Exploiting Data Resilience in Wireless Network-on-chip Architectures

GIUSEPPE ASCIA, VINCENZO CATANIA, SALVATORE MONTELEONE, MAURIZIO PALESI, and DAVIDE PATTI, University of Catania, Italy
JOHN JOSE, Indian Institute of Technology Guwahati, India
VALERIO MARIO SALERNO, University of Enna, KORE, Italy

The emerging wireless Network-on-Chip (WiNoC) architectures are a viable solution for addressing the scalability limitations of manycore architectures in which multi-hop long-range communications strongly impact both the performance and energy figures of the system. The energy consumption of wired links as well as that of radio communications account for a relevant fraction of the overall energy budget. In this article, we extend the approximate computing paradigm to the case of the on-chip communication system in manycore architectures. We present techniques, circuitries, and programming interfaces aimed at reducing the energy consumption of a WiNoC by exploiting the trade-off energy saving vs. application output degradation. The proposed platform—namely, xWiNoC—uses variable voltage swing links and tunable transmitting power wireless interfaces along with a programming interface that allows the programmer to specify those data structures that are error-resilient. Thus, communications induced by the access to such error-resilient data structures are carried out by using links and radio channels that are configured to work in a low energy mode, albeit by exposing a higher bit error rate. xWiNoC is assessed on a set of applications belonging to different domains in which the trade-off energy vs. performance vs. application result quality is discussed. We found that up to 50% of communication energy saving can be obtained with a negligible impact on the application output quality and 3% in application performance degradation.

CCS Concepts: • **Networks → Network on chip**; • **Computer systems organization → Interconnection architectures**; • **Hardware → Interconnect power issues**;

Additional Key Words and Phrases: Approximate computing, Approximate communication, wireless network-on-chip, energy quality trade-off

ACM Reference format:
Giuseppe Ascia, Vincenzo Catania, Salvatore Monteleone, Maurizio Palesi, Davide Patti, John Jose, and Valerio Mario Salerno. 2020. Exploiting Data Resilience in Wireless Network-on-chip Architectures. *J. Emerg. Technol. Comput. Syst.* 16, 2, Article 21 (April 2020), 27 pages.
https://doi.org/10.1145/3379448

Authors' addresses: G. Ascia, V. Catania, S. Monteleone, M. Palesi, and D. Patti, University of Catania, V.le A. Doria 6, 95125 Catania, Italy; emails: {giuseppe.ascia, vincenzo.catania, salvatore.monteleone, maurizio.palesi, davide.patti}@dieei.unict.it; J. Jose, Indian Institute of Technology Guwahati, 781039, Assam, India; email: johnjose@iitg.ernet.in; V. M. Salerno, University of Enna, KORE, 94100 Enna, Italy; email: valerio.salerno@unikore.it.

1550-4832/2020/04-ART21 $15.00
https://doi.org/10.1145/3379448

1 INTRODUCTION

The approximate computing (AC) paradigm has been revitalized by the emerging recognition, mining, and synthesis (RMS) applications [10], which cover a wide computing spectrum ranging from Internet of Things (IoT) to large-scale data centers. Several studies have shown that RMS applications possess a so-called "forgiving nature" that makes them intrinsically resilient to errors/imprecision affecting both the computation, the storage, and the communication [23]. AC techniques exploit the forgiving nature of applications by exploring a new dimension of the design space—namely, application output quality. Most of the AC techniques in the literature focus on the computing sub-system [28, 49] in which the computational requirement of the application is reduced by trading off with the degradation of its output quality. Although some of these techniques have also an indirect impact on the communication, in the sense that the reduction of the computational effort often results in a reduction of the communication requirements, only a few of them have been specifically designed to improve the performance of the communication sub-system.

In fact, the role played by the communication sub-system is becoming more and more important, especially in the many-core era [41]. If we look at the fraction of time spent in communication in current parallel exascale applications (Figure 1), then it is possible to realize that performance scalability is severely affected by communication as the number of processing elements increases. A similar trend can also be observed for energy consumption in which the Network-on-Chip (NoC) accounts for a relevant fraction of the overall energy budget [36].

In Reference [5], the *approximate communication* is defined as the application of AC techniques to parallel systems aiming at reducing the amount of communication between processing elements. In this article, we extend the approximate communication paradigm to the case in which the communication traffic is not reduced in volume, but the communication reliability is decreased in favor of energy saving.

We consider emerging wireless NoC (WiNoC) architectures and propose circuitries and programming interfaces for improving the energy-efficiency of the system by trading off the quality of the application results. Voltage overscaling is used as the main approximate computing technique in which both the voltage swing of the links and the transmitting power of the wireless interfaces are selectively tuned at run-time to reduce the overall communication energy consumption. We show how the forgiving nature of emerging RMS applications makes them tolerant to errors affecting the on-chip communication traffic induced by the access to those data structures that have been preemptively annotated by the programmer as being error-resilient.

This work goes to the same direction of AxNoC [1] but with less impact on the datapath of the router and extending the approximate communication paradigm to WiNoC architectures. Specifically, regarding the first aspect, in AxNoC all the modules in the router datapath (with the exception of the next router computation and virtual-channel/switch allocation) are voltage scaled. This imposes specific optimizations and dictates the use of look-ahead wake-up method [22] to start the voltage transition one clock cycle before the packets reach the router. In this work, we relax the need of using the look-ahead wake-up method, thus simplifying and generalizing the implementation, although with some performance degradation, as discussed in the rest of the article. Regarding the second aspect of the extension to WiNoC architectures, it should be pointed out that the use of the radio channel is advisable only for long-range communications, as its energy per bit is higher than that of electric links. Adopting the proposed technique, the transmitting power and, consequently, the energy per bit can be drastically reduced with an improvement of the utilization of the wireless network and overall energy figures, again, trading off with application quality metrics.

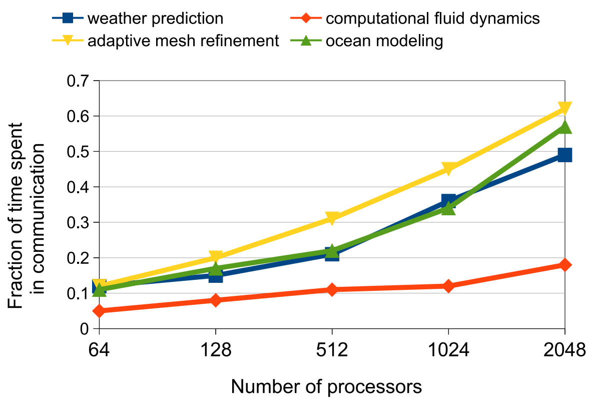


Fig. 1. Fraction of time spent in communication in multi-core systems with different processor count under different representative exascale parallel applications.

The contributions of the article are as follows:

- The extension of the approximate computing paradigm to the communication sub-system—namely, approximate communication. Specifically, the conventional approach of reducing the amount of traffic by employing approximated computing techniques [5] is extended by reducing the amount of energy spent for communication by exploiting the different nature of communication flows in terms of their specific reliability requirements.
- The application of the approximate communication to the emerging WiNoC architectures. In fact, current contributions in the context of approximate communication are limited to conventional wired NoC architectures. The energy spent for wireless communication in WiNoC architectures accounts for a relevant fraction of the overall communication energy [29]. This work provides techniques to improve the energy-efficiency of the wireless communication fabric. To the best of our knowledge, this is the first work that extends the approximate computing paradigm to WiNoC architectures.
- The design, implementation, and characterization of a tunable voltage swing NoC router and a tunable transmitting power wireless interface for WiNoC routers. Differently from the other techniques [1] that require changes of several stages of the router pipeline and the use of specific look-ahead strategies [22], the proposed technique is less invasive and only affects the link traversal stage of the router pipeline.

The rest of the article is organized as follows: In Section 2, we summarize some previous works related to the application of AC techniques to the communication sub-system. Next, in Sections 3 and 4, we introduce the xWiNoC platform and the architecture details, respectively. Finally, an extensive set of experiments is presented in Section 5 to demonstrate the effectiveness of the proposed approach in terms of energy/quality trade-off. Finally, conclusions are drawn in Section 6.

2 RELATED WORK

Several AC techniques have been proposed in the literature [19, 28, 39, 49]. The common factor among them is the relaxation of the numerical equivalence between specification and implementation of error-tolerant applications in return for higher, scalable performance and energy-efficiency. Although their application has a direct impact on computation capability, in many cases they also have an indirect positive impact on the communication part. For instance, AC techniques such as loop perforation [42], memory access skipping, and thread fusion [40] reduce the communication traffic to and from memory. Other AC techniques, such as data sampling [18], lossy compression [40], and precision scaling [45], reduce the message size. Of course, both the reduction of the number of messages and their size improves the communication performance.

The ever more important role played by the communication sub-system pushes toward the definition of approximate computing techniques specifically designed for the communication system. The use of compression techniques applied at the network interface level in NoC-based systems allows reducing the volume of data movement across the chip with a consequent energy and performance improvement. APPROX-NoC [6] can be used in conjunction with such compression mechanisms to increase their compression ratio. Specifically, it approximates data before their compression in such a way to increase the similarity among data to be transmitted. The higher the similarity, the higher the compression ratio. Such transformation is lossy (i.e., the information received at the destination might be different than that sent), thus only information-loss-tolerant data structures of the application are approximated. In the context of many-core GPGPU architectures, approximate communication techniques have been used for coping with the high volumes of read requests generated by the parallel warps of threads running on shader cores. DAPPER [38] implements a memory controller (MC) architecture that leverages the inherent approximability of the data value of certain applications by reducing the number of reply packets injected into the NoC by the MCs. Approximation-based dynamic traffic regulation (ABDTR) [46] approximates part of traffic data instead of network transmission for mitigating network congestion based on the inherent error-resilience of the application. Accuracy- and Congestion-aware Dynamic traffic Control (ACDC) [48] implements a lightweight control mechanism to solve the performance optimization problem for approximate NoC. In the context of wireless NoC, to reduce contention and latency in the wireless channel of WiNoC-based manycore systems, Replica [16] selectively drops wireless packets when the sender encounters a certain level of contention. Approximate programming techniques to restructure applications are also proposed to leverage wireless communication.

Other approximate communication techniques based on voltage overscaling have been presented in References [1, 4]. Here, the communications carrying error-tolerant information are routed through links operating at a reduced voltage swing level. The reduced voltage results in energy saving that is traded off with an increase of the bit error rate. Once again, the application has to be properly annotated to expose those data structures that are error-resilient for which their induced communication flows can be routed on low energy yet low reliable links.

The aforementioned works are defined and applied in the context of wired NoC architectures. To the best of our knowledge, there are still no contributions aimed at investigating the application of the approximate communication paradigm in wireless NoC architectures. In fact, the reduced reliability level of the radio medium, as compared to the wired counterpart, makes the approximate communication particularly oriented in such domain.

3 OVERVIEW OF XWINOC PLATFORM

3.1 Motivations

The on-chip communication system accounts for a relevant fraction of the overall energy budget [36]. In particular, the energy consumed by the links of the NoC represents one of the major contributions in the communication energy breakdown. Figure 2 shows the energy breakdown of a NoC router for different link lengths. Although the energy breakdown varies with different configurations of the router,[1] the overall picture does not change. In particular, as technology node

[1]The energy breakdown reported in Figure 2 refers to a 45 nm five-port router, four-flit input buffers, 64-bit, 2 GHz.

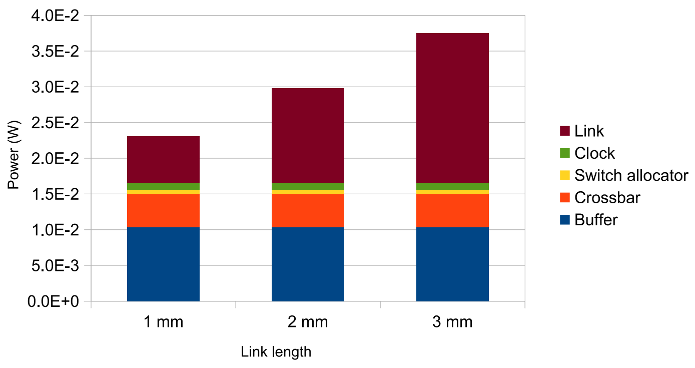


Fig. 2. NoC energy breakdown for different link lengths.

shrinks the wire capacitance vs. gate capacitance increases [20], the energy relevance of the NoC links is exacerbated. The same trend is observed in WiNoC architectures in which the energy consumption of wireless single-hop long-range communications is dominant [32]. This is due to the fact that the transmitting power used by the wireless interface is calibrated for the worst case to guarantee a minimum signal-to-noise ratio level for any pair of radio hubs. For this reason, a wireless communication becomes more energy-efficient than a wired communication only when the distance between the communicating nodes is greater than a certain threshold [32].

The approximate communication paradigm can be used to improve the energy-efficiency of WiNoC architectures. The approximate WiNoC (xWiNoC) architecture, proposed in this article, exploits the variable voltage swing of network links and the variable transmitting power of wireless interfaces to trade off communication energy with communication reliability.

3.2 Augmented Communication Primitives

In xWiNoC the `send` primitive is augmented with an additional parameter that allows to specify the expected reliability level of the communication. The reliability level measures the probability that the transmitted information reaches the destination without errors. `send(addr, data, rl)` sends `data` to `addr` with a reliability level `rl`. In a traditional NoC, all the communications are realized by using the highest reliability level. In xWiNoC the user (i.e., the application developer) has the freedom to specify the appropriate reliability level to be used for the specific communication. In the cases of implicit communication mechanisms, like in the case of a shared memory paradigm, the compiler is responsible for converting the load and store instructions to the appropriate reliability level used by the induced send primitive. This is realized by means of pragmas, by which the user declares the data structures that are error-tolerant and by which extent. The top part of Figure 3 shows a fragment of code in which the information to be read from `w` is assumed to be error-tolerant. Such annotation is carried out by means of pragma `resilient(w,rl)` by which the compiler is informed that the read accesses to `w` have to induce messages to be sent with the reliability level `rl`. This results in two messages:

- `send(&w[i], <load_op,rl>, rl_max)` by which the core informs the memory controller that wants to perform a load operation with reliability level `rl` at address `&w[i]`. The reliability level of the communication is set to `rl_max`, which defines the highest (nominal) reliability level.
- `send(core_id, w[i], rl)` by which the memory controller sends back to the core the requested information with the reliability level `rl` that has been communicated by the request message.

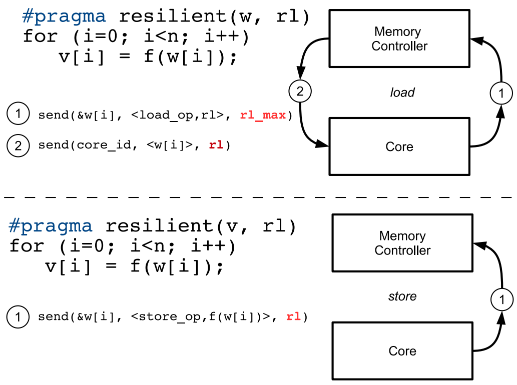


Fig. 3. Pragma-based programming interface for defining the resilient data structures and the reliability level to be used by the induced load (top) and store (bottom) communications.

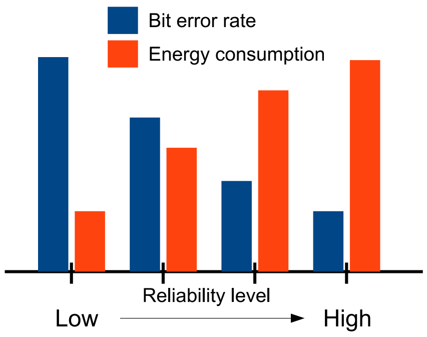


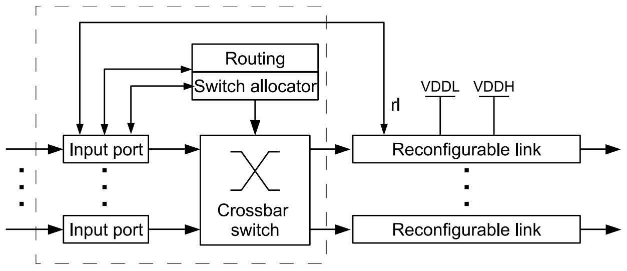
Fig. 4. Qualitative representation of the trade-off between reliability level and bit error rate.

The bottom part of Figure 3 shows another fragment of code in which, this time, the information to be written into `v` is considered to be error-tolerant. Such behavior is dictated by the use of pragma `resilient(v,rl)` by which the compiler is informed that a store to `v[i]` has to induce a message to be sent with the reliability level `rl`.

3.3 Implications and Trade-offs

The awareness of the network about the reliability level of the current message is realized by storing the reliability level into the header flit of the packet. To support n reliability levels, $\lceil log_2 n \rceil$ bits are reserved into the header flit. Each router traversed by the packet will use the reliability level field to configure the appropriate transmission mechanism. xWiNoC uses dynamic voltage swing and transmission power tuning as mechanisms to set the appropriate communication reliability level. A lower voltage swing for a wired transmission or a lower transmitting power for a wireless transmission will result in a lower reliability level (measured in terms of bit error rate, BER) and to a lower communication energy consumption as qualitatively shown in Figure 4 and quantitatively discussed in Section 4.

Based on the above overview of xWiNoC, the programmer is responsible for locating the error-tolerant data structures of the application. The degree of error tolerance of the selected data structures allows to select the appropriate reliability level to be used when they are accessed. The definition of an automatic technique, aimed at discovering the error-tolerance level of the data structures



(a)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Last body flit (L)	BW	-	ST	_ (L->L)	LT														
Header flit		BW-RC	SA	ST	LC (L->H)	LT													
First body flit (L)			BW	-	ST	LC (H->L)	LT												
Body flit (L)				BW	-	ST	_ (L->L)	LT											
Body flit (L)					BW	-	ST	_ (L->L)	LT										
Body flit (L)						BW	-	ST	_ (L->L)	LT									
Last body flit (L)							BW	-	ST	_ (L->L)	LT								
Header flit								BW-RC	SA	ST	LC (L->H)	LT							
First body flit (H)									BW	-	ST	_ (H->H)	LT						
Body flit (L)										BW	-	ST	_ (H->H)	LT					
Body flit (L)											BW	-	ST	_ (H->H)	LT				
Body flit (L)												BW	-	ST	_ (H->H)	LT			
Last body flit (H)													BW	-	ST	_ (H->H)	LT		
Header flit														BW-RC	SA	ST	_ (H->H)	LT	
First body flit (L)															BW	-	ST	LC (H->L)	LT

(b)

Fig. 5. Router architecture (a) and temporal diagram of the pipeline for the transmission of a low reliability packet and a high reliability packet (b).

involved in the application, is out of the scope of this article. Thus, this article assumes that the application is already annotated by means of pragmas as described before. The problem of defining a procedure to automatically characterize the data structures of the application in terms of their error tolerance is important, as it will make the proposed technique of general applicability. However, we leave this problem out for future development, and we focus on the definition of a communication platform able to support the approximate communication paradigm.

4 XWINOC ARCHITECTURE

This section describes the two main basic blocks that form the xWiNoC architecture—namely, router and radio hub. Routers are responsible for routing the packets through the wired network, whereas radio hubs are responsible for the wireless transmission of the packets. Both of them can be configured at run-time (at packet granularity) to transmit the packet with a specific reliability level. The reliability level knob configures the router to drive its output links with different voltage swing levels and the radio hub to use different transmitting power levels.

4.1 Router Architecture

The router in xWiNoC is a generic router that is augmented with a logic that allows it to use multiple voltage swing levels for packet transmission (see Figure 5(a)). The router implements a five-stage pipeline (see Figure 5(b)) with buffer write and route computation (BW-RC), switch allocation (SA), switch traversal (ST), link configuration (LC), and link traversal (LT). In LC stage the output link is configured with the appropriate voltage swing level. The reconfiguration process works at a packet granularity. Specifically, the reliability level encoded into the header flit is used to configure the voltage swing of the output link. Such configuration is valid starting from the first body flit, as the header flit is always transmitted using the highest reliability level. This is due to the fact that the header flit contains control information that is not error-tolerant. Thus, the

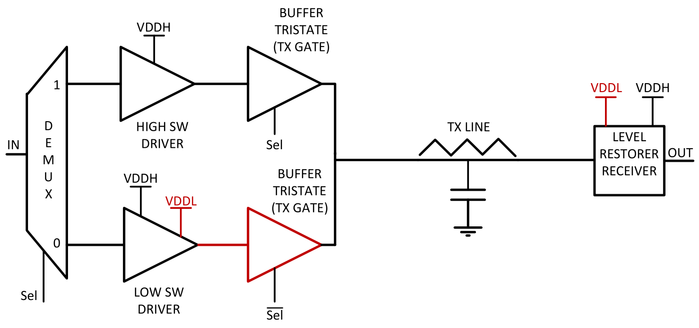


Fig. 6. Architecture of the reconfigurable bitline.

LC works only by the header flit (in which the output link is configured to work at the highest reliability level) and by the first body flit, which configures the output link to use the specific voltage swing level accordingly to the reliability level in the header flit. Based on this, Figure 5(b) shows the temporal diagram of the pipeline for all the combinations in which high reliability and low reliability packets alternate. Please note that, in the diagram, for stage LC, we report into brackets the voltage swing level before and after the link configuration. Cells highlighted in green show the case in which the link is actually reconfigured; otherwise, the LC label is replaced with the symbol underscore.

4.1.1 Reconfigurable Link Architecture. Without loss of generality and for the sake of clarity, the implementation discussed in this article uses two voltage swing levels—namely, low voltage swing (VDDL) and high (or nominal) voltage swing (VDDH). In this case, the reliability level (*rl*) is encoded with just one bit. The proposed configurable link architecture is formed by a set of configurable bitlines whose scheme is shown in Figure 6. As it can be observed, the bitline is preceded by a chain formed by a demultiplexer, two tapered buffers as line drivers, and two tristate buffers [27, 51]. The selection command of the demultiplexer is driven by the *rl* bit encoded in the header flit of the packet. The tristate buffers are based on transmission gate logic. Thus, if the select input is high (low)—that is, if the reliability level bit is high (low)—the full (low) swing path is active and the low (high) swing path is disconnected by the high impedance state introduced by tristate buffer. At the end of the bitline, a level restorer circuit, similar to the sense amplifier used in RAM memories, is used. The level restorer receiver block restores the signal at full swing if the signal on the line is set to low swing or maintains the original swing if the signal is in full swing mode. As in Reference [51], the logic threshold used by the restorer is set at half of the low swing.

We designed the configurable link targeting a clock frequency of 2 GHz, which corresponds to the clock speed of our baseline router [30]. The analysis has been carried out with HSPICE using a 45 nm CMOS low voltage threshold library from Nangate [43] that provides 10 metal layers. Table 1 reports technology-related information. The parasitics extraction from layout has been made with Cadence Virtuoso. With the same tool, we estimate the silicon area occupancy. Table 2 reports the synthesis results and other additional information. We considered three different configurable link options: RLink1, RLink2, and RLink3. Each of them uses a different low voltage swing level VDDL of 0.9 V, 0.8 V, and 0.6 V, respectively. The nominal VDDH is 1.1 V for all the configurations. The three different VDDL determines three different BER that increase as VDDL decreases. The increase of the BER is traded off by the reduction of the average energy consumption per bit that reduces by 41%, 50%, and 70% for VDDL 0.9 V, 0.8 V, and 0.6 V, respectively. However, due to the overhead of the reconfiguration logic, when the reconfigurable links work at their nominal voltage swing VDDH, the energy consumption per bit increases by 3%. The worst-case total delay

Table 1. Technology Information

Parameter	Value
Technology	1.1 V, 10 metal, 45 nm CMOS LVT
Interconnect width	0.4 μm
Interconnect space	0.32 μm
Interconnect length	2.8 mm
Wire resistance	225 Ω (metal 7)
Wire capacitance	946 fF (metal 7)

Table 2. Conventional Full-swing Link vs. Reconfigurable Links

		Full-swing link	RLink1	RLink2	RLink3
Voltage swing (V)	VDDH	1.1	1.1	1.1	1.1
	VDDL	—	0.9	0.8	0.6
Bit error rate	VDDH	1.3E-17	1.3E-17	1.3E-17	1.3E-17
	VDDL	—	2.2E-12	3.8E-10	3.6E-6
Average EPB (fJ)	VDDH	512	527	527	527
	VDDL	—	304	258	152
Worst case total delay (ps)		214	375	387	410
Line driver (μm^2)		1,237	2,624	2,624	2,624
Line receiver (μm^2)		181	960	960	960

increases as the VDDL decreases. However, even for RLink3, in which VDDL is the lowest one, the worst-case delay is below the clock period of the router and it does not impact the latency of the router.

The worst-case delay at VDDL is due to the slowdown of the link and the delay due to reconfigure the voltage in the worst case; that is, from VDDL to VDDH. The reconfigurable voltage swing link architecture uses a double path in which each path is already configured to work at a fixed voltage level—namely, VDDL and VDDH (see Figure 6). Thus, the only delay introduced by the reconfigurable logic is due to drive the multiplexer and the tristate buffer. We decided to add an additional pipeline stage (LC stage) that encapsulates the link configuration logic instead of merging it in the link traversal stage. Such choice is motivated by the fact that (i) it provides more margin to implement more complex reconfiguration strategies than the proposed one and (ii) it allows more aggressive voltage swing scaling that slows down the link traversal time. The performance degradation due to the addition of such pipeline stage, used in correspondence of link voltage swing change, is assessed and discussed in Section 5.6.

Finally, the table reports the line driver and line receiver area that, with respect to the baseline router [30] (configured for five ports, 64-bit flit, four-flit buffers), introduce an overhead of 3%.

4.1.2 Energy per Bit Computation. The EPB values reported in the table are computed by averaging the energy consumption due to the different types of transitions that occur on the current bitline and in the neighboring bitlines (contribution due to crosstalk) [35]. Specifically, we considered five transition types, as reported in Table 3. The energy consumption for each transition pattern is computed as $C_{eff} \cdot V_{dd}^2$, where C_{eff} depends on the specific transition pattern. C_{eff} is computed as the weighted sum between C_s and C_c, where C_s is the bitline's self capacitance and C_c is the coupling capacitance. A transition pattern is represented by a triple in which the symbol in the middle represents the current bitline (victim) and the other the neighboring bitlines

Table 3. Effective Capacitance, C_{eff}, for Each Type of Transition on a Victim and Aggressors Lines [35]

C_{eff}	Transition Pattern
C_s	$(\uparrow,\uparrow,\uparrow)$ $(\downarrow,\downarrow,\downarrow)$
$C_s + C_c$	$(-,\uparrow,\uparrow)$ $(-,\downarrow,\downarrow)(\uparrow,\uparrow,-)$ $(\downarrow,\downarrow,-)$
$C_s + 2C_c$	$(-,\uparrow,-)$ $(-,\downarrow,-)(\downarrow,\uparrow,\uparrow)$ $(\downarrow,\downarrow,\uparrow)$ $(\uparrow,\uparrow,\downarrow)$ $(\uparrow,\downarrow,\downarrow)$
$C_s + 3C_c$	$(-,\uparrow,\downarrow)$ $(-,\downarrow,\uparrow)(\uparrow,\downarrow,-)$ $(\downarrow,\uparrow,-)$
$C_s + 4C_c$	$(\uparrow,\downarrow,\uparrow)(\downarrow,\uparrow,\downarrow)$

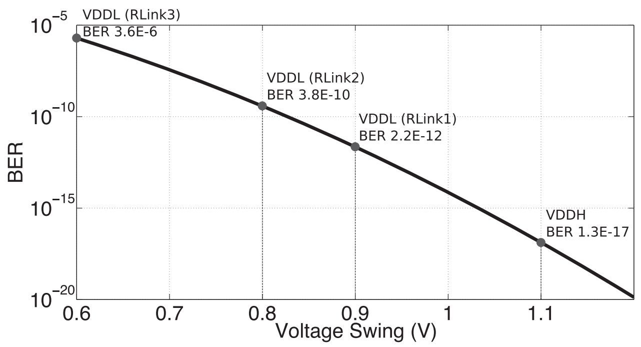


Fig. 7. Bit error rate versus voltage swing.

(aggressors). Symbols ↑ and ↓ represent the direction of the transition, 0 to 1 and 1 to 0, respectively. Symbol "–" represents no transition (0 to 0 or 1 to 1).

4.1.3 Bit Error Rate Estimation. The variation of the bit error rate (BER) of a bit line as V_{dd} is made to vary, as shown in Figure 7. The figure shows the VDDL used by the three considered reconfigurable links and the related BER when the reliability level is set to low. The curve has been computed by assuming that the overall transient UDSM noise effects are modeled by an additive Gaussian noise with variance $\sigma_N{}^2$ [44]. Based on this model, the BER is directly related to the voltage swing as:

$$\text{BER} = Q\left(\frac{V_{dd}}{2\sigma_N{}^2}\right), \tag{1}$$

where the Q function is the tail probability of the standard normal distribution:

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^{\infty} e^{-\frac{y^2}{2}} dy. \tag{2}$$

4.2 Radio hub Architecture

Radio hubs allow single-hop communication among faraway nodes in the network that would require multiple hops in the wired network. A radio hub is a non-terminal node connected to multiple routers. Figure 8 shows the main blocks into the radio hub. Overall, it provides a set of ports to be connected with a number of routers. The channel access token controller (MAC) implements the radio access control mechanism [34]. In particular, all the radio hubs capable of transmitting on a given channel belong to a logical token ring and share a token associated to the

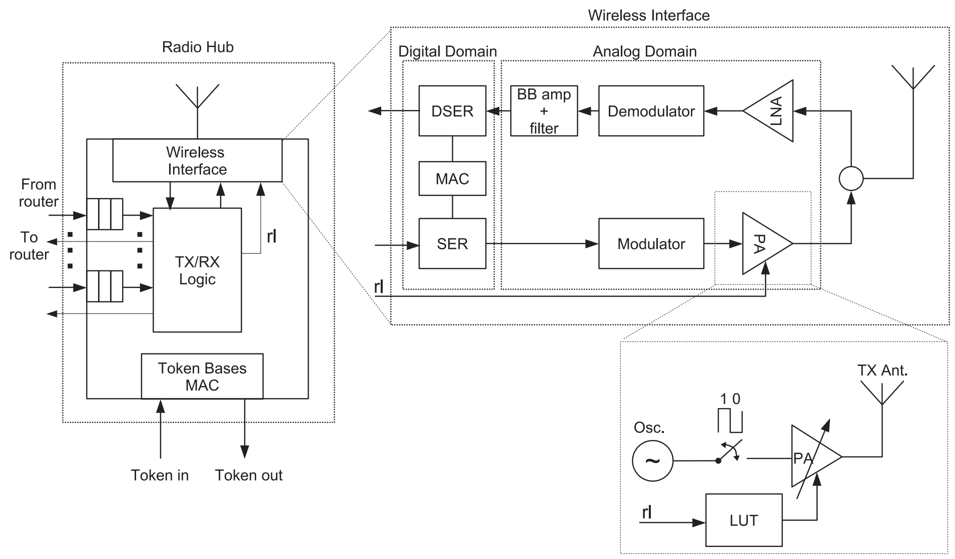


Fig. 8. Block diagram of the architecture of the radio hub and its network interface.

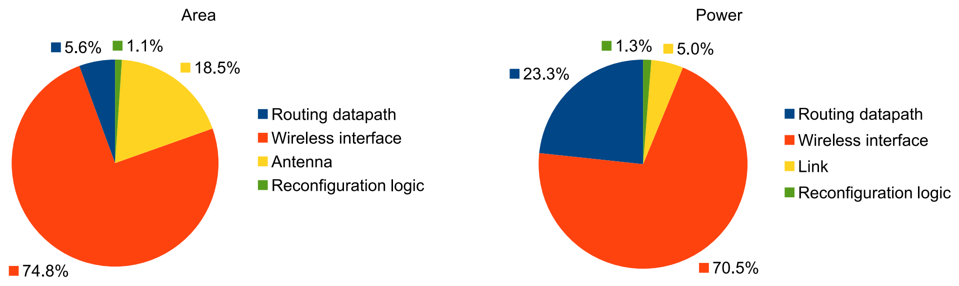


Fig. 9. Area and power breakdown of the radio hub.

channel. We assume a token packet transmission policy: When a given radio hub owns the token, it can start a transmission on the channel, keeping the token until the packet transmission has been completed. At the end of the packet transmission, the token is released and given the next radio hub belonging to the token ring.

We do not go into the internal details of the architecture of the radio hub, as it has been extensively presented [7]. Instead, we focus on the Wireless Interface (WI), which has been augmented with a circuitry that allows it to use different transmitting power levels based on the required reliability level. The WI consists of three main parts: antenna, analog, and digital modules. The digital domain includes the serializer and deserializer modules. The analog domain includes the Amplitude-Shift Keying or On-Off Keying (ASK-OOK), which is the most used modulation technique in mm-wave WiNoCs [13–15]. Starting from this reference architecture, we have replaced the PA with a variable power amplifier [21, 25, 26] supporting multiple transmitting power levels. Like in the router, the reliability level encoded into the header flit of the packet is used, in this case, to select the appropriate transmitting power by means of a lookup table. Figure 9 shows the breakdown of area and power for the radio hub. The additional module implementing the tunable transmitting power mechanism is referred to as reconfiguration logic and, as it can be observed, it accounts for a very limited fraction of the overall area and power of the radio hub.

Without loss of generality, here, we consider two transmitting power levels—namely, high (or nominal) and low—that have been computed as follows: We defined two reliability levels, high and low, corresponding to a BER of 10^{-12} and 10^{-6}, respectively. From the BER, we compute the minimum power to be received at the terminal of the receiver antenna as:

$$P_r = \left[Q^{-1}(BER)\right]^2 N_0 R_b, \tag{3}$$

where R_b is the data rate, N_0 is the transceiver noise spectral density (noise introduced by the transceiver), and the Q function is the tail probability of the standard normal distribution that is defined in Equation (2). The minimum transmitting power needed for reaching the receiving antenna for the considered BER is computed as:

$$P_t = P_r / G_a, \tag{4}$$

where G_a is the attenuation introduced by the wireless medium, and it is estimated by means of Equation (5):

$$G_a = \frac{|S_{12}|}{(1 - |S_{11}|)(1 - |S_{22}|)}, \tag{5}$$

where S_{11}, S_{12}, and S_{22} are the scattering parameters gathered by using field solver simulation tools [17]. We considered a zigzag antenna modeled and characterized with Ansoft HFSS [2] (High Frequency Structural Simulator) for obtaining the scattering parameters to compute the wireless medium attenuation with Equation (5). Based on this, the minimum transmitting powers for two considered BER levels have been computed. We calculated the attenuation G_a assuming the worst case of communicating radio hub pairs. This guarantees that each radio hub—equipped with a configurable PA—will be able to use a transmission power corresponding to the maximum low/high reliability levels of BER values. Specifically, considering a data rate of 16 Gbps, we found an energy per bit of 1.47 pJ/bit and 1.0 pJ/bit for the high and low transmitting power levels, respectively.

In our work, we considered a WI supporting a single wireless channel. The design and analysis of the proposed technique applied to a WI supporting multiple wireless channels is left for future work.

4.3 Remarks on the Use Virtual Channels

In this work, we consider WiNoC architectures that do not use virtual channels (VCs). In fact, the basic assumption is that all the flits of the same packet travel in sequence to the same wired/wireless link without being interleaved with flits belonging to other packets. Based on this, the link is configured by the header flit to work at a specific reliability level, and it maintains its configuration for all the flits of the packet.

For WiNoC architectures with VCs, the proposed technique can be extended by applying the configuration process at flit granularity. Specifically, when a flit of VC$_j$ moves to LC stage, the action to be taken depends on the type of the current flit and on the type of the flit in this stage at the previous cycle and related to a different VC$_i$. Flit types that determine a specific action are header flit, body flit of a high reliability packet, and body flit of a low reliability packet. Table 4 reports all the cases that may occur, and the related action taken, when a flit related to VC$_j$ at clock cycle k is preceded by a flit related to VC$_i$ at clock cycle $k - 1$.

Figure 10 shows the temporal diagram of the pipeline for a WiNoC with two VCs for several cases reported in Table 4. The cases highlighted in green correspond to the LC stage in which the link is reconfigured. When, in the LC stage, there are two consecutive header flits (case 1), or two consecutive body flits belonging to a high reliability packet (case 7), or a header flit alternating with a body flit belonging to a high reliability packet (case 3 and case 5), the link remains configured as VDDH. Such a situation is represented in the figure as "_ (H$_i$ →H$_j$)." When, in the LC stage,

Table 4. Action to Be Taken in LC Stage of the Pipeline in Case of VC-based Implementation

	VC_i (clock k-1)	VC_j (clock k)	Action taken
1	Header	Header	Link remains at VDDH
2	Header	Body (Low reliability)	Link configuration changes from VDDH to VDDL
3	Header	Body (High reliability)	Link remains at VDDH
4	Body (Low reliability)	Header	Link configuration changes from VDDL to VDDH
5	Body (High reliability)	Header	Link remains at VDDH
6	Body (Low reliability)	Body (Low reliability)	Link remains at VDDL
7	Body (High reliability)	Body (High reliability)	Link remains at VDDH
8	Body (Low reliability)	Body (High reliability)	Link configuration changes from VDDL to VDDH
9	Body (High reliability)	Body (Low reliability)	Link configuration changes from VDDH to VDDL

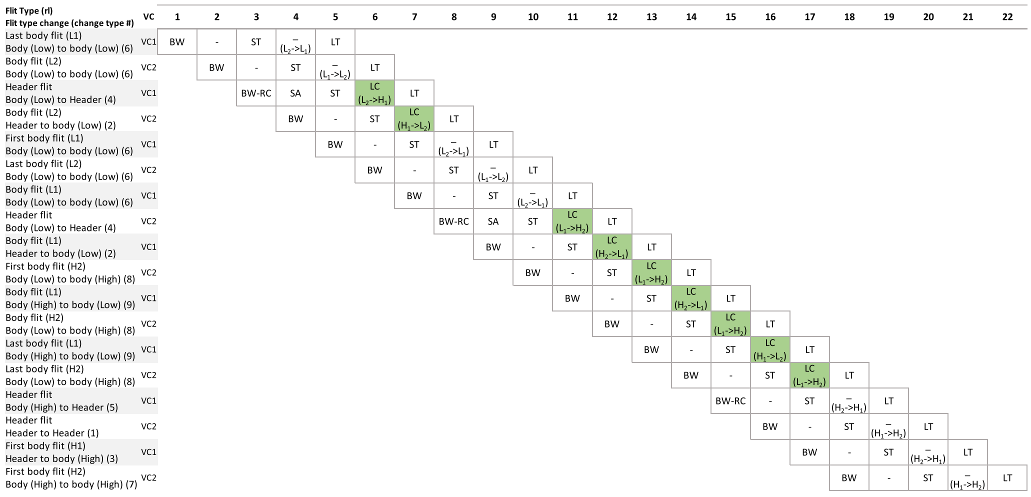

Flit Type (rl)
Flit type change (change type #)
VC
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
Last body flit (L1) Body (Low) to body (Low) (6) VC1 BW - ST _ (L2->L1) LT
Body flit (L2) Body (Low) to body (Low) (6) VC2 BW - ST _ (L1->L2) LT
Header flit Body (Low) to Header (4) VC1 BW-RC SA ST LC (L2->H1) LT
Body flit (L2) Header to body (Low) (2) VC2 BW - ST LC (H1->L2) LT
First body flit (L1) Body (Low) to body (Low) (6) VC1 BW - ST _ (L2->L1) LT
Last body flit (L2) Body (Low) to body (Low) (6) VC2 BW - ST _ (L1->L2) LT
Body flit (L1) Body (Low) to body (Low) (6) VC1 BW - ST _ (L2->L1) LT
Header flit Body (Low) to Header (4) VC2 BW-RC SA ST LC (L1->H2) LT
Body flit (L1) Header to body (Low) (2) VC1 BW - ST LC (H2->L1) LT
First body flit (H2) Body (Low) to body (High) (8) VC2 BW - ST LC (L1->H2) LT
Body flit (L1) Body (High) to body (Low) (9) VC1 BW - ST LC (H2->L1) LT
Body flit (H2) Body (Low) to body (High) (8) VC2 BW - ST LC (L1->H2) LT
Last body flit (L1) Body (High) to body (Low) (9) VC1 BW - ST LC (H1->L2) LT
Last body flit (H2) Body (Low) to body (High) (8) VC2 BW - ST LC (L1->H2) LT
Header flit Body (High) to Header (5) VC1 BW-RC - ST _ (H2->H1) LT
Header flit Header to Header (1) VC2 BW - ST _ (H1->H2) LT
First body flit (H1) Header to body (High) (3) VC1 BW - ST _ (H2->H1) LT
First body flit (H2) Body (High) to body (High) (7) VC2 BW - ST _ (H1->H2) LT

Fig. 10. Temporal diagram of the pipeline for a WiNoC with two VCs for several cases reported in Table 4.

there are two consecutive body flits belonging to a low reliability packet (case 6), the link remains configured as VDDL. Such a situation is represented in the figure as "_ (L_i →L_j)." When, in the LC stage, a header flit or a body flit of a high reliability packet is followed by a body flit of a low reliability packet (case 2 and case 9) the link is reconfigured to switch from VDDH to VDDL. Such a situation is represented in the figure as "LC (H_i →L_j)." Finally, in the LC stage, when a body flit of a low reliability packet is followed by a header flit or by a body flit of a high reliability packet (case 4 and case 8), the link is reconfigured to switch from VDDL to VDDH. Such a situation is represented in the figure as "LC (L_i →H_j)."

5 EXPERIMENTAL ANALYSIS

In this section, we assess xWiNoC on a set of applications belonging to different domains, focusing on the trade-off energy saving vs. application results quality degradation.

5.1 Evaluation Flow

The evaluation flow is shown in Figure 11. It gets as inputs the application and the application quality criterion and provides in output the application output quality and the energy consumption.

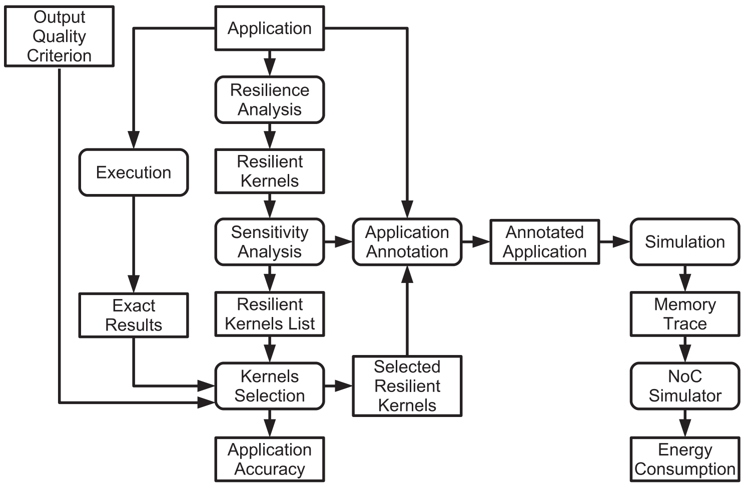


Fig. 11. Evaluation flow.

The application quality criterion depends on application quality metrics that are, of course, application-specific.

First, a resilience analysis of the application is performed to classify the data structures in the application in two classes: resilient and sensitive. To do this, the application is first profiled to detect its computational kernels. Specifically, we consider innermost loops that account for over a specified fraction of the program's execution time (we used 1% in the experiments) as atomic kernels [11]. As the program executes over the provided input dataset, we add random errors to the program data structures that are modified in a kernel and used outside it, i.e., the kernel's outputs. If the application crashes or hangs, then the kernel is marked as sensitive; otherwise, it is marked as potentially resilient.

The output of the resilience analysis is a set of potential resilient computational kernels. These are used as input for a sensitivity analysis [12] task aimed at identifying an ordering of the resilient kernels based on their impact on the application output quality. Data involved in each kernel are perturbed by injecting errors that reflect the lowest reliability level. Then, the variation on the results is observed and used to sort the kernels based on such variation.

The output of the sensitivity analysis is used by the kernel selection task aimed at selecting the set of data structures to be annotated as resilient and the corresponding reliability level. To do this, it starts to inject errors to the data structures of the least sensitive kernel. The errors injected reflect the property of the lowest reliability level. If the impact on the application output quality exceeds the user-defined output quality criterion, then the perturbation is repeated by injecting errors reflecting the property of the second lowest reliability level and so on until the output quality criterion is satisfied. Then, the second lowest sensitive kernel is explored, and so on until all the kernels have been explored or any additional kernel perturbation does not allow to meet the quality criterion. The output of this phase is the list of the kernels that will be approximated along with the reliability level to be used and the corresponding application results quality level.

To determine the energy consumption, we proceed as follows: The data structures involved in the selected kernels are pragma annotated with the appropriate reliability levels provided by the previous phase. Then, the annotated application is simulated for obtaining the memory reference trace file. The latter is used as input for the WiNoC simulator [7], which provides in output an estimation of communication energy figures.

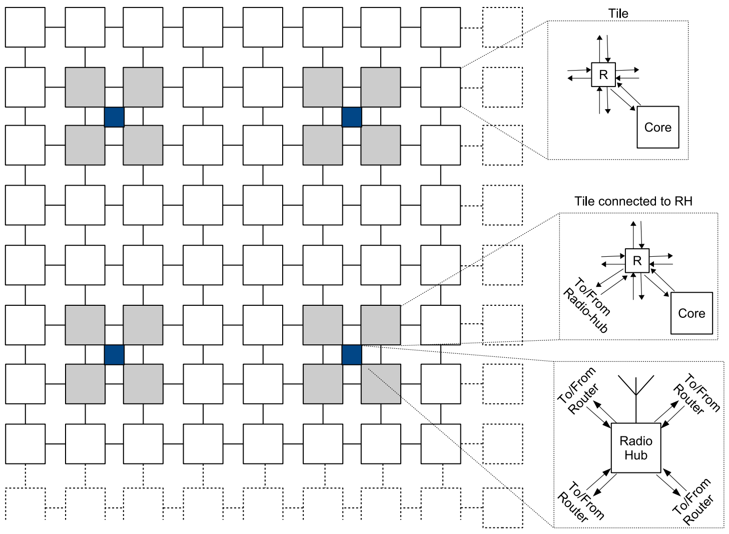


Fig. 12. Considered topology for xWiNoC.

5.2 Network Topology and Simulation Setup

Although xWiNoC is agnostic with respect to the network topology, in the experiment, we consider a 256-node network shown in Figure 12. The considered topology is based on a 16×16 mesh that is partitioned into 16 4×4 sub-networks. There is a radio hub in each sub-network. The radio hub is placed into the center of the sub-network and connected to the four neighboring routers.

We have implemented the topology in Noxim [7] and updated all the energy costs using the data reported in Section 4. To simulate the application, we used a Graphite Multicore Simulator [24] that has been extended to trace all the memory references. Finally, the conversion of the memory reference traces to messages injected into the NoC has been carried out by means of the technique presented in Reference [8]. Both the power (static and dynamic) and latency overhead due to the logic enabling the proposed technique and implemented into the router and wireless interface is taken into account in all the experiments reported in the article.

5.3 Applications and Sensitivity Analysis

We assess xWiNoC on a set of applications belonging to AxBench benchmark suite [50], which collects a set of applications and kernels typical of the RMS domain. Table 5 reports a brief description of the applications along with the input dataset and the considered output error metric.

According to the evaluation flow of Figure 11, let us start by showing in Figure 13 the normalized sensitivity level and the fraction of memory usage of each data structure involved in the computational kernels of the considered applications. For the sake of reproducibility of the experiments, we reported the original name of the data structures as it appears in the source code. Table 6 reports, for each application, the selected error-resilient data structures along with a short description and their fraction on the total accesses. We ordered the data structures according to their normalized sensitivity levels. The graph also shows the cumulative fraction of memory usage of each data structure. We plot the cumulative value as, in the rest of the experiments, we consider several versions of the same application in which we apply the proposed technique to the first data structure, the first and second data structures, and so on in the same order as reported in the

Table 5. Applications Considered in the Experiments along with Their Input Dataset and Error Metric

Name	Domain	Description	Input dataset	Error metric
`blackscholes`	Financial analysis	Mathematical model for finance	100K options	Average Relative Error
`inversek2j`	Robotics	Inverse kinematics for two-joint arm	300K (x,y) random coordinates	Average Relative Error
`jmeint`	3D gaming	Triangle intersection detection	100K random pairs of 3D coordinates	Miss Rate
`jpeg`	Compression	JPEG encoding	512×512 pixel color image	Image Diff (RMSE)
`k-means`	Machine learning	K-means clustering	262,144 pairs of random (r,g,b) values	Image Diff (RMSE)
`sobel`	Image processing	Sobel edge detector	512×512 pixel color image	Image Diff (RMSE)

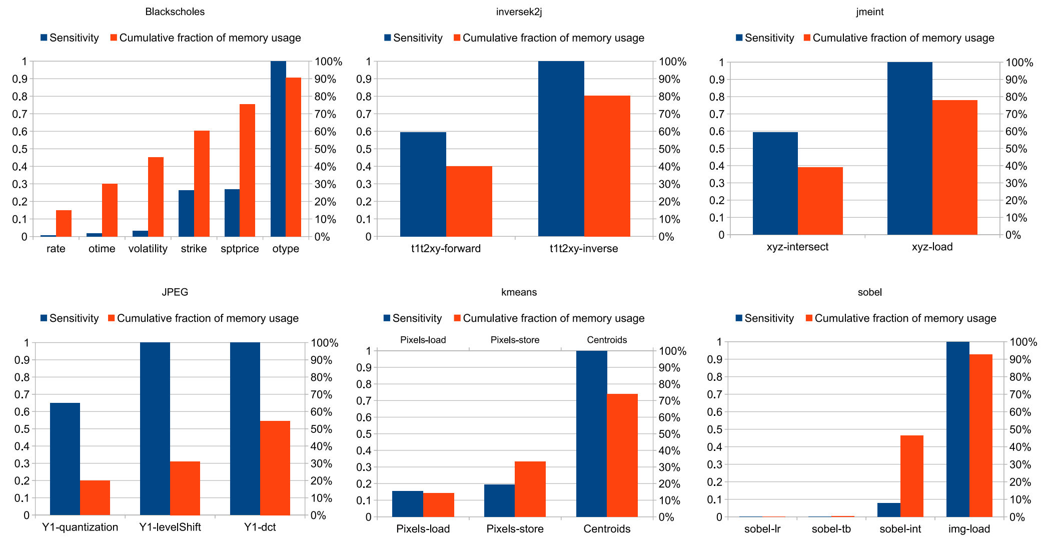


Fig. 13. Normalized sensitivity level and fraction of memory usage of each data structure involved in the computational kernels of the application.

graph. The order reflects the increasing level of sensitivity of the considered data structures. As it can be observed, the variance of the sensitivity level of the different data structures spans over a wide range, and the fraction of memory accesses that fall into the considered data structures is relevant (e.g., from 15% to 90% for `blackscholes`). Such analysis suggests the potential impact of the technique. In fact, as the selected resilient data structures capture a high percentage of the memory accesses, the corresponding memory-induced communications can be performed with a reduced reliability level with a consequent energy saving.

The use of sensitivity analysis for ranking the data structures to be transmitted with a low reliability level can be exploited for determining the appropriate reliability level to be used in case multi-reliability levels are available. For instance, one possible technique to be investigated might be that of using the sensitivity level of each data structure as a parameter for selecting the appropriate reliability level; that is, using the low reliability levels for low sensitive data structures and high reliability levels for high sensitive data structures.

Table 6. Description of the Data Structures Marked as Resilient for the Different Applications and Fraction of Total Accesses

Name	Description	Fraction accesses
	`blackscholes`	
rate	Array of rates of return on the riskless asset	15%
otime	Array of time to maturity or option expiration in years	15%
volatility	Array of average future volatility of the underlying asset	15%
strike	Array of strike prices	15%
sptprice	Array of spot prices	15%
otype	Array of option types	15%
	`jpeg`	
Y1-quantization	Pixels of the image manipulated by the quantization module of the encoder	20%
Y1-levelShift	Pixels of the image manipulated by the level shift module of the encoder	11%
Y1-dct	Pixels of the image manipulated by the DCT module of the encoder	24%
	`k-means`	
Pixels-load	Pixels of the input image	14%
Pixels-store	Pixels of the output image	19%
Centroids	Array of clusters' centroids	41%
	`jmeint`	
xyz-intersect	Array storing the triangles' coordinates used by the intersect function	39%
xyz-load	Array storing the triangles' coordinates used in the load function	39%
	`inversek2j`	
t1t2xy-forward	Array storing the angles of the arms used during the forward computation	40%
t1t2xy-inverse	Array storing the angles of the arms used during the inverse computation	40%
	`sobel`	
sobel-lr	Pixels accessed during the application of the filter at the boundary (left and right side) of the image	0.2%
sobel-tb	Pixels accessed during the application of the filter at the boundary (top and bottom side) of the image	0.2%
sobel-int	Pixels accessed during the application of the filter at inner part of the image	46%
img-load	Pixels accessed during the load phase	46%

5.4 Energy vs. Quality Trade-off

The primary goal of any approximate computing technique is to provide an appropriate trade-off between the considered optimization metrics. In our case, we consider the communication energy consumption and the impact on the application result quality degradation. In this subsection, we explore the trade-off energy vs. error using a specific error metric for each application. For each application, Figure 14 shows six graphs corresponding to the six considered xWiNoC configurations as follows: RLink1, RLink2, and RLink3 are xWiNoC configurations in which the wired network uses the reconfigurable link architectures RLink1, RLink2, and RLink3 described in Section 4.1 (see Table 2), respectively. RLink1+RW, RLink2+RW, and RLink3+RW are the xWiNoCs configured to use the reconfigurable link architectures RLink1, RLink2, and RLink3 and the tunable transmitting power wireless interface described in Section 4.2. Each graph shows on the x-axes the percentage of energy saving with respect to a baseline conventional WiNoC architecture in which a traditional link architecture (full swing link) and a traditional wireless interface (with single transmitting power) are used. The y-axes report the absolute error of the application based on the specific error metric as reported in Table 5. In each graph, a number of points related to the number of resilient data structures selected for that application are shown. Each point corresponds to a particular configuration in which a sub-set of the selected data structures have been marked as resilient. Specifically, the resilient data structures are sorted in ascending order based on their sensitivity

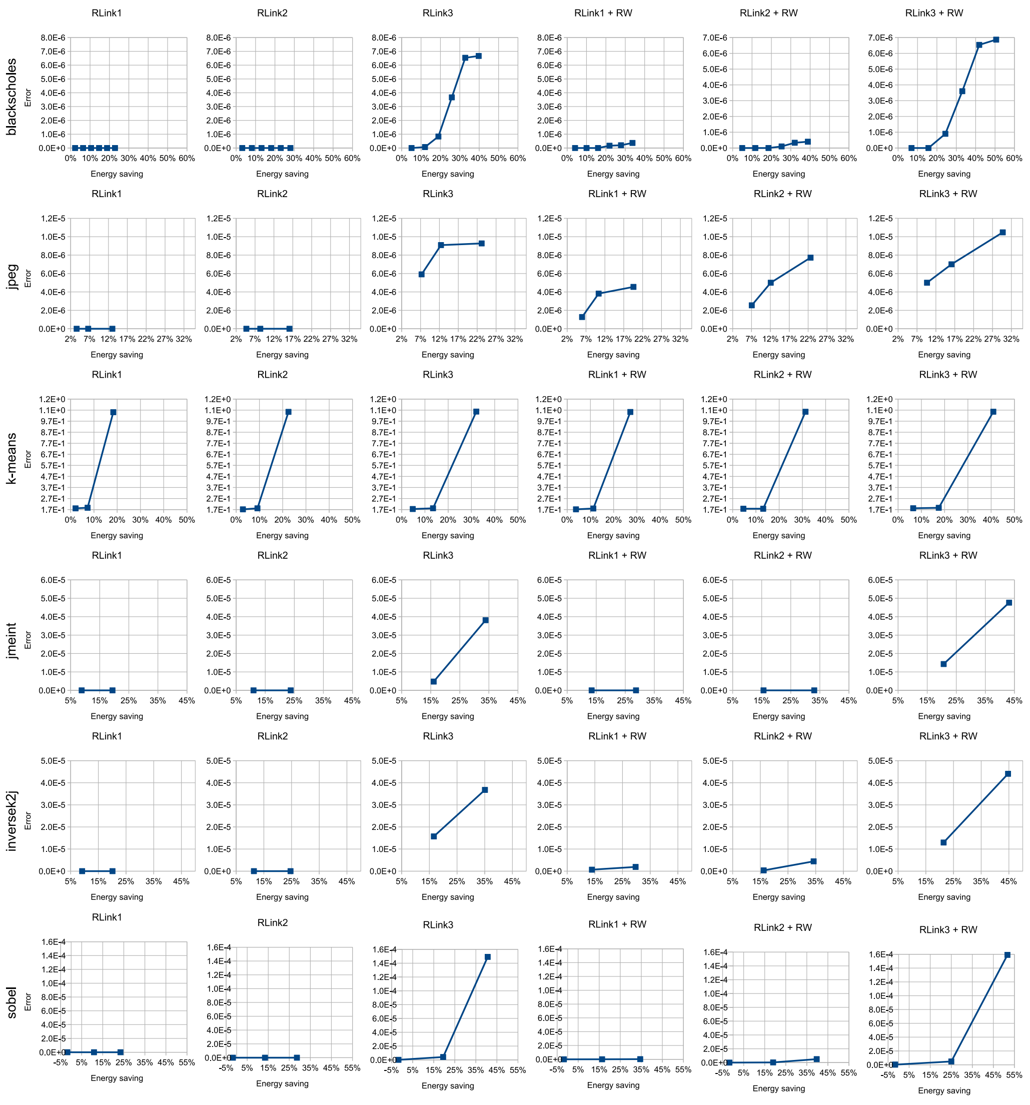


Fig. 14. Energy saving vs. error trade-off for different applications, different configurations of the xWiNoC, and different data structures marked as resilient.

level. Then, the first point in each graph corresponds to the configuration in which the first data structure (i.e., that one with the lowest sensitivity level) is marked as resilient. The second point in the graph corresponds to the configuration in which the first and the second data structure (i.e., those ones with the lowest and second lowest sensitivity level) are marked as resilient, and so on. The last point in the graph is, therefore, the configuration in which all the selected data structures are marked as resilient.

For each network configuration, the general trend is the increase of energy saving and the increase of error as the number of data structures marked as resilient increases. In several cases, for some configurations, the degradation of the application output is negligible. For instance, in `blackscholes`, the output error is negligible for configurations RLink1, RLink2, RLink1+RW, and

RLink2+RW although with 25%, 30%, 35%, and 40% energy saving, respectively. For configurations RLink3 and RLink3+RW, the error rapidly increases, starting from the third sensitive data structure marked as resilient. However, even in the worst case, the error is below 7×10^{-6}, which can be considered negligible, as, on average, options prices are in the order of cents.

For what it concerns `jpeg`, the quality of the output image is unaffected when assessed by the human sensory pathways. The interested reader is invited to visit Reference [33] for a gallery of the output images generated by each xWiNoC configuration. From a quantitative analysis, the image diff error metric based on the root mean square error between RGB values of the pixels is in the order of 10^{-6}. Such value, if used for computing the PSNR (ratio of the largest pixel value and the RMS of error), provides a value greater than 30 dB that, based on Reference [47], represents a threshold for an acceptable image quality. In terms of energy, up to 30% of energy saving is observed for configuration RLink3+RW.

In `kmeans`, we considered three resilient data structures. However, the impact on the error metric due to the most sensitive data structure (centroids) exceeds the minimum classification error we considered for this application, which is 20%. Thus, due to the limited fraction of communications induced by the data structures marked as resilient, the energy saving is limited, about 20%.

In `jmeint`, the two selected data structures have a limited impact on the error metric. Even with the more aggressive configuration, RLink3+RW, characterized by the highest BER, the output error measured by the miss rate in detecting a collision is in the order of 10^{-5}, which can be considered negligible. The overall communication energy saving is close to 45%, and it is limited by the more computational—rather than communication—intensive nature of the application. Similar to `jmeint`, even in `inversek2j`, we found that the two selected data structures have a low impact on the error metric. Up to 45% of energy saving is observed for RLink3+RW configuration with a negligible variation in the computed angles of the two-joint arm.

In `sobel`, it is interesting to note that for configurations in which only the least sensitive data structure is considered, the energy saving is negative; that is, the overhead introduced by the proposed techniques is higher than the energy saving. This is due to the fact that the fraction of communications related to such data structure is too low to be exploited. By marking the second data structure as resilient, the energy saving starts to increase up to 52% when all the selected data structures are marked as resilient and for configuration RLink3+RW. In all the cases, the error is negligible and below 10^{-4} based on the same metrics used for `jpeg`.

5.5 Pareto-optimal Configurations

For the sake of summarizing the results presented in the previous subsection, Figure 15 shows the Pareto-front energy saving vs. error annotated with the corresponding Pareto-optimal configurations for all the applications. For each application, we reported only the non-dominated configurations of Figure 14. In addition, we have annotated each point in the graph with the corresponding configuration formed by the xWiNoC configuration, followed by the list of data structures that have been marked as resilient. As can be noticed, there is not a single optimal configuration, but the number and the type of configurations belonging to the Pareto-set depends on the specific application. However, some common features can be extrapolated. Configuration RLink3+RW, in all the cases, provides the highest energy saving, although with the highest error. With the exception for `k-means`, such configuration has not exceeded the considered output quality criterion. This result, however, was expected, as RLink3+RW is the most aggressive configuration in which the lowest voltage swing level and the lowest transmitting power are used, resulting in the highest BER for communications. On the opposite side, RLink1 is the configuration that, when present in the Pareto set, determines the lowest impact on the error. Even in this case, such behavior was

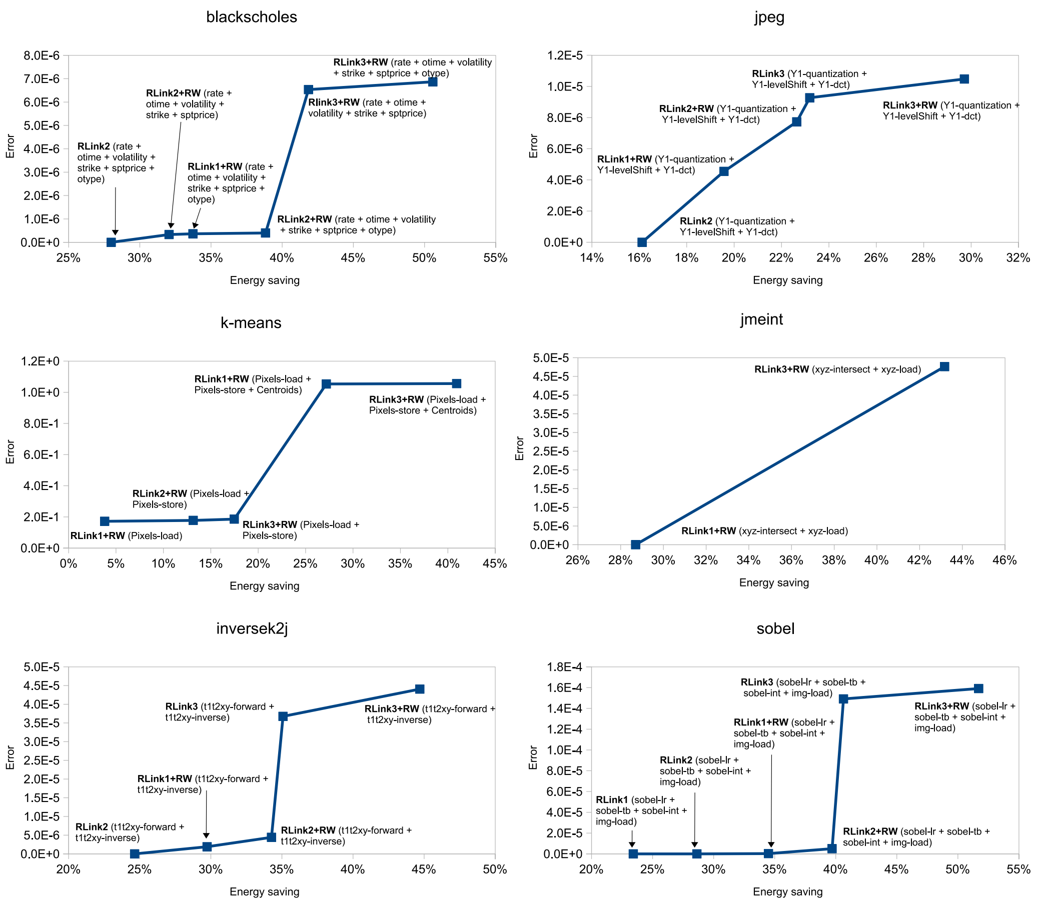


Fig. 15. Pareto-front energy saving vs. error annotated with the corresponding Pareto-optimal configurations.

expected, as the xWiNoC configured to use RLink1 for its wired network fabric exposes an average BER that does not deviate too much from that of a conventional WiNoC.

The sharp variation of the curves for `blackscholes`, `inversek2j`, `sobel`, and `k-means` can be explained as follows: In `blackscholes`, the increase of the BER, when passing from configuration RLink2 to RLink3, has a stronger impact on the application accuracy than the inclusion of the additional data structure (`otype`). Contrarily, in `k-means`, the inclusion of the `centroid` data structure into the set of resilient data structures has a strong impact on the accuracy of the application due to its high sensitivity level, as shown in Figure 13. In `inversek2j` and `sobel`, the low utilization of the wireless network makes the application more sensitive to the increase of the BER due to the use of RLink3 rather than the use of a low reliability level when transmitting the resilient data structures through the wireless sub-network. The abrupt curves in the Pareto fronts are due to the fact that we have to approximate all or none of each data structure. Additional Pareto configurations might be obtained by approximating some proportion of the data structure.

To further summarize the overall results, Figure 16 shows the percentage presence of xWiNoC configurations in the Pareto set found for the different applications. The xWiNoC configurations using both the reconfigurable voltage swing and tunable transmitting power technologies (i.e.,

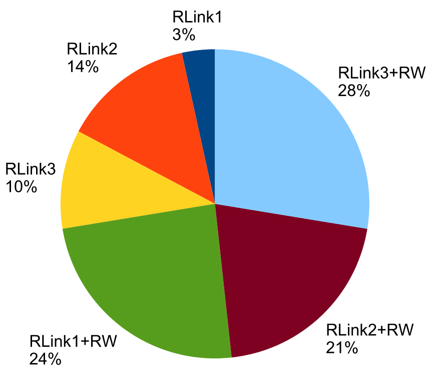


Fig. 16. Percentage of presence of different xWiNoC configurations in the Pareto set of the considered applications.

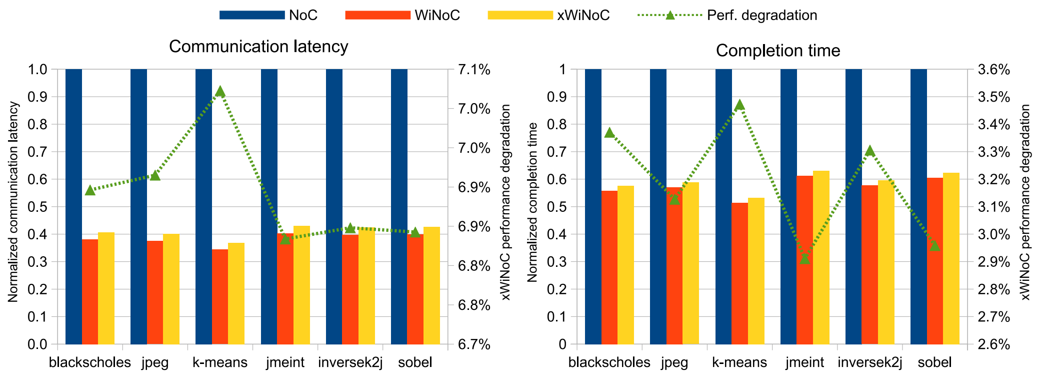


Fig. 17. Communication latency (left) and completion time (right) normalized with respect to a wired NoC.

RLink* + RW) are representative of the 72% of the configurations in the Pareto sets. Thus, based on the above analysis, we can conclude that the best trade-offs between communication energy saving and degradation of the application output are found by joint-exploiting the reconfigurable link architecture and the tunable transmitting power–based wireless interface while designing the proposed xWiNoC architecture.

5.6 Performance Analysis

In the previous analysis, the focus has been put on energy consumption. In this subsection, we discuss the impact on communication performance due to the use of the proposed technique. Figure 17 (left) shows, for each application, the average communication latency for the conventional WiNoC and the xWiNoC normalized by the communication latency of a wired NoC. The graph also reports on the secondary y-axes the performance degradation of xWiNoC with respect to WiNoC. The xWiNoC is configured for the most aggressive trade-off energy vs. error—namely, combination RLink3+WR. As it can be observed, the impact on the communication performance is on average 7% compared to an average energy saving of 43%.

It should be pointed out that the performance degradation shown in Figure 17 (left) refers to the fraction of the time spent in communication. However, the ultimate performance metric is the completion time of the application. Figure 17 (right) shows the completion time of the applications

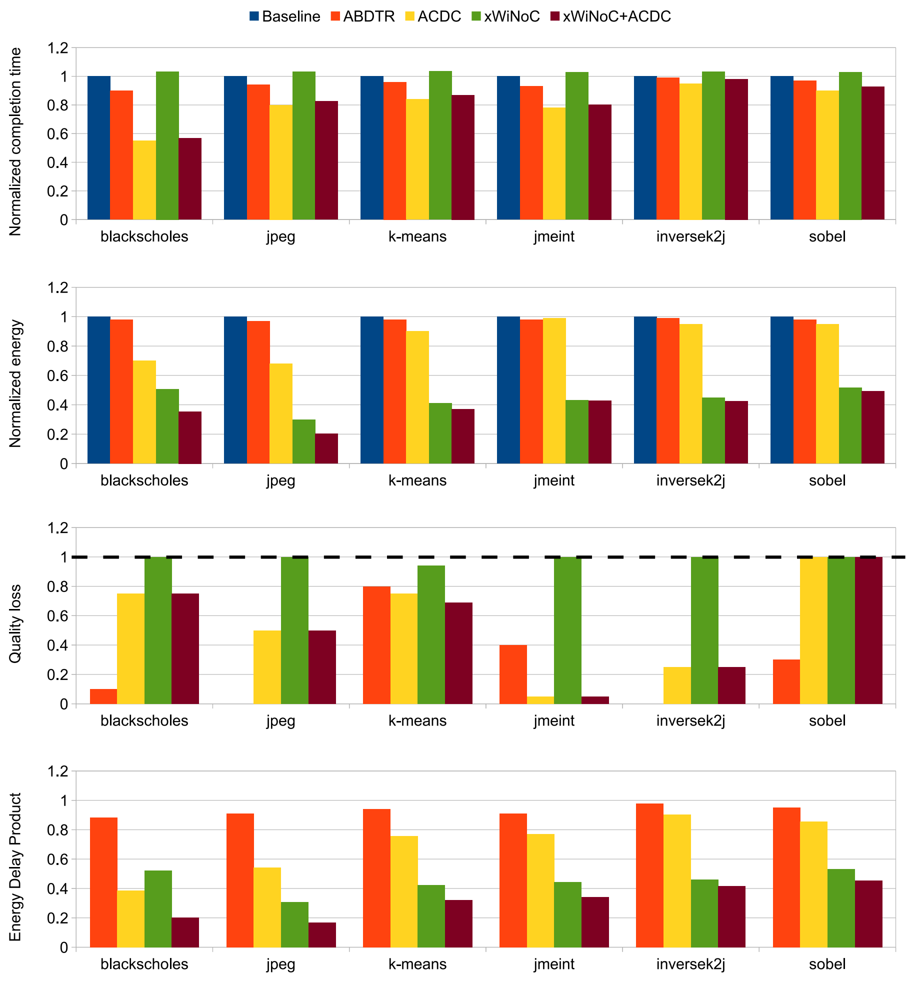


Fig. 18. Performance, energy, and accuracy comparison between different approximate communication techniques and their combination.

normalized by the completion time on a conventional wired NoC. It can be observed that the average performance degradation of completion time is as limited as 3% once again for the most aggressive xWiNoC configuration—namely, RLink3+WR.

5.7 Comparison with Other Approximate Communication Techniques

In this section, we compare the proposed technique with two recent approximate communication techniques: ABDTR [46] and ACDC [48]. For each considered application, we compare ABDTR and ACDC with the xWiNoC configured with the most aggressive trade-off energy vs. error—namely, combination RLink3+WR—as we did in the performance analysis. Further, we also consider the case in which xWiNoC is combined with ACDC aimed at showing the generality of the proposed technique that makes it suitable to be coupled with other existing techniques.

Figure 18 shows the normalized execution time, energy, quality loss, and energy delay product when the considered approximate communication techniques are applied to the baseline WiNoC. Contrary to xWiNoC, both ABDTR and ACDC improve performance. Thus, when we apply ACDC on top of xWiNoC the small performance degradation introduced by xWiNoC is overcompensated by the performance improvement of ACDC, thus resulting in overall performance improvement. Regarding energy consumption, the proposed technique provides higher energy savings. This is due to the fact that both ABDTR and ACDC have been proposed as techniques for communication performance improvement in which energy reduction is only a derived effect; that is, a consequence of the execution time reduction. Regarding the accuracy, in this experiment, we used the same metrics considered in Reference [48] in which the dashed line in the quality loss evaluation dictates the quality requirement, which is normalized to 1. As it can be observed, the proposed technique does not introduce any appreciable quality loss (according to the considered metric) as compared to the other approximate communication techniques. Regarding the energy consumption, as compared to ACDC, xWiNoC and xWiNoC+ACDC, on average, improve the energy saving by 49% and 57%, respectively. Finally, considering the energy delay product (EDP), it can be observed as xWiNoC outperforms ABDTR and ACDC in almost all the considered applications. Only for `blackscholes` the EDP for xWiNoC is a bit higher than that of ACDC due to the fact that the average communication distance is low, thus making less effective the proposed technique.

5.8 Case Study: Deep Neural Network Accelerator

Many applications in the realm of natural language processing, visual data processing, and speech and audio processing, are nowadays very effectively implemented by means of Convolutional Neural Networks (CNNs) [37]. The increasing use of deep learning–based techniques in many domains is pushing research and industry toward the design and development of high performance, low cost, high energy efficient Artificial Neural Network (ANN) accelerators [9]. The reference architecture of such accelerators is a manycore system in which cores are high parallel arithmetic specialized Processing Elements (PEs) interconnected by means of a NoC [9, 52]. Neural networks possess a forgiving nature towards the imprecision of their weights [53]. Thus, in this section, we assess the energy vs. accuracy trade-off when we map several different CNNs into xWiNoC.

The architecture of the CNN accelerator we considered is inspired by Simba [52]. Specifically, we considered a 576-node WiNoC partitioned as 6×6 regions, each of which is a 4×4 mesh-based wireline NoC. In each region there is a radio hub that allows wireless inter-region communications. Intra-region communication is carried out by means of the wireline NoC. Each PE includes 8 KB of local memory and 8 parallel lanes of vector Multiply-Accumulate (MAC) units. Each vector MAC performs an 8-way dot product and accumulates the partial sum into the accumulation buffer every cycle. The experimental platform is a simulated parameterized NoC-based Deep Neural Network that allows to assess different architectural configurations in terms of performance and energy [3]. The RTL models of the PE and router have been synthesized with Synopsis Design Compiler and mapped on a 45 nm CMOS LVT library from Nangate [43]. The links have been modeled with HSPICE and the parasitics extraction from layout has been made using Cadence Virtuoso. The power figures collected by the circuit level analysis have been used to back-annotate the cycle-accurate NoC simulator [7]. For both local and main memory, we used CACTI [31] to estimate the energy consumption (both leakage and dynamic) and timing information.

We considered six representative network models: LeNet-5, AlexNet, VGG-16, MobileNet, Inception-v3, and ResNet, which cover a wide spectrum in terms of complexity both in the number of layers and number of parameters. Figure 19 shows the normalized inference energy consumption and top-5 accuracy for the different CNNs mapped on WiNoC and xWiNoC. There are primarily three kinds of traffic: scatter, gather, and local. Scatter flow is the traffic distribution from memory to PEs to deliver input feature maps and weights. Based on the considered parallelization scheme, feature map distribution results in broadcast communication flows, whereas weight distribution results in unicast communication flows. Gather flow is the traffic distribution from

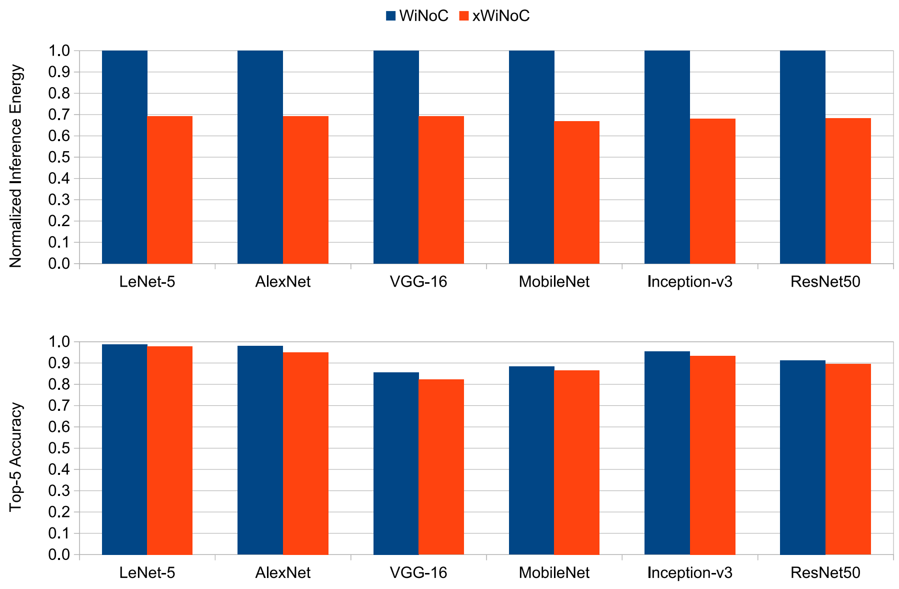


Fig. 19. Normalized inference energy and top-5 accuracy for different CNNs mapped on WiNoC and xWiNoC.

PEs to memory for storing back the output feature map. The amount of local memory in each PE determines the gather traffic volume. Finally, local flow is the traffic distribution among PEs (broadcast traffic) due to dispatch of the computed output feature map channels among PEs. All the communications carrying filters, weights, and feature maps are marked as resilient and thus transmitted at a low reliability level. As it can be observed, on average, 32% of total energy can be saved with less than 3% top-5 accuracy degradation.

6 CONCLUSION

We extended the approximate computing paradigm to the communication sub-system of many-core architectures in which a NoC is used as a communication backbone. At the circuit level, we proposed the use of reconfigurable voltage swing links for the wired NoC and tunable transmitting power wireless interfaces for the WiNoC. The first ones can be configured to work at different voltage swing levels for wired communications, whereas the second ones allow using different transmitting power levels for radio communications. In both cases, a trade-off between energy consumption and communication reliability (measured in terms of communication bit error rate) is exposed to the programmer. The programmer, by means of pragma annotation of the source code, marks those data structures that are error-resilient. The communications induced by the access to such data structures are then operated by means of less reliable yet energy-efficient transmissions. A set of experiments carried out on applications belonging to different domains has been presented in which the trade-off energy saving vs. application output degradation has been discussed. The obtained results have shown that up to 50% of communication energy saving can be obtained with a negligible impact on the application output quality.

Future work along this direction includes the definition of an automatic technique for discovering the resilient data structures of an application and the selection of the most appropriate reliability level to be used for their induced communications.

REFERENCES

[1] Akram Ben Ahmed, Daichi Fujiki, Hiroki Matsutani, Michihiro Koibuchi, and Hideharu Amano. 2018. AxNoC: Low-power approximate network-on-chips using critical-path isolation. In *Proceedings of the IEEE/ACM International Symposium on Networks-on-Chip (NOCS'18)*. IEEE, 1–8.

[2] ANSYS. 2014. Ansoft HFSS. Retrieved from http://www.ansys.com/.

[3] Giuseppe Ascia, Vincenzo Catania, John Jose, Salvatore Monteleone, Maurizio Palesi, and Davide Patti. 2019. Analyzing networks-on-chip based deep neural networks. In *Proceedings of the International Symposium on Networks-on-Chip.*

[4] Giuseppe Ascia, Vincenzo Catania, Salvatore Monteleone, Maurizio Palesi, Davide Patti, and John Jose. 2018. Improving energy consumption of NoC based architectures through approximate communication. In *Proceedings of the Mediterranean Conference on Embedded Computing (MECO'18)*. IEEE, 1–4.

[5] Filipe Betzel, Karen Khatamifard, Harini Suresh, David J. Lilja, John Sartori, and Ulya Karpuzcu. 2018. Approximate communication: Techniques for reducing communication bottlenecks in large-scale parallel systems. *ACM Comput. Surv.* 51, 1 (Jan. 2018), 1:1–1:32.

[6] Rahul Boyapati, Jiayi Huang, Pritam Majumder, Ki Hwan Yum, and Eun Jung Kim. 2017. APPROX-NoC: A data approximation framework for network-on-chip architectures. *SIGARCH Comput. Archit. News* 45, 2 (June 2017), 666–677.

[7] Vincenzo Catania, Andrea Mineo, Salvatore Monteleone, Maurizio Palesi, and Davide Patti. 2016. Cycle-accurate network on chip simulation with Noxim. *ACM Trans. Model. Comput. Simul.* 27, 1 (Nov. 2016), 4:1–4:25.

[8] Vincenzo Catania, Salvatore Monteleone, Maurizio Palesi, and Davide Patti. 2018. Packetization of shared-memory traces for message passing oriented NoC simulation. In *High Performance Computing*. Springer International Publishing, Switzerland, 311–325.

[9] Yu-Hsin Chen, Tien-Ju Yang, Joel Emer, and Vivienne Sze. 2019. Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices. *IEEE J. Emerg. Select. Topics Circ. Syst.* 9, 2 (June 2019), 292–308.

[10] Y. K. Chen, J. Chhugani, P. Dubey, C. J. Hughes, D. Kim, S. Kumar, V. W. Lee, A. D. Nguyen, and M. Smelyanskiy. 2008. Convergence of recognition, mining, and synthesis workloads and its implications. *Proc. IEEE* 96, 5 (May 2008), 790–807.

[11] Vinay K. Chippa, Srimat T. Chakradhar, Kaushik Roy, and Anand Raghunathan. 2013. Analysis and characterization of inherent application resilience for approximate computing. In *Proceedings of the 50th Design Automation Conference (DAC'13)*. ACM, New York, NY, Article 113, 9 pages. DOI: https://doi.org/10.1145/2463209.2488873.

[12] Jose B. Cruz. 1973. *System Sensitivity Analysis.* Hutchinson & Ross, Stroudsburg, PA.

[13] Sujay Deb, Kevin Chang, Miralen Cosic, Amlan Ganguly, Partha Pratim Pande, Deukhyoun Heo, and Benjamin Belzer. 2010. Enhancing performance of network-on-chip architectures with millimeter-wave wireless interconnects. In *Proceedings of the IEEE International Conference on Application-specific Systems Architectures and Processors*. IEEE, 73–80.

[14] Sujay Deb, Amlan Ganguly, Partha Pratim Pande, Benjamin Belzer, and Deukhyoun Heo. 2012. Wireless NoC as interconnection backbone for multicore chips: Promises and challenges. *IEEE J. Emerg. Sel. Topics Circ. Syst.* 2, 2 (2012), 228–239.

[15] Dominic DiTomaso, Avinash Kodi, Sava Kaya, and David Matolak. 2011. iWISE: Inter-router wireless scalable express channels for network-on-chips (NoCs) architecture. In *Proceedings of the Symposium on High Performance Interconnects.* IEEE Computer Society, 11–18.

[16] Vimuth Fernando, Antonio Franques, Sergi Abadal, Sasa Misailovic, and Josep Torrellas. 2019. Replica: A wireless manycore for communication-intensive and approximate data. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'19)*. 849–863.

[17] Brian A. Floyd, Chih-Ming Hung, and Kenneth K. O. 2002. Intra-chip wireless interconnect for clock distribution implemented with integrated antennas, receivers, and transmitters. *IEEE J. Solid-State Circ.* 37, 5 (2002), 543–552.

[18] Inigo Goiri, Ricardo Bianchini, Santosh Nagarakatte, and Thu D. Nguyen. 2015. ApproxHadoop: Bringing approximations to MapReduce frameworks. *SIGARCH Comput. Archit. News* 43, 1 (Mar. 2015), 383–397.

[19] Srinivasan Gopal, Pawan Agarwal, Joe Baylon, Luke Renaud, Sheikh Nijam Ali, Partha Pratim Pande, and Deukhyoun Heo. 2018. A spatial multi-bit sub-1-V time-domain matrix multiplier interface for approximate computing in 65-nm CMOS. *IEEE J. Emerg. Sel. Topics Circ. Syst.* 8, 3 (2018), 506–518.

[20] ITRS. 2015. International Technology Roadmap for Semiconductors 2.0. Retrieved from http://www.itrs2.net.

[21] S. Kaushik, M. Agrawal, H. K. Mondal, S. H. Gade, and S. Deb. 2017. Path loss-aware adaptive transmission power control scheme for energy-efficient wireless NoC. In *Proceedings of the International Midwest Symposium on Circuits and Systems (MWSCAS'17)*. IEEE, 132–135.

[22] Hiroki Matsutani, Michihiro Koibuchi, Daisuke Ikebuchi, Kimiyoshi Usami, Hiroshi Nakamura, and Hideharu Amano. 2011. Performance, area, and power evaluations of ultrafine-grained run-time power-gating routers for CMPs. *IEEE Trans. Comput.-aided Des. Integ. Circ. Syst.* 30, 4 (Apr. 2011), 520–533.

[23] Jiayuan Mengte, Anand Raghunathan, Srimat Chakradhar, and Surendra Byna. 2010. Exploiting the forgiving nature of applications for scalable parallel execution. In *Proceedings of the International Symposium on Parallel Distributed Processing*. IEEE, 1–12.

[24] Jason E. Miller, Harshad Kasture, George Kurian, Charles Gruenwald, Nathan Beckmann, Christopher Celio, Jonathan Eastep, and Anant Agarwal. 2010. Graphite: A distributed parallel simulator for multicores. In *Proceedings of the IEEE 16th International Symposium on High Performance Computer Architecture (HPCA'10)*. IEEE, 1–12.

[25] Andrea Mineo, Maurizio Palesi, Giuseppe Ascia, and Vincenzo Catania. 2016. Exploiting antenna directivity in wireless NoC architectures. *Microproc. Microsyst.* 43 (2016), 59–66.

[26] Andrea Mineo, Maurizio Palesi, Giuseppe Ascia, and Vincenzo Catania. 2016. Runtime tunable transmitting power technique in mm-wave WiNoC architectures. *IEEE Trans. Very Large Scale Integ. (VLSI) Syst.* 24, 4 (Apr. 2016), 1535–1545.

[27] Andrea Mineo, Maurizio Palesi, Giuseppe Ascia, Partha Pratim Pande, and Vincenzo Catania. 2016. On-chip communication energy reduction through reliability aware adaptive voltage swing scaling. *IEEE Trans. Comput.-aided Des. Integ. Circ. Syst.* 35, 11 (Nov. 2016), 1769–1782.

[28] Sparsh Mittal. 2016. A survey of techniques for approximate computing. *ACM Comput. Surv.* 48, 4 (Mar. 2016), 62:1–62:33.

[29] Hemanta Kumar Mondal, Shashwat Kaushik, Sri Harsha Gade, and Sujay Deb. 2017. Energy-efficient transceiver for wireless NoC. In *Proceedings of the International Conference on VLSI Design and the 16th International Conference on Embedded Systems (VLSID'17)*. 87–92.

[30] Alireza Monemi, Jia Wei Tang, Maurizio Palesi, and Muhammad N. Marsono. 2017. ProNoC: A low latency network-on-chip based many-core system-on-chip prototyping platform. *Microproc. Microsyst.* 54 (2017), 60–74.

[31] Naveen Muralimanohar, Rajeev Balasubramonian, and Norm Jouppi. 2007. Optimizing NUCA organizations and wiring alternatives for large caches with CACTI 6.0. In *Proceedings of the IEEE/ACM International Symposium on Microarchitecture (MICRO'07)*. IEEE Computer Society, Washington, DC, 3–14.

[32] Jacob Murray, Ryan Kim, Paul Wettin, Partha Pratim Pande, and Behrooz Shirazi. 2014. Performance evaluation of congestion-aware routing with DVFS on a millimeter-wave small world wireless NoC. *ACM J. Emerg. Technol. Comput. Syst.* 11, 2 (2014).

[33] Maurizio Palesi. 2019. Output Images Generated by JPEG Mapped on xWiNoC. Retrieved from http://utenti.dieei.unict.it/users/mpalesi/xwinoc/.

[34] Maurizio Palesi, Mario Collotta, Andrea Mineo, and Vincenzo Catania. 2015. An efficient radio access control mechanism for wireless network-on-chip architectures. *J. Low Power Electron. Applic.* 5, 2 (2015), 38–56.

[35] J. Philippe, S. Pillement, and O. Sentieys. 2006. Area efficient temporal coding schemes for reducing crosstalk effects. In *Proceedings of the 7th International Symposium on Quality Electronic Design (ISQED'06)*. IEEE.

[36] Jacob Postman, Tushar Krishna, Christopher Edmonds, Li-Shiuan Peh, and Patrick Chiang. 2013. SWIFT: A low-power network-on-chip implementing the token flow control router architecture with swing-reduced interconnects. *IEEE Trans. Very Large Scale Integ. (VLSI) Syst.* 21, 8 (Aug. 2013), 1432–1446.

[37] Samira Pouyanfar, Saad Sadiq, Yilin Yan, Haiman Tian, Yudong Tao, Maria Presa Reyes, Mei-Ling Shyu, Shu-Ching Chen, and S. S. Iyengar. 2018. A survey on deep learning: Algorithms, techniques, and applications. *ACM Comput. Surv.* 51, 5 (Sept. 2018), 92:1–92:36.

[38] Venkata Yaswanth Raparti and Sudeep Pasricha. 2018. DAPPER: Data aware approximate NoC for GPGPU architectures. In *Proceedings of the 12th IEEE/ACM International Symposium on Networks-on-Chip (NOCS'18)*. IEEE Press, Piscataway, NJ, Article 7, 8 pages. Retrieved from http://dl.acm.org/citation.cfm?id=3306619.3306626.

[39] Gennaro Severino Rodrigues, Ádria Barros de Oliveira, Fernanda Lima Kastensmidt, Vincent Pouget, and Alberto Bosio. 2019. Assessing the reliability of successive approximate computing algorithms under fault injection. *J. Electron. Test.* 35, 3 (2019), 367–381.

[40] Mehrzad Samadi, Janghaeng Lee, D. Anoushe Jamshidi, Amir Hormati, and Scott Mahlke. 2013. SAGE: Self-tuning approximation for graphics engines. In *Proceedings of the 46th IEEE/ACM International Symposium on Microarchitecture (MICRO'13)*. ACM, New York, NY, 13–24. DOI: https://doi.org/10.1145/2540708.2540711

[41] John Shalf, Sudip Dosanjh, and John Morrison. 2011. Exascale computing technology challenges. In *Proceedings of the Conference on High Performance Computing for Computational Science (VECPAR'10)*, José M. Laginha M. Palma, Michel Daydé, Osni Marques, and João Correia Lopes (Eds.). Springer Berlin, 1–25.

[42] Stelios Sidiroglou-Douskos, Sasa Misailovic, Henry Hoffmann, and Martin Rinard. 2011. Managing performance vs. accuracy trade-offs with loop perforation. In *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering (ESEC/FSE'11)*. ACM, New York, NY, 124–134. DOI: https://doi.org/10.1145/2025113.2025133

[43] Silvaco. 2019. 45nm Open Cell Library. Retrieved from http://www.nangate.com.

[44] Srinivasa R. Sridhara and Naresh R. Shanbhag. 2005. Coding for system-on-chip networks: A unified framework. *IEEE Trans. Very Large Scale Integ. Syst.* 13, 6 (June 2005), 655–667.

[45] Ye Tian, Qian Zhang, Ting Wang, Feng Yuan, and Qiang Xu. 2015. ApproxMA: Approximate memory access for dynamic precision scaling. In *Proceedings of the 25th Great Lakes Symposium on VLSI (GLSVLSI'15)*. ACM, New York, NY, 337–342. DOI: https://doi.org/10.1145/2742060.2743759

[46] Ling Wang, Xiaohang Wang, and Yadong Wang. 2017. ABDTR: Approximation-based dynamic traffic regulation for networks-on-chip systems. In *Proceedings of the IEEE International Conference on Computer Design (ICCD'17)*. 153–160.

[47] Stefan Winkler and Praveen Mohandas. 2008. The evolution of video quality measurement: From PSNR to hybrid metrics. *IEEE Trans. Broadcast.* 54, 3 (Sept. 2008), 660–668.

[48] Siyuan Xiao, Xiaohang Wang, Maurizio Palesi, Amit Kumar Singh, and Terrence Mak. 2019. ACDC: An accuracy- and congestion-aware dynamic traffic control method for networks-on-chip. In *Proceedings of the Design, Automation Test in Europe Conference Exhibition (DATE'19)*. 630–633.

[49] Qiang Xu, Nam Sung Kim, and Todd Mythkowicz. 2016. Approximate computing: A survey. *IEEE Des. Test* 33, 1 (Feb. 2016), 8–22.

[50] Amir Yazdanbakhsh, Divya Mahajan, Hadi Esmaeilzadeh, and Pejman Lotfi-Kamran. 2017. AxBench: A multiplatform benchmark suite for approximate computing. *IEEE Des. Test* 34, 2 (Apr. 2017), 60–68.

[51] Hui Zhang, Varghese George, and Jan M. Rabaey. 2000. Low-swing on-chip signaling techniques: Effectiveness and robustness. *IEEE Trans. Very Large Scale Integ. Syst.* 8, 3 (June 2000), 264–272.

[52] Brian Zimmer, Rangharajan Venkatesan, Yakun Sophia Shao, Jason Clemons, Matthew Fojtik, Nan Jiang, Ben Keller, Alicia Klinefelter, Nathaniel Pinckney, Priyanka Raina, Stephen G. Tell, Yanqing Zhang, William J. Dally, Joel S. Emer, C. Thomas Gray, Stephen W. Keckler, and Brucek Khailany. 2019. A 0.11 pJ/Op, 0.32-128 TOPS, scalable multi-chip-module-based deep neural network accelerator with ground-reference signaling in 16nm. In *Proceedings of the Symposium on VLSI Circuits*. C300–C301.

[53] Kaiwei Zou, Ying Wang, Huawei Li, and Xiaowei Li. 2019. Learn-to-scale: Parallelizing deep learning inference on chip multiprocessor architecture. In *Proceedings of the IEEE/ACM Design, Automation and Test in Europe Conference (DATE'19)*. 1172–1177.

Received July 2019; revised January 2020; accepted January 2020