Evaluation of Switch-to-Switch Header Flit Protection Schemes in Networks-on-Chip

Martin Gag, Philipp Gorski, Tim Wegner and Dirk Timmermann
Institute of Applied Microelectronics and Computer Engineering
University of Rostock, Germany
{martin.gag, philipp.gorski, tim.wegner and dirk.timmermann}@uni-rostock.de

Abstract

Signal integrity and packet data protection against soft errors represent highly relevant challenges for Networks-on-Chip regarding the shrinking of process technology. Therefore, data protection and error recovery strategies at the End-to-End level are the most convenient solutions considering overall implementation costs and performance penalties. Nevertheless, protection of packet header control information at the data link layer is indispensable, since the header contains data highly relevant for correct routing. Corrupted data by the presence of soft errors may lead to misdirected packets affecting latency, data misinterpretation due to corrupted encoding flags or loss of the packet. Thus, additional Switch-to-Switch header protection is inevitable.

By using error correcting and detecting codes the number of additional components of a Network-on-Chip router can grow inappropriately. Accordingly we considered different techniques like resource sharing and minimizing the number of decoders. In this work we evaluate several concepts by means of simulations and syntheses utilizing a 45 nm process technology. Our results show that header protection can be applied with very low performance degradation under minimal additional hardware costs by gaining a very reliable header flit.

1 Introduction

The ongoing process technology scaling down to sizing dimensions of 45 nm, 32 nm and less enables the integration of billions of transistors on a single chip. Thus, it becomes reality for modern IC designs and embedded systems to integrate up to hundred and more functional and storage resources in a single System-on-Chip (SoC) [1]. Along with the growing complexity of these SoC solutions the existing communication infrastructures and global interconnection architectures do not scale accordingly regarding the demands of increasing on-chip data traffic, communication performance, reusability issues and energy consumption. On-chip communication becomes a major bottleneck to system design and recently the Network-on-Chip (NoC) paradigm, a packet-based on-chip communication network, seems to be the most promising solution to face these complexity problems [2]–[6]. But shrinking technology scales also makes integrated circuits more sensitive and vulnerable to internal and external noise (i. e. crosstalk, coupling noise, electromagnetic interference or radiation), Single Event Upsets (SEUs), Single Event Transients (SETs) or process variability (i. e. voltage drops, delay and frequency variations) in fabrication [1]. Downsizing the dimensions of a transistor is accompanied by higher uncertainties in the lithography process, a reduction of the power supply voltage, an increased vulnerability of the circuit structures to defects regarding electrical overstress and smaller signal-to-noise ratios. Thus, signal integrity and protection of transmitted data against soft errors become the next major challenges to establish the NoC paradigm as a dependable and error resilient design solution for SoCs [1], [7].

In this work, we focus on the protection of header data of on-chip packets, containing relevant information for transmission control and routing, to ensure that each packet will be routed to its correct destination in presence of possible soft errors. Therefore, different concepts based on Error Detection Codes (EDCs) and Error Correction Codes (ECCs) are analyzed, implemented and evaluated regarding their impact on hardware complexity, silicon area and performance. The basic idea of the presented investigations relies on the separation of data protection concerns. The NoC as communication infrastructure has to avoid that data packets will be misdirected or lost. Protection mechanisms facing this challenge have to be implemented on the Switch-to-Switch (S2S) level because each intermediate transmission node between source and destination of the packet performs routing decisions based on the information in the header data. The protection of payload data is performed by the endpoints and is

highly application dependent. Thus, payload protection corresponds to a problem that has to be tackled on the End-to-End (E2E) level in a manner that applications can configure their protection level as it is needed to trade off data throughput against reliability. Furthermore, E2E protection of payload data is a hardware/software design problem shared between the NoC hardware (i. e. network interface) and the computational resources. Those mechanisms for E2E data protection are beyond the scope of the evaluation presented in this work, but will be highly relevant for future investigations to develop a fully suited error protection and recovery strategy.

The remainder of this paper is organized as follows: Section 2 summarizes the related work and highlights the major contribution. Section 3 introduces the evaluated header protection concepts and contains preliminary analyses. Section 4 presents the results of implementation and synthesis using a 45 nm process technology. In section 5 results of functional simulations will be given and put in relation to the implementation costs. Finally, conclusions and the outlook are presented in Section 6.

2 Related Work

The protection of communication data in NoCs is a highly relevant issue and over the last years many different approaches were presented and compared with each other. Most of the published studies do not differentiate between header and payload of a NoC packet. Consequently, the employed mechanisms for error protection and recovery are applied to each flit or the complete packet independent of its contents.

Frantz *et al.* [8] evaluate mechanisms for flit based data protection on the S2S level. They apply different error correction strategies, based on single-error correction and double-error detection Hamming Codes (SEC-DED HC) or timing error recovery by delayed multiple data sampling, to protect each flit of the packet against soft errors originating from links and in buffers. In [9] Frantz *et al.* extend this evaluation by an E2E-based hardware/software solution, where links are protected by hardware against timing errors and the complete packet is protected with an attached cyclic redundancy checksum (CRC). The CRC is generated and checked at the endpoint resources by a mechanism implemented in software. In the event of packet corruption due to soft errors E2E retransmission is initiated. Misdirected packets are covered by retransmission initiated through a timeout mechanism. The evaluation results show that this hardware/software E2E approach clearly outperforms pure S2S hardware solutions regarding implementation costs, performance penalty and energy overhead.

Dutta *et al.* [10] introduce a S2S packet based protection scheme for store-and-forward routing against soft errors in links and buffers. Therefore, each component (link and buffer) is equipped with an encoder/decoder pair. The encoding is implemented with an adapted unequal error protection code with improved error correction capabilities.

Yu *et al.* [11] present a heavy weighted adaptive S2S flit protection scheme facing soft errors originating from link interconnects. The solution offers multiple adaptive transmission modes for error protection using three different HCs (from SEC-DED up to 4-bit-error correction and 8-bit-error detection) and variable block sizes for encoding and interleaving.

In [12] different crosstalk avoidance codes (CAC) for S2S flit based protection mechanisms are evaluated regarding error protection and energy reduction. The presented codes are based on the additional insertion of parity bits or dual rail coding strategies. Unfortunately, these codes require at least an increase of the available link width by the factor of 2 compared to the uncoded data transmission. This work is extended in [13] with the capability for double-error correction using a combination of CAC and HC. A similar solution is presented in [14], but instead of HCs this solution applies bit triplication in combination with a three-input majority voter and the CAC is implemented with an own table-driven approach. Sridhara *et al.* [15], [16] evaluate the combination of different flit based coding techniques on S2S level for low power, crosstalk avoidance and error correction, regarding a trade-off for implementation costs, performance and reliability.

In [17] Rossi *et al.* present a configurable flit based error protection mechanism with three different protection modes and controllable S2S or E2E policies to implement different Quality-of-Service (QoS) levels. Applied correction mechanisms are a SEC Hamming code, a SEC-DED Hsiao code and a symbol error correction code.

Zimmer and Jantsch [18] evaluate an encoding scheme, which separates the header from the payload flits to propose a quality of service implementation. While the header information is secured by a SEC-DED approach, the code of the payload can be selected out of four implemented alternatives (no, SEC, DED, SEC-DED). However, the work concentrates on the selection of codes and a fault model, this process is based on. It lacks evaluation of implementation possibilities and costs in terms of area, power and delay.

The evaluation results of [9], [19], [20] indicate that E2E error protection mechanisms outperform S2S strategies regarding hardware overhead, energy consumption and latency. Especially, the holistic analysis in [21] and [22] for different error recovery strategies consolidates these statements. Unfortunately, the complete abandonment of S2S level error protection is not an option, since error protection on S2S level is indispensable to secure that control information for packet transmission is not corrupted in presence of soft errors. This control information is normally

included in the header of each packet and must be protected at all costs to guarantee a reliable packet delivery. Furthermore, the header may contain data that is relevant for the reliable operability of the intermediary switch nodes (i. e. the packet length to control if the number of routed flits matches this value). Single flit packets used for the implementation of request/acknowledgment flow control and event signaling scenarios need to be protected as well. Additionally, misdirected packets can decrease overall system performance in a unpredictable manner depending on current traffic utilization.

Most of the aforementioned works are handling error protection as flit or packet based solutions on S2S level, but do not offer a more differentiated error protection where only selected flits of a packet (e. g. header flits) are protected. Moreover, resource sharing in terms of reducing the number of encoders and/or decoders per router is not considered. On the one hand, a more selective protection mechanism would lead to higher energy efficiency and performance, since only flits relevant for reliable packet delivery are examined. On the other hand, reduction of encoding and decoding units to a minimum seems to be a promising opportunity to reduce hardware overhead and power consumption. Of course, it has to be assured that all relevant flits still can be encoded and decoded by the remaining units. Thus, in this paper we examine to which degree selective error protection of flits and resource sharing for encoders and decoders on the S2S level is applicable. Furthermore, analyses and evaluations regarding implementation costs, latency and correction capabilities will serve to identify the best trade-off between reliability, performance and hardware overhead.

3 Header Protection Concepts & Analysis

The proposal of encoding only the header flits and feeding through the body flits (see **Fig. 1**) achieves a separation of communication layers. While the link level reliability is assured at this level, the payload can be secured by an E2E solution of a higher level. This provides a higher degree of freedom to the E2E error protection, because the decision between simple coding with high throughput and advanced error control with high reliability is left to the endpoints. As we are applying different encodings to header and body, we are proposing to secure the flit identifier (header or body) by lower level approaches like physical design constraints or redundancy (i. e. multiple links with majority voter).

The implementation of header protection mechanisms on the S2S level has to fulfill different constraints regarding the implementation costs and performance penalties. Since it will be combined with an E2E protection for application payload the resulting area overhead has to be minimal to establish a complete solution that will need fewer resources

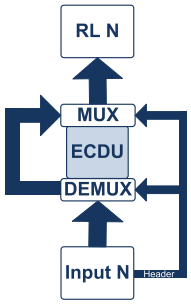


Fig. 1. Scheme of decoding in each switch: Header flits are going from input through the ECDU to the routing logic. Body flits are bypassing the ECDU.

than the fully suited S2S approach. Furthermore, the introduced additional hardware components must limit the performance degradation to a minimum compared to the unprotected case. Moreover, any performance decrease will affect a NoC packet at each intermediary node between source and destination. This may reduce the performance advantage of E2E over S2S solutions. In any case the concept must be easy to integrate without increasing the hardware complexity or being dependent of any specific technological aspect. The following subsections will introduce the selected solution strategies.

3.1 Selected Encodings for Error Protection or Correction

Data encoding represents the best alternative to implement a protection mechanism with high independence of any process technology and is best suited for the implementation of combined hardware/software solutions. We have selected different standard codes that provide enough detection and/or correction capabilities to be suitable for the considered amounts of data. The packet header contains different types of control information that are relevant at the S2S (i. e. packet length, source and destination address) or at the E2E level (i. e. source task identifier, payload encoding information). The header information needed by the switch depends on the routing algorithm. Assuming simple XY-Wormhole Switching [23], the destination address is the control information contained in the header on which the routing decision is based. The address width depends on the number of IP cores in the mesh. We assume a size of up to 16x16 processing elements which results in 8 address bits. Assumptions about the other necessary header information extremely vary with the methods used (i. e. flow control, E2E error coding). Consequently, four different flit widths were considered. The gross flit width was chosen to match popular word widths like 32, 64, 128 and 256 bits for compatibility reasons. Accordingly, this results in lower net flit widths for header data (see **Table 1**).

We selected different codes to enable scenarios for double error detection with retransmission (DED),

TABLE 1
Characteristics of selected encodings and resulting net and gross flit widths

Encoding	Detection/Correction	Net	Gross
HC(3,1)	SEC or DED in 3	11, 22, 43, 86	33, 66, 129, 258
HC(7,4)	SEC or DED in 7	20, 40, 76, 148	35, 70, 133, 259
HC(12,8)	SEC or DED in 12	24, 48, 88, 176	36, 72, 132, 264
EHC(4,1)	SEC-DED in 4	8, 16, 32, 64	32, 64, 128, 256
EHC(8,4)	SEC-DED in 8	16, 32, 64, 128	32, 64, 128, 256
EHC(13,8)	SEC-DED in 13	24, 40, 80, 160	39, 65, 130, 260

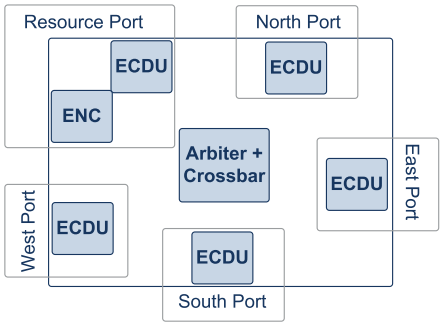


Fig. 2. Each input port and the resource output port is using one decoder (ECDU)

single-error correction (SEC) and single-error correction with double-error detection with retransmission (SEC-DED). These are promising lightweight implementations of protection circuits and simple codes. The use of various block widths allows for different complex encodings by dividing the data into differently sized chunks. The SEC scenarios were implemented with three different HCs. HC(7,4) and HC(12,8) represent the candidates for four and eight bit chunk encoding. The HC(3,1) is a special case and equal to a triple modular redundancy encoding with majority voting. As an alternative the HCs can be used to detect one and two bit errors instead of correcting the one bit errors by using a slightly changed decoder. Additionally, to implement SEC-DED extended HCs (EHC) were applied. Therefore, HC was extended with a parity bit generated over the data and check bits to enable the DED. The selected codes in combination with the block encoding allow to trade-off reliability versus overhead for the necessary check bits. Smaller chunks paired with stronger encodings provide better detection and correction capabilities, but result in a higher amount of additional check bits.

3.2 Area Reduction Approaches

The method of data encoding implies the information bits are changed or some parity bits are added at the encoder and this process is reversed at the decoder. The straight forward approach to place this processing elements in a NoC switch is to let each input port be followed by a decoder and put an encoder at each output port resulting in coding units in the doubled number of ports. This serves to protect the links and the (input) buffers against soft errors. However, we need to minimize the number of encoding and decoding elements to reduce hardware overhead. In case of HCs the encoder adds a number of parity bits to the original data and the decoder as an error correction and/or detection unit (ECDU) corrects respectively detects errors in the payload data by using the parity information. If the data should be forwarded in an encoded way, i. e. with the parity bits present, there is no need for a further encoder. Thus, we are placing one decoder unit after each input port and no encoders at the output ports, but are feeding the encoded data through the switch (see **Fig. 2**). With this method the number of needed coding units is reduced to the number of ports.

Still the data needs to be encoded somewhere. The resource port connecting the IP core is a special case in the switch. It receives uncoded data from the resource and has to encode it. The outgoing resource port needs to decode the data for the connected IP core (i. e. extracting the payload from the parity bits). Thus, the coding concept is transparent for the IP cores of the NoC, which means the higher layers of the communication system do not need to know anything about the header protection.

Another approach to reduce the number of components in a router is to leave the decoding at the resource port to the ECDUs of the other input ports. The decoded signal is transported through the crossbar and arbiter of the switch and put out directly to the resource output. This approach saves another decoding unit, the ECDU of the resource port, introducing eventually more wiring overhead. The drawback of this approach is due to the fact that the header is unprotected for the time it is passing the routing logic and crossbar of the last switch. This might be considered if most of the errors are occurring at the links or input buffers.

A further promising concept is to share the decoders of the input ports. This approach is examined in detail in the next section.

3.3 Shared Resource Concept

Reducing the protection at the S2S level to the header data implies that less portions of each packet will be checked and the protection unit is more inactive over an increased time interval (depending on the packet length). Thus, it might be possible to implement the

TABLE 2
Simulation configuration for analysis of the simultaneous occurrence of one or more headers in a router

NoC parameter	Configuration
Topology	2D mesh
Dimensions	10x10
Switching	Wormhole
Routing	XY
Traffic pattern	Randomly distributed
Buffer size	1
Packet size	1 up to 5 flits
Injection rate	0.05 up to 0.25 flits/clock cycle

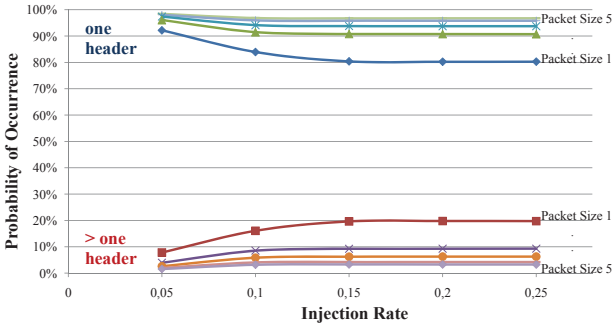


Fig. 3. Relative frequency of one resp. two or more incoming header flits at the same time at the same switch

protection mechanisms as shared resources between different ports.

Therefore, we analyzed the feasibility of a shared resource concept with multiple simulation runs of a NoC (based on a cycle-accurate SystemC-TLM environment configured as given in **Table 2**). In each simulation the number of incoming header flits per clock cycle for each of the 100 NoC switches was logged. By independently varying the parameters of packet size and injection rate for every simulation run we were able to acquire the overall behavior of the observed phenomenon. Afterwards, we summarized the cases where only one respectively two or more header flits simultaneously arrived at the same switch node and divided this by the sum of all occurrences of an arriving header flit at a switch in the simulated NoC. The resulting value represents the relative frequency for those situations.

Figure 3 shows these simulation results. It is obvious that at least in 80 % of all cases only one header flit arrives at the same switch in one clock cycle for a simulated packet size of one (each packet is a header flit). Increasing the packet size to two raises the relative frequency up to 90 % and for even higher values it reaches more than 93 %. Respectivly, the number of multiple arriving header flits at the same time at one switch is under 20 % in worst case and under 10 % in case of a packet length of two or more flits. This analysis encouraged us to implement

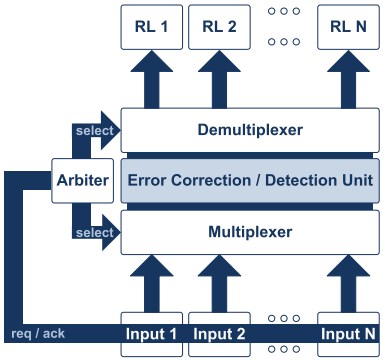


Fig. 4. Shared resource concept, multiple ports are using the same ECDU

and evaluate the introduced shared resource header protection concept.

Figure 4 depicts a schematic overview of the integration of this protection solution into the NoC switches. Each input port (Input 1 to Input N) will be connected to a multiplexer sourcing the ECDU. This coding unit is connected to a demultiplexer that sources the routing logic (RL 1 to RL N) of the connected ports. The demultiplexer and the multiplexer will be controlled by the outputs of an arbiter. This arbiter works with a round-robin scheduling and handles the resource requests from the inputs. If a header flit arrives at an input it has to pass through the ECDU. Thus, the input component sends a request to the arbiter, disables the reception of new incoming flits and waits until the access will be acknowledged. Afterwards, the relevant bits of the header flit will be passed from the input to the ECDU. The ECDU itself is connected to the handshake logic of each coupled input to initiate necessary retransmissions or to block new incoming flit transmissions until the correction has finished.

This shared resource concept will be implemented for two different cases. For the NoC switch we assume that the port, directly connected to the resource, does not need any header protection (it will be in scope of the E2E protection) and only those ports connected to other switches (North, East, West and South) will be integrated with an ECDU coupling. Thus, case one includes that these four ports share one ECDU resource (4:1). In the second scenario only two of four ports will share one ECDU and each switch includes two protection units (2:1). The shared resource approach itself leads to possible additional delay of maximal one (2:1) or three (4:1) clock cycles until the resource can be accessed, because in worst case all other connected requesters come first accessing the ECDU. Of course, occurred errors in case of error detection and retransmission scenarios can introduce additional wait cycles. But the aforementioned simulation results showed that the average latency penalty will be less even in case of high traffic situations.

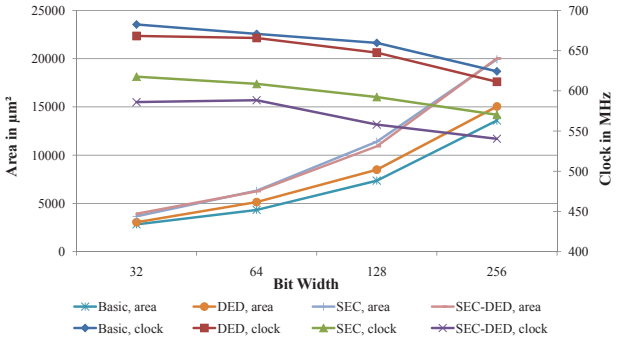
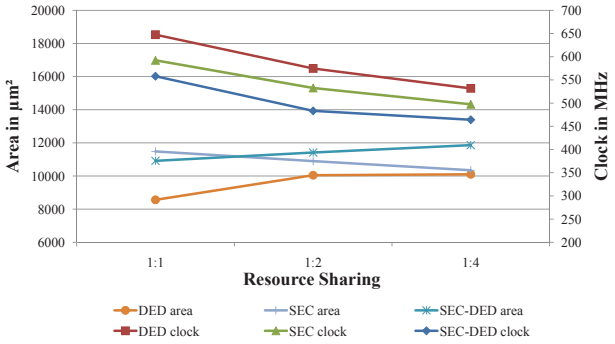
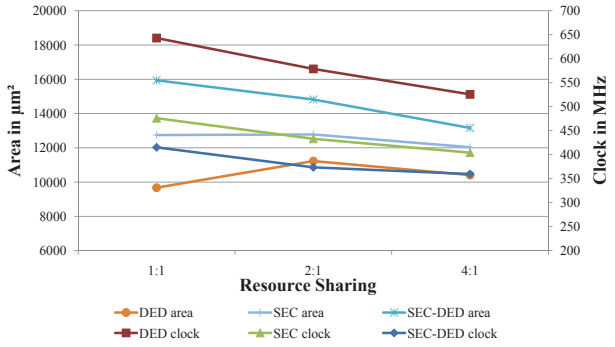


Fig. 5. Synthesis results for area and clock rate of different encodings (no, HC(3,1), EHC(4,1)) under the use of different flit widths



(a) HC(3,1), EHC(4,1)



(b) HC(12,8), EHC(13,8)

Fig. 6. Synthesis results for area and clock rate of different resource sharing schemes of the ECDU and different encodings at 128 bit

4 Implementation

The presented solution concepts were integrated into a reference architecture based on VHDL and afterwards synthesized with a comparable standard design flow. For synthesis we used the Synopsys Design Compiler (SDC) tool and to create comparable results, regarding the process technology, the 45 nm Nangate FreePDK45 Generic Open Cell Library was applied. The synthesis process of the SDC was configured to a medium effort level for all relevant parameters. Thus, no SDC specific optimization strategy influenced the produced results. As reference architecture we utilized a 2D-Mesh NoC switch implementing XY-routing, round-robin port arbitration, a two-phase handshake protocol for flit transmission and wormhole switching. The input buffer size for each port was set to one and the link width of the NoC was specified to be 32, 64, 128 or 256 bit. Because of the possibility to divide the flits into chunks and encode these independently, the link widths do not exactly match these values in every case (see **Table 1**).

Figure 5 depicts the results of a synthesized router using different flit widths and encodings, as well as no encoding. The correlation of growing area demands and increasing flit width is obvious. Simultaneously, the clock frequency is decreasing. The reference implementation without any data encoding is the smallest and fastest design, while DED is more complex, followed by SEC, which suffers from a more complex decoder due to the correction capability. SEC-DED is the most complex architecture, processing an additional parity bit. DED is approximately 10 MHz slower, which corresponds to 1.5 %, while SEC and SEC-DED introduce more relevant delays (60-100 MHz or 9-15%). The area overhead of SEC and SEC-DED is approximately 30 %, whereas DED is not much bigger than no encoding (6-10 %).

Furthermore, a comparison of the different protection concepts (no sharing, share 2:1 and share 4:1) states that an implementation of protection mechanisms as shared resource components will not benefit in additional silicon area saving in every case (see **Fig. 6a**). The silicon efforts seem to be dominated by the management components (arbiter and de/multiplexer) and the more complex wiring. Only the SEC encoding shows an area improvement (appr. 5 %) with 2:1 and 4:1 resource sharing. The performance in terms of possible clock rates decreases about 20 up to 70 MHz with the number of ports that share one resource, because of the additional management components. The area benefit increases if the encoding gets more complex, i. e. the chunk size of the protected header increases to higher ranges like 4 or 8 bits. In the case of 8 bit wide chunks SEC and SEC-DED produce 8 % respectively 17 % smaller designs if the resources are shared 4:1 (see **Fig. 6b**). The reachable clock rate is decreasing by 15 % and 13 %.

Hence for this synthesized configuration a shared resource solution is up to 17 % smaller than the straightforward implementation of data encoding with one ECDU per input port. The downsides of this approach are the introduced wiring effort and additional multiplexers, causing longer critical paths (up to 15 %). Thus, the 1:1 integration without any resource sharing could be superior considering a timing critical NoC design.

The approach of saving one additional unit by using the input port decoders on the output to the resource performs about 30 MHz faster. In terms of silicon the design uses slightly less area and consumes 3 up to 12 % less power, which depends on

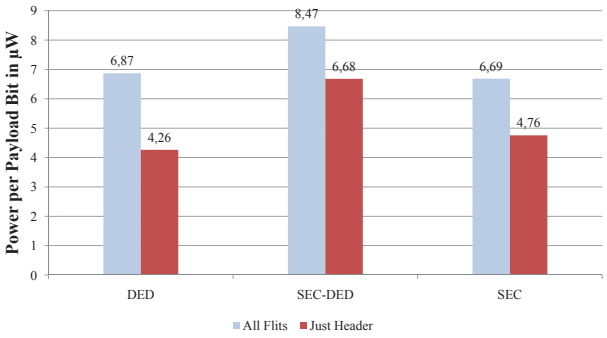


Fig. 7. Power per payload bit for encoding all flits vs. encoding just the header flits, 128 bit flit width, packet size: 2

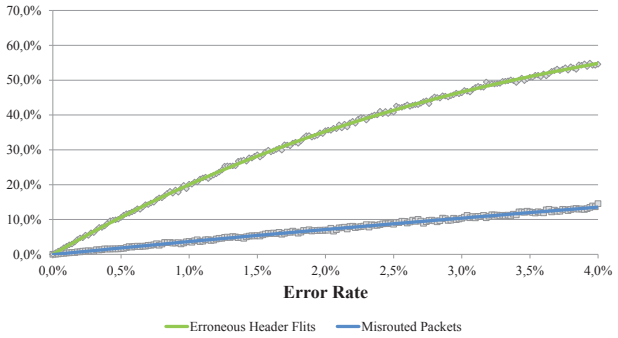
TABLE 3
Complexity of encodings by means of synthesis results, 128 bit flit width

Encoding	Clock in MHz	Area in µm^2	Power in mW
HC(3,1)	592	11484	4.1
HC(7,4)	499	12223	4.6
HC(12,8)	476	12745	4.5
EHC(4,1)	558	10915	3.9
EHC(8,4)	461	14665	5.4
EHC(13,8)	415	15937	5.8

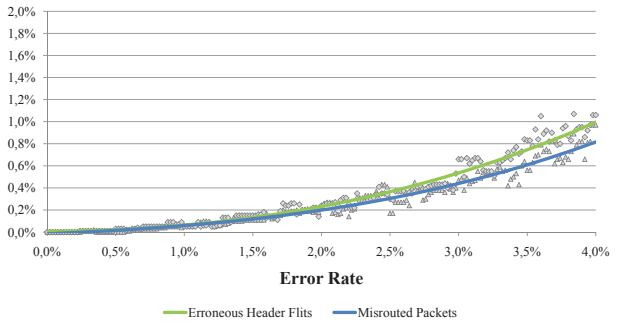
the resource sharing and encoding configuration.

To review the concept of header protection instead of generic flit protection, we compared the power per payload bit for different encodings (see **Fig. 7**). It can be stated that we are able to save 1,7-2,6 µW which corresponds to 20-40 %. Because our proposed concept needs an E2E error protection the arising overhead would have to be added to the reduced power values.

As **Table 3** shows, the complexity of the encoders is growing with increasing chunk size. Area and power values are slightly higher, but there is a significant performance degradation by up to 140 MHz.



(a) Without Encoding



(b) With EHC(4,1)

Fig. 8. Rate of erroneous flits and misrouted packets, 32 bit flit width

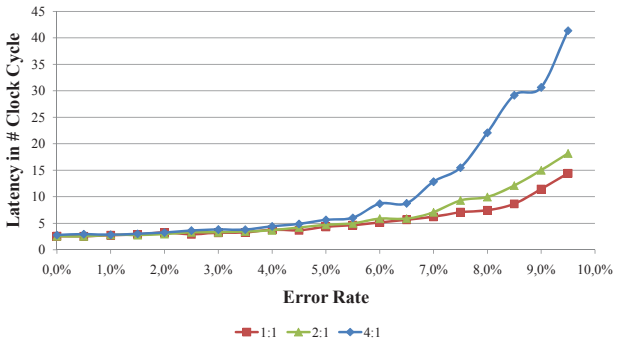


Fig. 9. Average packet latency of a switch at 15 % injection rate, 128 bit flit width

5 Simulation Results

To evaluate the encoding concept we ran functional simulations of the router designs. The header protection was challenged with an error injection model inverting bits of the data flit before it arrives at the ECDU. In this way bit faults caused by soft errors like SEU, SET or noise could be introduced. An error rate (f) for all the data flits was set and with a chance of f each bit of the flit could change its value. Thus, in case of a flit width of 32 bits the probability of getting an erroneous flit is $1-(1-f)^{32}$. If this error occurs in the header flit, the packet has a chance of not being routed correctly due to erroneous control information (see **Fig. 8a**). However, the performance of the SEC-DED code is shown in **figure 8b**, where the error rate in header flits is drastically cut. At an error rate of 2 % the faulty header flit rate is 170 times lower than without encoding. In consequence the number of misrouted packets decreases by far.

The error rate is influencing the latency of the packets in cases of error detecting codes like DED and SEC-DED. **Figure 9** shows this behavior. Resource sharing makes it even worse, because the ECDU is blocked by one erroneous header waiting for correct retransmission blocking the other ports. The preceding simulations promised that in most cases just one header is arriving at the router at the same time. The functional simulation reveals that the additional wait cycles for retransmission can pile up the packets at the buffer and seriously increase the latency if the error rate increases above 2 %. As expected the latency of 4:1 and 2:1 is higher than of 1:1. However, it must be stated, that the latency increase of the non shared router is nearly as high as

the 2:1 shared router and that an error rate of 2 % is surely high.

It can be stated, that under the used codes EHC with SEC and DED provides the best protection against errors. However, in combination with the smallest chunk size of one bit the resulting encoding needs four data bits for a payload of one. This is reducing the effective flit width which can be used by the header by three quarter. The EHC with increased chunk width of 4 or 8 bits still provides good error protection but the performance of the router decreases due to the growing logic depth of the decoders.

6 Conclusion

The presented work shows that S2S header flit protection is an inevitable addition to E2E error recovery for NoCs, which can be applied for different coding scenarios and amounts of protected data with a minimum of implementation costs. Our proposal of encoding only the header flits and preserving the payload for E2E error protection performs well. Furthermore, the introduced solutions only slightly affect the overall system performance, because light weight error codes were chosen. Different concepts were introduced to minimize the hardware overhead, implemented with a state-of-the art design flow and evaluated regarding the synthesis results for a 45 nm technology process. As a result for the evaluated synthesis configuration the proposed concept of shared decoding units is the superior solution in terms of area (up to 17 %), while dedicated resource implementations have shorter critical paths (up to 12 %). It showed, that as long as small and simple coding hardware is used, shared resources are not outperforming straight forward implementations. The more complex the coding hardware gets, the more the design benefits from shared resources.

For future work the exploration of the best fitting E2E recovery and flow control is planned to create a full suited error protection solution. Furthermore, we will focus on the compatibility of error protection with improved capabilities for online testing.

References

[1] The International Technology Roadmap for Semiconductors, "2009 edition," ITRS, Tech. Rep., 2009. [Online]. Available: http://www.itrs.net

[2] T. Bjerregaard and S. Mahadevan, "A survey of research and practices of network-on-chip," *ACM Comput. Surv.*, vol. 38, June 2006.

[3] F. Gebali, H. Elmiligi, and M. W. El-kharashi, *Networks-On-Chips: Theory and Practice*. Boca Raton, Florida: CRC Press/ Taylor and Francis Group LLC, 2008.

[4] A. Hemani, A. Jantsch, S. Kumar, A. Postula, J. Öberg, M. Millberg, and D. Lindqvist, "Network on chip: An architecture for billion transistor era," in *Proc. of the IEEE NorChip Conf.*, 2000.

[5] A. Ivanov and G. D. Micheli, "The Network-on-Chip Paradigm in Practice and Research," *Design & Test of Computers*, vol. 22, 2005.

[6] U. Y. Ogras, J. Hu, and R. Marculescu, "Key research problems in NoC design: A holistic perspective," in *Proc. of the Intl. Conf. on Hardware/Software Codesign and System Synthesis*, 2005.

[7] R. Marculescu, U. Y. Ogras, L.-S. Peh, N. E. Jerger, and Y. Hoskote, "Outstanding research problems in NoC design: System, microarchitecture, and circuit perspectives," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, 2009.

[8] A. P. Frantz, F. L. Kastensmidt, L. Carro, and E. Cota, "Dependable network-on-chip router able to simultaneously tolerate soft errors and crosstalk," in *IEEE Proc. of Int'l Test Conf. (ITC 06)*, 2006.

[9] A. P. Frantz, M. Cassel, F. L. Kastensmidt, E. Cota, and L. Carro, "Crosstalk- and seu-aware networks on chips," *IEEE Des. Test*, vol. 24, 2007.

[10] A. Dutta and N. A. Touba, "Reliable network-on-chip using a low cost unequal error protection code," *Defect and Fault-Tolerance in VLSI Systems, IEEE International Symposium on*, vol. 0, pp. 3–11, 2007.

[11] Q. Yu and P. Ampadu, "Adaptive error control for nanometer scale network-on-chip links," *IET Computers Digital Techniques*, vol. 3, 2009.

[12] P. P. Pande, A. Ganguly, B. Feero, B. Belzer, and C. Grecu, "Design of low power & reliable networks on chip through joint crosstalk avoidance and forward error correction coding," in *Proc. of the 21st IEEE International Symposium on on Defect and Fault-Tolerance in VLSI Systems*, 2006.

[13] A. Ganguly, P. P. Pande, B. Belzer, and C. Grecu, "Design of low power & reliable networks on chip through joint crosstalk avoidance and multiple error correction coding," *J. Electron. Test.*, vol. 24, 2008.

[14] P.-T. Huang, W.-L. Fang, Y.-L. Wang, and W. Hwang, "Low power and reliable interconnection with self-corrected green coding scheme for network-on-chip," in *Proc. of the Second ACM/IEEE International Symposium on Networks-on-Chip*, ser. NOCS '08, 2008.

[15] S. R. Sridhara and N. R. Shanbhag, "Coding for system-on-chip networks: a unified framework," in *Proc. of the 41st annual Design Automation Conference*, 2004.

[16] ——, "Coding for reliable on-chip buses: A class of fundamental bounds and practical codes," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 26, 2007.

[17] D. Rossi, P. Angelini, and C. Metra, "Configurable error control scheme for noc signal integrity," in *Proc. of the 13th IEEE International On-Line Testing Symposium*, 2007.

[18] H. Zimmer and A. Jantsch, "A fault model notation and error-control scheme for switch-to-switch buses in a network-on-chip," in *Proceedings of the 1st IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, ser. CODES+ISSS '03. New York, NY, USA: ACM, 2003, pp. 188–193. [Online]. Available: http://doi.acm.org/10.1145/944645.944694

[19] A. Jantsch, R. Lauter, and A. Vitkowski, "Power analysis of link level and end-to-end data protection in networks on chip," in *2005 IEEE International Symposium on Circuits and System*, 2005.

[20] D. Bertozzi, L. Benini, S. Member, and G. D. Micheli, "Error control schemes for on-chip communication links: the energy-reliability tradeoffs," *IEEE TCAD*, 2005.

[21] S. Murali, *Designing Reliable and Efficient Networks on Chips*. Springer Publishing Company, Incorporated, 2009.

[22] S. Murali, T. Theocharides, N. Vijaykrishnan, M. J. Irwin, L. Benini, and G. D. Micheli, "Analysis of error recovery schemes for networks on chips," *IEEE Des. Test*, vol. 22, 2005.

[23] L. M. Ni and P. K. McKinley, "A survey of wormhole routing techniques in direct networks," *Computer*, vol. 26, 1993.