error-tolerant. Energy and performance improvement varies with the approximation ratio, which depends on the error-tolerance of the application(s). Because of the reduction of the data movement, throughput in the approximate NoC drops, but this may not hamper the performance of the application. But the output of the application needs to be within the ET to meet the accuracy requirements of the application. The output quality of an application can be measured by running the application separately on a host machine by cutting-off (truncating) the least significant bits of the variables in the approximable code sections (responsible for communication), and then we can compare the quality of approximate output and actual output of the application.

5 APPROXIMATE COMMUNICATION SOLUTIONS AND OPPORTUNITIES

We highlight several important AComm techniques for NoC.

5.1 Packet Prediction

Different packets in NoC face different latency because of shared resources of NoC. Some packets may face more congestion, and then can further introduce congestion for other packets. We can identify the packets that are introducing congestion in the NoC, and then predict those packets at the destination to reduce the number of communication packets and so to minimize congestion in NoC. Because of the packet prediction, core can continue their computation without waiting for the data to arrive from remote core or memory controller. For example, on a cache miss, core can predict the load value instead of fetching data from memory, which reduces the high cost of accessing memory [18]. This packet prediction technique can significantly reduce data movement in NoC, and can significantly increase parallelism in the NoC while reducing communication power consumption. ML can help to predict the packets (needed for computation) at the computing node. Here the

advantage is that approximate packet prediction (for error tolerant applications) does not need the recovery mechanism for misprediction. The authors in [25] propose an approximation-based dynamic traffic regulation technique to reduce congestion in NoC. This technique drops a fraction of packet data before they are injected into network, based on the error resilience of application, and predicts the lost data in packet after being received in the destination node.

5.2 Data Approximation with Compression

Based on the ET, data can be approximated by approximating the precise value. Approximate pattern can be computed by calculating the allowed deviation from the precise values. Furthermore, data compression technique can be used in combination with the approximation to further compress the data patterns into more compact bits before they are stored or transmitted. Compression methods have been used to reduce data movement for high-performance and/or low-power NoC [8, 10, 16]. Frequent-pattern compression [2] and dictionary-based compression [16] are the state-of-the-art compression mechanisms. Compression reduces network traffic, benefits both latency and power. Compression increases the effective bandwidth of the links and routers of NoC. After the packet reaches the destination NI, the packet needs to be decompressed for the destination processing element. So, the compression mechanism needs both compressor and decompressor modules in every node, NI, of the NoC. Therefore, the compression techniques incurs additional latency and hardware overhead for its adaptation in NoC. The authors in [8] propose a hardware data approximation framework, namely Approx-NoC, with an online data error control mechanism for high performance NoCs. Approx-NoC reduces the transmission of approximately similar data in the NoC by delivering approximated versions of precise data using state-of-the-art compression mechanisms. [16] applies compression at run-time on the congested paths only if the compression improves NoC performance.

By approximating the cache, we can reduce both data storage and data movement in NoC. For example, by compressing the cache, we can store more data in the same cache size, which improves the memory system performance. The authors in [10] proposes cache compression mechanism, besides NI compression. As shown in [10], cache compression provides on an average 21% reduction in network latency and 7% reduction in power consumption due to the reduction in the communication. However, to apply the cache approximation in the multi-core architecture, cache banks in the IP block (e.g., core) need to be modified, which is not possible for hard IP blocks. Approximation (e.g. compression) in the NI, as proposed in this paper, does not require any modification in the cache banks. So, the NI approximation is applicable for any on-chip systems.

5.3 Error Detection and Correction Reduction

Error correction introduces additional checking to check the correctness of the packets in the destination. If there is any bit-error found in the output, then the source needs to resend the packet again to destination. This retransmission incurs additional energy consumption. Error checking requirements can be reduced for the applications that can tolerate errors. By truncating the data bits based on the ET requirements of applications, we can reduce the probability of re-transmission because of less number of error checking in the destinations.

In packet transmission, we can have both integer and floating-point packets based on the nature of application. An integer value has higher precision requirement than that of floating point value. That means we need to protect more bits of integer than that of floating point value. For example, to achieve 10% data ET for absolute data value 10, we need to protect 31 MSBs and 13 MSBs for integer and floating-point values, respectively. However, we can convert integer packet to floating point packet to reduce the error detection/correction length. This will reduce the error checking length in the destination and will result in less number of retransmission requirement due to errors. In the floating point data, as shown in Figure 8, 1-bit sign and 8-bits exponent are the critical information. Critical bits needs to be protected to maintain the quality of approximation. Other bits of the packet can be approximated based on the ET requirements of the application. We can truncate a part of the lower significant bits (LSBs) while protecting a part of most significant bits (MSBs). However, data packet conversion is needed in the destination. [9] uses an AComm approach, which reduces the probability of retransmission by reducing the number of protected bits checking for correctness at the destination, based on the error resiliency of the application.

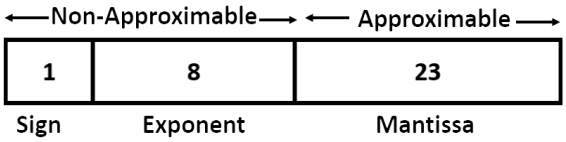

Figure 8: Floating Point Data Approximation

6 CHALLENGES OF APPROXIMATE COMMUNICATION

Approximate techniques need to overcome following challenges for its effective application in NoC.

6.1 Determination of Error Tolerance of Application

One major challenge in AComm is to determine the error tolerance (ET) requirement of an application. ET is the maximum acceptable accuracy loss of a value. One solution is to annotate the code sections of the application to find out the approximable and accurate (not approximable) data [13, 20]. Sensitivity analysis of an application can be done by running applications at different ETs, and determine the quality of the outputs (at different ETs). This sensitivity analysis can give us the picture of the extent of ET of an application. The ET requirement of an application can also suddenly change at run-time. Furthermore, different applications have different requirements on output quality, and it makes the task of determining ETs and applying approximation technique a challenging task.

6.2 Approximation Error and Machine Learning Solution

The major challenge of approximate techniques is to control the errors in the output. AComm error needs to be controlled to meet

the output quality requirements of an application. Recent study [17] have shown that overall output quality varies with the change in the approximation error of the individual elements. Therefore, an AComm technique needs to control individual errors in each data packet while controlling output errors across the whole application. Different code sections of an application may require different accuracy requirements. Determination of approximation error takes time, and this time may not be feasible for approximate technique implementation in run-time systems. That's why, proactive technique is needed to determine errors with the AComm. To ensure the quality of the output, ML techniques, such as reinforcement learning (RL) and neural networks, can predict the approximation errors and take steps (train the predictor) to improve the accuracy. ML agent can be trained with errors for data packet approximation. For example, RL agent can continually monitor the approximation error in the packets, and train with the data sets collected during (online) experience. The implementation of ML can improve the AComm error while introduce new challenges (e.g., overhead).

6.3 Bit Error Rate

Links of the NoC consume a significant percentage of the overall NoC and chip power. Voltage-level of the links can be reduced by reducing the data movement through links by the AComm. As power consumption decreases quadratically with the decrease in voltage-level, AComm can reduce the power significantly. However, reduction of the voltage swing of the NoC links increases the probability of bit errors, which can affect the reliability of the NoC [12]. Because of the error resilient nature, increase of the bit error rate (BER) may not affect the application. However, BER needs to be controlled to keep the output error within the ET of an application.

6.4 Software and Hardware Overheads

The application of approximation incurs software and hardware overheads. Approximation error determination for different packets and applications incurs some overhead. Approximation also incurs extra cycles (latency) for applying approximation on the data. Hardware module is needed for AComm logic implementation. For example, approximate module is needed to truncate the data based on the ET of the data. Data packet conversion modules are needed for converting integer and floating point packets to reduce the bit-width of the data to be moved. Module is also needed for voltage and frequency scaling of links and routers. To apply ML for approximation error prediction, it (ML) needs hardware in terms of adder, multiplier and registers for calculating and storing data (for training and prediction). The implementation of approximation needs to be carefully designed so that overheads of approximation do not exceed the energy and performance gain.

7 CONCLUSIONS

We have proposed an architecture for approximate communication for error tolerant applications in NoC based multi/many core systems. Simulation results were presented to show the energy and delay performance improvement from approximation in NoC. We discussed several approximation communication solutions that have shown great promise to gain performance and energy improvement by reducing the number of packets and/or the size of the packets. Finally, we presented several challenges for the effective application of approximate communication techniques in NoC.

REFERENCES

[1] N. Agarwal, T. Krishna, L. S. Peh, and N. K. Jha. 2009. GARNET: A detailed on-chip network model inside a full-system simulator. In *Proc. ISPASS*. 33–42. https://doi.org/10.1109/ISPASS.2009.4919636
[2] Alaa R. Alameldeen and David A. Wood. 2004. Frequent Pattern Compression: A Significance-Based Compression Scheme for L2 Caches.
[3] Keren Bergman et al. 2008. ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems.
[4] Davide Bertozzi, Luca Benini, and Giovanni De Micheli. 2005. Error control schemes for on-chip communication links: the energy-reliability tradeoff. *IEEE TCAD* 24, 6 (2005), 818–831.
[5] Filipe Betzel et al. 2018. Approximate Communication: Techniques for Reducing Communication Bottlenecks in Large-Scale Parallel Systems. *ACM Comput. Surv.* 51, 1, Article 1 (Jan. 2018), 32 pages. https://doi.org/10.1145/3145812
[6] Nathan Binkert et al. 2011. The Gem5 Simulator. *SIGARCH Comput. Archit. News* 39, 2 (2011), 1–7. https://doi.org/10.1145/2024716.2024718
[7] S. Borkar. 2013. Exascale computing - A fact or a fiction?. In *Keynote at IPDPS*. https://doi.org/10.1109/IPDPS.2013.121
[8] Rahul Boyapati et al. 2017. APPROX-NOC: A Data Approximation Framework for Network-On-Chip Architectures. In *Proceedings of ISCA*. 666–677. https://doi.org/10.1145/3079856.3080241
[9] Y. Chen, M. F. Reza, and A. Louri. 2018. DEC-NOC: An Approximate Framework Based on Dynamic Error Control with Applications to Energy-Efficient NoCs. In *Proceedings of ICCD*. 480–487. https://doi.org/10.1109/ICCD.2018.00078
[10] R. Das et al. 2008. Performance and power optimization through data compression in Network-on-Chip architectures. In *Proceedings of HPCA*. 215–225. https://doi.org/10.1109/HPCA.2008.4658641
[11] Jose Duato, Sudhakar Yalamanchili, and Ni Lionel. 2002. *Interconnection Networks: An Engineering Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
[12] Alireza Ejlali et al. 2010. Performability/energy tradeoff in error-control schemes for on-chip networks. *IEEE TVLSI* 18, 1 (2010), 1–14.
[13] Hadi Esmaeilzadeh, Adrian Sampson, Luis Ceze, and Doug Burger. 2012. Neural acceleration for general-purpose approximate programs. In *Proceedings of MICRO*. 449–460.
[14] Y. Hoskote et al. 2007. A 5-ghz mesh interconnect for a teraflops processor. *IEEE Micro* 27, 5 (2007), 51–61.
[15] JEDEC. 2008. Failure Mechanisms and Models for Semiconductor Devices, JEDEC publication JEP122C. http://www.jedec.org.
[16] Y. Jin, K. H. Yum, and E. J. Kim. 2008. Adaptive data compression for high-performance low-power on-chip networks. In *Proceedings of MICRO*. 354–363. https://doi.org/10.1109/MICRO.2008.4771804
[17] Daya S Khudia, Babak Zamirai, Mehrzad Samadi, and Scott Mahlke. 2015. Rumba: An online quality management system for approximate computing. In *Proceedings of ISCA*. 554–566.
[18] J. S. Miguel, M. Badr, and N. E. Jerger. 2014. Load Value Approximation. In *Proceedings of MICRO*. 127–139. https://doi.org/10.1109/MICRO.2014.22
[19] Sparsh Mittal. 2016. A survey of techniques for approximate computing. *ACM Computing Surveys (CSUR)* 48, 4 (2016), 62.
[20] Adrian Sampson et al. 2011. EnerJ: Approximate data types for safe and general low-power computation. In *ACM SIGPLAN Notices*, Vol. 46. 164–174.
[21] John Shalf, Sudip Dosanjh, and John Morrison. 2010. Exascale computing technology challenges. In *Proceedings of VECPAR*. 1–25.
[22] Chen Sun et al. 2012. DSENT - A Tool Connecting Emerging Photonics with Electronics for Opto-Electronic Networks-on-Chip Modeling. In *Proceedings of NOCS*. 201–210. https://doi.org/10.1109/NOCS.2012.31
[23] Michael Bedford Taylor et al. 2002. The Raw Microprocessor: A Computational Fabric for Software Circuits and General-Purpose Programs. *IEEE Micro* 22, 2 (2002), 25–35. https://doi.org/10.1109/MM.2002.997877
[24] Hangsheng Wang, Li-Shiuan Peh, and S. Malik. 2003. Power-driven design of router microarchitectures in on-chip networks. In *Proceedings of MICRO*. 105–116. https://doi.org/10.1109/MICRO.2003.1253187
[25] L. Wang, X. Wang, and Y. Wang. 2017. ABDTR: Approximation-Based Dynamic Traffic Regulation for Networks-on-Chip Systems. In *Proceedings of ICCD*. 153–160. https://doi.org/10.1109/ICCD.2017.31
[26] Zhe Wang et al. 2014. A Case Study on the Communication and Computation Behaviors of Real Applications in NoC-Based MPSoCs. In *Proc. ISVLSI*. 480–485. https://doi.org/10.1109/ISVLSI.2014.36
[27] Qiang Xu, Todd Mytkowicz, and Nam Sung Kim. 2016. Approximate computing: A survey. *IEEE Design & Test* 33, 1 (2016), 8–22.
[28] F. Zahn, S. Lammel, and H. FrÃűning. 2017. Early Experiences with Saving Energy in Direct Interconnection Networks. In *Proceedings of HiPINEB*. 33–40. https://doi.org/10.1109/HiPINEB.2017.10