Chapter 6

ERROR-TOLERANT INTERCONNECT SCHEMES

Heiko Zimmer
Institute of Microelectronic Systems
Darmstadt University of Technology, Darmstadt, Germany
heiko@mes.tu-darmstadt.de

Axel Jantsch
Institute of Microelectronics and Information Technology
Royal Institute of Technology (KTH), Stockholm, Sweden
axel@imit.kth.se

1. Introduction

The Network-on-Chip paradigm targets future Systems-on-Chip built of a large number of reusable, independent Intellectual-Property-blocks (IPs). A typical approach is to align these IPs as tiles in a regular manner, each associated with a wrapper providing access to the on-chip network. The network itself is a regular structure composed of switches/routers and the interconnecting links.

The objective of implementing a Network-on-Chip is to decouple computation from communication by offering a uniform, reliable, and versatile communication platform for all the inter-IP communication required by a typical SoC application. Thus, the need for custom wiring to build an application-specific communication infrastructure is overcome. Furthermore, placement and routing are simplified for the whole NoC because both the IPs and the network components are encapsulated from one another except for a defined network interface providing network access in terms of services usable by the IP for all communication it requires with its surroundings.

To fully exploit the advantages this approach offers, the network must provide defined, reliable communication services to the resources attached to it. This problem is far from trivial since the network links will most likely not be error-free in future deep submicron technology generations.

While much work is done to develop robust transmission schemes, it is expected that especially crosstalk will seriously affect interconnect reliability. When the physical layer of an on-chip network fails despite all preventive measures taken, the effect should be controlled. This is what the error-tolerant interconnect schemes provide: They increase the network reliability and hide imperfections from the applications which use the communication services offered.

Reliable network services are of great importance since applications have demands on communication that must be fulfilled to achieve correct application behaviour. In a classical approach, the communication infrastructure of a SoC is a combination of shared buses and custom designed interconnect to meet specific requirements. However, when all communication should be transported over a common medium, the on-chip network, it must be taken into account that the demands are different for different applications and may include bandwidth guarantees, integrity requirements or deadlines for completion of a specific task in real-time applications.

Typically, an application running on a SoC is comprised of multiple processes associated to different NoC-resources. Naturally, the characteristics of data transport within one application are not uniform since different traffic types such as control messages, audio signals and video streams have to coexist. Even seemingly regular data streams become irregular during processing. For instance video streams are usually encoded such that only the delta between frames are transmitted, which makes the communicated data volume higly dependent on the video content.

Closing the gap between the hardware platform's possibilities and the applications' requirements is the demanding task of error-tolerant interconnect schemes. Their aim is to provide a network with defined properties to the application. In the ideal case, what the applications see is an error-free communication medium fulfilling all their communication needs.

Furthermore, this idea hides the physical implementation details of a specific technology. By providing defined services, the border between platform design (technology, layout, error-tolerant interconnect scheme) and application design (using communication at defined QoS-levels) is clear.

Which measures are taken to implement an error-tolerant interconnect scheme depends on the specific applications' requirements and the constraints imposed by the selected platform/architecture. Therefore, we give a general overview before focusing on one specific example.

2. Architectural Considerations

One main objective of networks on chip is the idea that they are formed by a regular structure. This helps to overcome the problems associated with custom

and global interconnects: Designing and verifying specialized custom interconnects takes considerable amounts of time, global interconnects traversing the whole chip will be inefficient from an energy viewpoint and very susceptible to crosstalk-related errors. By building a regular interconnection architecture that is defined by the platform, the electrical parameters of the interconnects can be tightly controlled. For instance, in a regular mesh all links have the same length. Obviously, much effort can be put into fine-tuning the electrical parameters of these links (in order to optimize reliability, energy consumption etc.) and an efficient implementation will be very rewarding. This example shows that the NoC-approach enables designers to fully take advantage of a given technology to implement the low-level details of transmission in a place- and energy-efficient manner. At the same time, the technological details influencing interconnect design are hidden from the application.

In the past, various NoC-architectures have been proposed, among them tori, hypercubes, meshes and trees. In defining the architecture, many characteristics are decided that have an impact on the possibilities to realize fault-tolerant (i.e., reliable) communication.

2.1 Architecture-imposed Constraints

Among the most important parameters for data protection schemes are (a) the width of the links between resources and switches, (b) if parallel or serial transmission is employed on these links, and (c) how much delay, area and power overhead can be tolerated in the switches.

Considering for instance a $n \times m$ mesh it is obvious that an area-efficient switch design will be very rewarding: Since a total of $n \cdot m$ switches form the NoC, even small savings can have a significant effect on the overall area overhead introduced by the on-chip network. This clearly motivates carefully balanced design decisions when implementing network components, especially those dealing with error-tolerance for they can be arbitrarily complex.

2.2 Errors in DSM

Compared to today's circuits, the number of transient errors is expected to increase significantly in future technology generations [1]. The reliability of long interconnects, especially buses, will suffer from crosstalk faults. At the same time, technology scaling gives rise to new error sources. Among them is radiation which has previously only been a concern with memories, but will affect logic and low capacitance/low power buses as well. Generally speaking, the amount of transient faults increases with the decrease of feature size.

Crosstalk. The main source of errors on buses will be crosstalk, which is strongly influenced by the coupling capacitance between wires and can lead to

increased delay or cause voltage glitches on wires [2]. Furthermore, crosstalk can inherently cause multi-bit and bidirectional errors [3] (i.e., disturbing multiple wires, causing both $0 \rightarrow 1$ and $1 \rightarrow 0$ errors).

Crosstalk depends not only on the physical and electrical parameters but also on the dynamic switching behaviour. Thus, the system has either to be designed for the worst case or significant information about the transmitted data has to be known at design time. One example is an algorithm for wire placement in address buses [4]: A lot of information about the switching activity (data patterns) on the bus is needed to achieve good results. For data buses carrying random data, however, this approach is not useful.

Many other approaches aim at reducing the bus crosstalk energy by minimizing the number of simultaneous transitions on adjacent bit-lines. A good summary of approaches to reduce the number of transitions can be found in [5] which describes a dictionary-based encoding scheme for data buses. The authors investigated various data flows and found correlation between adjacent bus lines. At the expense of additional hardware (so that encoding/decoding can take place within one cycle), a dictionary-based compression method can be used to transfer less data, thereby causing less signal transitions.

Another possibility to reduce crosstalk energy is to use time redundancy. It was shown in [6] that it is very energy efficient to detect errors and retransmit erroneous data. By comparing various codes, the authors found that more powerful codes imply higher power consumption due to more complex encoders and decoders. The best energy efficiency has been achieved using a detection scheme with retransmit.

The work presented in [7] is targeted towards encoding the bus in a way that minimizes crosstalk. The authors present a special encoding, which is shown to reduce the crosstalk delay variations by a factor of two. The main drawback is that an area overhead of more than 60% is required, and an important finding is that encoding for low crosstalk and low energy may not be compatible since worst-case crosstalk patterns can occur among the patterns that have very low energy consumption.

Electro-Magnetic Interference. Electro-magnetic interference clearly is a mechanism that can induce transient errors. And on-chip interconnections are very likely to be the victims of such influence because of their relatively long wires. Errors due to electro-magnetic interference can thus be expected on the buses between the switches of the Network-on-Chip.

With higher integration of more complex building blocks on one chip, electrical noise from the environment will no longer be the only error source. Especially mixed-signal ICs will also be susceptible to distortions generated on-chip by HF components or by the large number of devices switching simultaneously at high clock frequencies.

Radiation. Transient faults caused by alpha particles (emitted by the package) or cosmic radiation have first appeared in semiconductor memories, which are typically the most highly integrated devices fully exploiting a given technology's capabilities. Error correcting codes have been successfully applied to these memories in order to minimize the effect of both transient errors and permanent errors introduced during the manufacturing process (resulting in an improved yield).

Due to decreased node capacitances, logic [8] and low capacitance/low power buses [9] are likely be affected by radiation-induced errors as well.

Process Variations. At decreasing feature size, unavoidable process variations may lead to a greater variance of circuit parameters. While transistors are the devices whose parameters are easily changed by process variations, the effect on the high number of interconnects on an always-increasing number of layers has also to be taken into account. Even a small change in the cross-coupling capacitances may lead to additional delays on several or all lines of a bus, thereby increasing the delay variation.

Other Error Sources. Given the high costs of microchip production (masks, manufacturing, testing etc.) that increase with each technology generation, it may become necessary to relax the current, inherent requirement of 100% correctness for devices and interconnects on purpose. According to the International Technology Roadmap for Semiconductors, this paradigm shift might be forced during the time-frame the NoC-architecture is developed for, since it may be the only means to reduce production, verification and test costs per chip by increasing the yield significantly [1].

Fault-tolerance will then not only be implemented to cope with runtime errors but also to mask out a certain amount of permanent errors introduced during manufacturing. This could lead to a characterization of acceptable errors, which will not render the chip unusable but can be tolerated by the implemented error-control scheme.

Furthermore, it is expected that the number of soft errors only occurring under specific conditions of voltage, timing and temperature will increase. Since these types of errors are difficult or impossible to test for, they will lead to an higher overall rate of transient errors during circuit operation.

While interconnect reliability has been tackled successfully on the physical layer in the past, this will not necessarily be possible in future deep submicron NoC-applications. Measures like shielding (adding a ground wire between two signal wires) might be too expensive in a highly integrated network environment built of massively parallel buses. Furthermore, Worm et. al. raise the question whether maintaining all design margins for on-chip buses to cope

with worst-case manufacturing variations will be too expensive in terms of area consumption. [10]

Taking all these arguments into consideration, it becomes clear that guaranteeing completely reliable links between the NoC-switches by manufacturing – while desirable – will perhaps not be the most efficient solution when both technical and economical arguments are taken into account. When the physical layer of an on-chip network is not able to hide all errors that occur from the upper protocol layers, it is important to understand what the error sources are and which error characteristics are expected in order to design suitable error-tolerance schemes.

3. Application Demand and Quality of Service

An implementation objective of the on-chip network is the principle of communication between resources. This communication is not as tightly coupled as it could be if custom interconnect (along with a custom protocol) was used. This leads to a new way of looking at SoC design with a strong emphasis on communication. After all, a future NoC-application will rather resemble a distributed system than a traditional SoC.

When an application is mapped onto a NoC-platform, it is split into multiple blocks which need to communicate. Likewise, when multiple applications are running on one NoC, they must be able to interact. While this is not new and requirements for specific communication have formerly been implemented by buses or custom designed connections, the NoC-approach aims at transferring all inter-block communication over a chip-wide uniform network infrastructure.

This in turn means that the network must be so versatile that it can handle (adapt to) a large number of communication streams of which each one can have its own characteristics. These characteristics will certainly not be uniform. Therefore, a closer look at parameters for traffic description is appropriate. Since communication aspects will play a major role in future NoC-based SoC design, applications will have to explicitly describe their communication streams.

Especially in a large-scale NoC that handles a significant number of concurrent communication streams, the traffic will not be uniform. Supporting different types of traffic can be rewarding and efficient in terms of resource usage and overall transmission times. An example for this are the advantages of combining guaranteed-throughput and best-effort traffic shown in [11] for the Æthereal network.

An overview of parameters that will be helpful in describing on-chip communication streams is listed in Table 6.1.

Parameter	Description
latency	maximum latency/delay allowed
bandwidth	minimum bandwidth required
completion time	maximum duration of a certain transmission
traffic distribution	e.g., burstiness
resilient error rate	error rate that is tolerable

Table 6.1. Quality-of-Service Parameters to describe communication streams

A set of parameters like these enables applications to describe their communication connections in a form that is advantageous for the NoC. Based on these parameters, the network determines whether and how a connection request can be served.

4. Error-Tolerance Schemes

Combining the contradicting requirements presented in the previous sections leads to a trade-off in terms of error-tolerance: Which error characteristics have to be coped with? Which QoS guarantees have to be realized? This gap must be closed by an error-tolerant interconnect scheme.

Obviously, a NoC can be customized and optimized to be a platform for a class of applications. On the one hand is the implementation of error-tolerance schemes guided by constraints specific for an application-class (e.g., real-time applications). On the other hand impose architecture- and technology-dependent constraints limits on the design freedom.

Error-tolerance can only be achieved by adding some form of redundancy so that errors can be compensated. As the following list suggests, there are many approaches possible, each with its unique advantages and shortcomings.

1 information redundancy (additional information is transported)

   (a) additional information on one link (e.g., a checksum)

   (b) identical information sent over multiple links (broadcast to increase the probability that this information will reach its intended recipient)

2 time redundancy (e.g., repeated transmission or retransmission on request)

3 space redundancy (multiple components working in parallel, the output is typically determined by a majority vote)

The important questions that have to be answered when deciding on an error-tolerance scheme are not only which of the above approaches to achieve redundancy is chosen but also where the error-tolerance scheme is implemented.

In a NoC, error-tolerance can be implemented at various levels. One option is to implement error-control measures on the switch-to-switch links, another option is to define a protocol between connection end-points to cope with packets which were lost or garbled on their way through the network. Regardless of the type of redundancy chosen, implementing switch-to-switch error control usually has the advantage that errors cannot accumulate when the data is transported over multiple hops but are noted as soon as they appear. On the other hand is the design simplified when only end-to-end error-tolerance measures are used.

Typically, a good solution will be comprised of both switch-to-switch and end-to-end error-tolerance schemes, fine-tuned to complement one another.

When making design decisions, it is important to keep in mind that a typical NoC is comprised of a large number of identical network components, namely switches and network interfaces. This leads to energy and area constraints which influence both how much computational complexity can be included in network components and where which amount of buffers can be used. For instance has the number of buffers or pipeline-stages in switches a direct influence on the latency of a communication stream.

When designing a NoC, many parameters can be controlled directly, for sure an advantage in comparison to traditional network concepts. Additionally, in comparison to larger-scale networks new parameters become critical and must be considered during network design in order to achieve a well-balanced trade-off. The most obvious one is energy consumption, a major concern in SoC design.

This motivates a look at error-tolerance schemes from an energy-efficiency viewpoint. In contrast to traditional networks is the energy consumed by encoding/decoding logic in the same order of magnitude as the energy needed for transmitting data. While low-power transmission schemes are susceptible to transmission errors, they will nevertheless prevail future NoC-designs because communication energy minimization will be the only means to meet global energy consumption limits. The result of [6] where the energy-efficiency of transmission schemes was examined indicates that a combination of error-detection and retransmission (if necessary) can be very efficient. Approaches incorporating more complex codes to protect the transmitted data needed too much energy for their encoders and decoders in comparison to the retransmissions they could avoid.

In order to be able to retransmit erroneous packets, buffering has to be implemented into the on-chip network. Additionally, communication between network components becomes necessary so that retransmissions can be requested. Typically, positive or negative acknowledgements along with some sort of sequence number are employed.

The most simple retransmission scheme expects an acknowledgement for each sent packet. This can either be a positive acknowledgement if the packet was successfully received or a negative acknowledgement indicating a transmission error. Optimizations of this basic protocol incorporate sequence numbers to acknowledge a whole group of packets or selectively request retransmission of erroneous packets. These actions aim at reducing the number of acknowledgement packets that have to be sent and result in both a lower network load and higher throughput. Problems with this kind of protocols usually arise when acknowledgement packets are scrambled or lost. Therefore, timers are implemented and the resulting protocols can become very complex due to the large number of possible states.

An approach different to costly error- and flow-control are broadcast-based transmission schemes. For instance, [12] proposes a so-called "gossip protocol" which transmits data from sender to receiver over multiple routes in a broadcast fashion. Since every node broadcasts received packets, the data will eventually reach its intended destination. However, the additional network load this scheme causes in comparison to unicast transmission is hardly compensated by its conceptual simplicity.

As this short overview illustrates, there are as many potential sources of faults and errors threatening future systems on chip as there are strategies and techniques to protect data from corruption and loss. In the remaining sections of this chapter we become more concrete and focused and discuss a two level error protection scheme for a particular NoC architecture, the Nostrum. Even more specifically we focus mostly on the data protection from errors occurring on the wires between the switches of the communication network.

5. Error-Tolerant Interconnects in the Nostrum NoC

5.1 The Nostrum Architecture

The Nostrum NoC architecture [13] is based on a regular $n \times m$ mesh of (network) switches and resources. Every resource is connected to one switch and the switch can be connected to up to four neighbors. The network is based on packet switching with packets of a fixed size determined by the width of the connection between adjacent switches. This is a massively parallel bus structure that allows for bidirectional transmission of two network packets each transfer cycle.

In the current implementation, a packet is 128 bit wide and comprised of both a header and application payload. The Nostrum NoC transports a whole network packet over the wide links between the network switches each cycle. Every switch can send and receive up to five packets at the same time. Since the switches use deflective routing [14, 15] and neither pipelining nor buffering

takes place in the switches, no retransmission of scrambled packets between adjacent switches is possible.

This buffer-less design style is motivated by area and power consumption. Introducing buffers into switches would mean to introduce buffers to every input or output connection. Since each switch has five unidirectional inputs and five outputs, each 128 bit wide, the area overhead introduced by the network components would drastically increase. Without buffers, however, tight speed-constraints are introduced: The switch must be designed in a way that allows for fast analysis of the packets. All packets have to be analyzed before the switching decision can be made.

5.2 Connection Types

Nostrum offers best effort (BE) and guaranteed latency (GL) communication services. BE communication is based on individual packets that are communicated and routed independently. Hence, each of these packets contains a relatively long header with the full address information. Since BE packets can deviate from the shortest path between source and destination when traffic load is high [14, 15], no maximum delay can be guaranteed. However, packets that have been longer in the network increase their priority and thus their chance to reach the destination. In contrast, GL communication is based on a *virtual circuit* that is a direct stable connection between a source and a destination. A virtual circuit has to be opened and closed. A virtual circuit is opened by reserving particular time slots in every switch from source to destination. These time slots cannot be used by other packets. Consequently, packets traversing an open virtual circuit cannot be disturbed by other BE or GL packets, thus they experience a deterministic latency from source to destination. The GL packets have a much shorter header without address information, because the switches know where GL packets are to be sent.

The discussion below about error protection is valid for both BE and GL packets. Thus the different reliability services are available for both kinds of traffics. The only difference is that GL packets have a much shorter header and therefore allow for a simplified protection scheme in some of the services described below. For the sake of simplicity we therefore discuss only BE traffic in the following sections.

5.3 Quality of Service for Data Integrity

As discussed above, implementing an on-chip network to handle all inter-IP traffic benefits from offering different QoS-characteristics to its applications.

End-to-End Services. Nostrum offers a two level data protection scheme: end-to-end and link layer data protection. The end-to-end service denotes a

technique implemented in the network interfaces between the resources and the communication network. We distinguish two modes: the *send-and-forget* (SaF) service and the *acknowledge-and-retransmit* (AaR) service. We only sketch a possible end-to-end protection mechanism very briefly here and concentrate then on the link layer data protection in the following sections.

In SaF mode no additional action at the network interface is taken, based either on the assumption that the underlying communication is lossless and fully reliable, or because the application can tolerate loss and data corruption at the level provided by the lower layers.

AaR mode defines a window of size $N, 1 \leq N \leq 64$. The receiver, i.e. the network interface at the receiver side, sends an acknowledgement packet to the sender for each N received packets. The acknowledgement packet has one bit for each of the N packets indicating if that packet should be retransmitted by the sender. There are three possible reasons why the receiver requests a retransmission of a packet. (a) The packet has been flagged "corrupted" by the link level data protection mechanism (see below). (b) An end-to-end error detection code, again implemented in the network interface, has detected a corrupted packet. (c) A packet has not arrived after a maximum elapsed time. This maximum time can be determined easily for guaranteed latency packets but has to be set heuristically for best effort traffic.

The sender in an AaR communication buffers $2N$ packets. After having sent N packets it expects the corresponding acknowledgement packet while keeping sending packets. If $2N$ packets have been sent and the first N packets have not been acknowledged, the sender stops and waits for the acknowledgement packet. When it arrives, the sender retransmits requested packets and removes the positively acknowledged packets from its buffer. Then the normal transmission of packets can be resumed.

As mentioned this is only a sketch of a possible data protection mechanism. For a complete AaR protocol all time-out periods and the end-to-end data encoding have to be decided. Also, the actions have to be defined for the case when data corruption and loss increase beyond an acceptable level and jeopardize for example maximum latency guarantees.

In the following we elaborate in more detail the lower level protection mechanism and define four reliability levels for the link layer. It should be noted that all four levels can be combined arbitrarily with both the SaF and the AaR mechanisms leading to eight distinct data protection classes. All of them can be applied to both best effort and guaranteed latency services. Thus, an application can choose from a variety of communication services with different performance and reliability properties.

Link Level Services. For the Nostrum link layer we focus on protecting data from errors occuring on the wires between the switches and not in the

switches themself. This is motivated by the fact that the area of the switch to switch interconnect is significantly larger than the area of the switch. In Nostrum the resources form a two dimensional mesh. Making the assumption that the resources have a footprint of $2\mathrm{mm} \times 2\mathrm{mm}$ and the switches have a footprint of $100\mu\mathrm{m} \times 100\mu\mathrm{m} = 10^4\mu\mathrm{m}^2$, the wires between two switches will occupy an area of $2\mathrm{mm} \times 100\mu\mathrm{m} = 2 \cdot 10^5\mu\mathrm{m}^2$, i.e. 20 times the area of the switches. Under the simplifying assumption that the the number of faults is proportional to the area, we expect many more faults to affect the inter-switch wires than the switch logic. Also, we hope to capture the faults in the switches with the end-to-end data protection scheme. However, these assumptions need yet to be substantiated with realistic fault models of future technology generations and quantitative experiments.

In Nostrum, link level error protection is achieved by bus encoding. We propose four QoS-classes describing the characteristics of the switch-to-switch transport of a packet. The characteristics of these service classes range from offering maximum bandwidth at the expense of error tolerance to reduced application bandwidth allowing for more redundancy so that tighter integrity or latency bounds can be met: While the transport of multimedia data may require maximum bandwidth, data integrity is the main concern for transfers from and to memories.

Maximum Bandwidth. Since the amount of wires available for routing the inter-switch buses is limited, the redundancy introduced by coding will directly influence the application payload that can be transported in a packet. Consequently, maximum bandwidth is available to the application when no encoding is applied.

Guaranteed Integrity. To ensure data integrity as far as possible, error detection methods can be used. Data correction is not attempted since it might result in miscorrection. Instead, if erroneous data is detected, a flag can be set or the whole packet can be dropped. Due to the lack of packet buffering in Nostrum's switches, however, no immediate retransmission can be performed. Thus, the main property of this mode is that if the data arrives and no error has been detected, it can be assumed to be correct.

Minimum Latency. To achieve minimum latency, packets must always be forwarded. Only error correction is performed, the underlying assumption is that all errors can be corrected by the code used. This mode is well suited for applications that can tolerate rare errors (due to miscorrection) but depend on receiving data at a constant rate.

It is worthwhile noting that this mode cannot decrease the latency which mainly depends on the routing process in the Nostrum NoC. *Minimum Latency*

tries to ensure that a packet arrives at its destination without additional delay introduced by error-tolerance measures.

High Reliability. By using codes capable of correcting errors and detecting uncorrectable errors at the same time, the characteristics of the two previously described modes can be combined at the expense of lower bandwidth: Since more redundancy is necessary, less application payload can be transported per packet.

5.4 Error-Control Coding on the Switch-to-Switch Links

We assume a NoC architecture that requires very low area overhead and high performance from the switches. Thus, for error-control between adjacent switches we consider only codes that can process 100-200 bit wide buses in a single clock cycle and that can be implemented in a few gates of hardware for each bit.

In fact, we focus on parity-based, linear block codes, for the necessary encoders/decoders can be implemented by combinatorial logic, namely EXOR-trees. This helps to meet the tight speed constraints discussed above, because neither time-consuming multi-cycle decoding operations are necessary nor requires the implementation the parallelization of a multi-cycle algorithm, which is usually costly in terms of area (e.g., due to loop unrolling etc.).

A code protecting data on the switch-to-switch buses must allow for fast decoding because decoding (at least of the packet header) has to be completed before the switch can make a routing decision. Additionally, area constraints motivate a switch design with as few gates as possible.

Since the requirement of fast decoding can be fulfilled by combinatorial logic circuits of low logic depth, parity-based codes (e.g., Hamming or Hsiao codes [16]) are considered. These can be employed at various coding schemes: A single-error correcting (SEC) code can correct all single errors. All other errors lead to miscorrection. A double-error detecting (DED) code, on the other hand, detects all single and double errors. In fact, a DED code detects even more errors: If there are 2^k valid codewords of length n, $2^n - 2^k$ error patterns are detected (i.e. all patterns that do not represent a valid codeword). An interesting property is that every SEC code can alternatively be used as a DED code. The properties of these two operating modes are combined in SEC-DED codes which require one more bit of redundancy to encode the same amount of information. Due to that, the error detection capability of SEC-DED codes is always slightly less than that of a comparable DED code. Using codes with the ability to detect/correct more than two random errors usually makes decoding slower because arithmetic decoding becomes necessary [16].

Later, it will be shown how these codes can be employed to implement different protection levels corresponding to multiple QoS requirements.

5.5 Fault Model

From the discussion of faults which will effect future DSM circuits it stands to reason that the errors imposed by these faults will appear clustered both in time and in space. While faults may easily affect multiple adjacent wires, or cause errors in subsequent time steps (e.g., due to delay variations), the probability that they influence wires further away or for a longer period of time will be significantly less.

For the simulation results reported later on, we assumed a combination of three statistically independent error sources, each of the errors was assigned a probability of occurrence of 10^{-9} per wire and time-step. The first two error sources generate simple error patterns that account for errors due to multiple random effects. One source causes errors limited to just one wire, the second one almost always affects two (and very seldom three) adjacent wires. While these two models are quite simple, the third one was extracted from a more detailed experiment and hence attempts to be a little more realistic.

Knowing that the inter-wire capacitance C_I between adjacent wires of a bus has a significant influence on the total capacitance which has to be charged during a bus transition, it can be derived that the signal delay depends strongly on the bus transition pattern. Considering three adjacent wires, transition patterns can be classified in 1C. . . 4C sequences depending on how much they slow down the transition of the middle signal [7].

Name	Transitions
4C	$v_1\overline{v_1}v_1 \longrightarrow \overline{v_1}v_1\overline{v_1}$
3C	$v_1\overline{v_1}v_2 \longrightarrow \overline{v_1}v_1v_2$
	$v_2v_1\overline{v_1} \longrightarrow v_2\overline{v_1}v_1$
2C	$v_1v_2v_3 \longrightarrow v_1\overline{v_2}v_3$
1C	$v_1v_2v_2 \longrightarrow v_1\overline{v_2}v_2$
	$v_2v_2v_1 \longrightarrow \overline{v_2}\overline{v_2}v_1$

Table 6.2. Classification of crosstalk sequences

Out of a large number of random data patterns, we identified all 4C sequences (i.e., the ones causing maximum signal delay) and marked the middle wire as fault location.

The error patterns extracted from this simulation were formulated according to a fault model notation which describes error duration and number of affected adjacent wires [17, 18]. For instance, at most 7 adjacent wires were affected at the same time. These error patterns served as third error source in subsequent simulations. More details on this along with a formal fault model notation can be found in [17, 18]. Obviously, these simulations have to be re-evaluated when more realistic information about fault scenarios in future technology becomes available. While this will change the numbers and thus

can have an impact on design decisions, the general observations discussed below will remain valid.

5.6 Tuning Error-Tolerance by Interleaving

As discussed above, linear block codes were chosen for their simple implementation in switches. During the encoding, a certain amount of redundancy is added to the information bits in form of parity bits. Due to this redundancy, the decoder is able to detect errors and/or to recover the sent information even if the codeword has a limited number of erroneous bits.

Since an encoding that can tolerate only a very low number of errors might not provide the necessary robustness, one possibility is to divide the network packet into several smaller blocks encoded separately. The so-called *parallel coding* increases the redundancy, but enhances the overall error detection and correction capabilities and reduces the length of the critical path in the decoder. This approach also enables using different codes for different parts (blocks) of the packet.

Also knowing that errors are likely to appear locally clustered motivates the idea of using interleaving: If the data is split into multiple blocks encoded separately, the n bits of a data block can be assigned to n adjacent wires of a bus or they can be interleaved, so that they will be assigned to the wires $x, x+\delta, x+2\delta, \ldots, x+(n-1)\delta$. The distance δ between two wires that belong to the same block is called *interleaving degree*. Based on this definition, assigning a block of n bits to adjacent wires is interleaving with degree 1.

To compare the capabilities of different coding schemes, we use the probability of uncorrected and undetected errors in a packet, $P_{err,UC}$ and $P_{err,UD}$ respectively. It can be seen from Figure 6.1 that in the presence of multi-bit errors, interleaving can lead to the reduction of those probabilities by several orders of magnitude. At small interleaving degrees, multi-wire faults causing errors on adjacent bit-lines may easily affect multiple bits of one block. If this block is protected by a SEC-DED code, more than one erroneous bit will not be correctable and more than two erroneous bits may cause error detection to fail. With increasing interleaving degree, the probability that a fault affects multiple wires of one block decreases. Instead, multiple blocks will have an erroneous bit each. This can be tolerated if each block is protected by an appropriate code. In the fault scenario used, faults can affect up to 7 adjacent wires. This is why in Figure 6.1, $P_{err,UC}$ and $P_{err,UD}$ constantly decrease with increasing interleaving degree. For interleaving degrees greater than 7, no further improvement can be achieved for the fault hypothesis used.

Encoding multiple small blocks of data can considerably increase the amount of parity signals, i.e. the redundancy. It has, however, several advantages: First, the decoder logic will have a low logic depth. Second, the smaller the

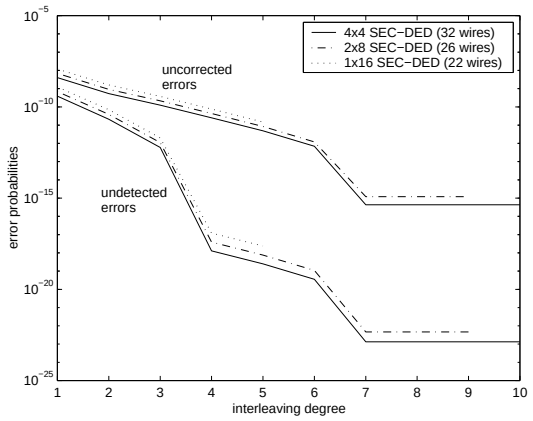

Figure 6.1. Reduction of error probabilities by interleaving

block-size, the higher interleaving degrees are possible. When locally clustered errors are assumed, this usually leads to a significant decrease in the probability of errors that cannot be corrected or detected.

For Nostrum, fast and simple decoder logic was a main requirement. Thus, this method to increase the bus reliability was examined. Given a different set of constraints, a completely different approach might have been chosen. Since the proposed method has the potential of adding a large amount of redundancy to the data transported, the appropriate encoding scheme has to be selected carefully.

5.7 Design Flow

In a packet-switched network, each packet typically carries header and payload. Thus, routing is done on a per-packet basis and each header has to be inspected. The importance of the header's correctness for the network's functionality is obvious. Therefore, the network has to take care of transporting the header information in a way that is as error-tolerant as possible while each application can select a protection level for the data it sends over the network (i.e., the payload part of the packet). Applications cannot, however, influence the header protection scheme.

Arguing that the header is most important naturally leads to the following design flow proposal for selecting protection schemes:

1 Select a header encoding scheme that fulfills a minimum reliability constraint to ensure correct network operation.

2 Determine possible payload encoding schemes and their characteristics.

3 Select appropriate payload encoding schemes to meet the QoS-requirements set by the application class the NoC is designed for.

For the Nostrum architecture which is based on a fixed number of wires connecting adjacent switches, the encoding scheme selected for the header directly influences the possibilities to choose from for payload encoding since transmitting the encoded header needs a certain amount of the fixed number of available wires. This implies at the same time that not necessarily all combinations of header and payload error tolerance requirements can be fulfilled.

5.8 Implementation Example

This example is intended to show how the proposed design flow and encoding scheme can be used in the implementation of a NoC capable of carrying traffic of multiple QoS-classes.

We assume here a 128 bit wide bus between adjacent switches, and 16 bit of header information. The figures given are based on estimation results which were determined using the fault scenario discussed above.

Header Protection. Since the header protection should correct as many errors as possible while at the same time leaving as little errors as possible undetected, the obvious choice is a SEC-DED code.

The packet header (16 bits) can be encoded as one block. In this case, 22 wires are required and the code is referred to as $1 \times (22, 16)$ SEC-DED code. The notation $b \times (n, k)$ is used to denote an encoding scheme in which b blocks are independently encoded by a code which transmits k information bits within codewords of n bits.

Alternatively, better error resilience can be achieved if the 16 bit wide header is encoded as 2 blocks of 8 bits each ($2 \times (13, 8)$ SEC-DED code) or as 4 blocks ($4 \times (8, 4)$ SEC-DED code).

Code	Wires	Max. interl.	Crit. path	$\mathbf{P_{err,UC}}$	$\mathbf{P_{err,UD}}$
no protection	16	-	0	$5.33 \cdot 10^{-8}$	$5.33 \cdot 10^{-8}$
$1 \times (22, 16)$ SEC-DED	22	5	9	$1.47 \cdot 10^{-11}$	$2.37 \cdot 10^{-18}$
$2 \times (13, 8)$ SEC-DED	26	9	8	$1.21 \cdot 10^{-15}$	$4.63 \cdot 10^{-23}$
$4 \times (8, 4)$ SEC-DED	32	16	6	$4.33 \cdot 10^{-16}$	$1.32 \cdot 10^{-23}$

Table 6.3. Comparison of header protection with different SEC-DED codes

Table 6.3 shows the amount of wires required for these different encoding schemes along with information about the maximum interleaving degree, expected decoder logic depth and the probabilities of uncorrected and undetected errors ($P_{err,UC}$ and $P_{err,UD}$) when maximum interleaving degree is used to

allocate bus lines. As a comparison the first line shows the error probability without a protecting code.

A constraint of one undetected error in the header of transported packets per year in a NoC with 100 buses and an operation frequency of 1GHz translates to $P_{err,UD} \leq 3.1710 \cdot 10^{-19}$. Based on the information from Table 6.3 and complementing experiments, the $2 \times (13, 8)$ SEC-DED code was chosen as header encoding scheme for this example. It is a reasonable trade-off between the remaining error probability and the decoder speed (critical path). Since this code requires 26 wires, 102 of the 128 wires remain to transmit the payload.

Payload Protection. Regarding the four Quality-of-Service classes discussed previously, the *Maximum Bandwidth* mode is inherent in every implementation since it does not use any payload encoding. In the current example, this mode's bandwidth is 102 bits/packet and the probability of an erroneous transfer is $3.36 \cdot 10^{-7}$.

Both the *Guaranteed Integrity* and *Minimum Latency* modes can be implemented with one code, using the property that every SEC code can also work as DED code: The same codewords are used for transmission. Hence, the encoder is identical and the DED code just uses the first part of the decoder block (which detects erroneous transmissions) but not the second one which performs correction and thereby alters the received data if necessary. Thus, these two services offer the same bandwidth to the application, but different protection characteristics. It is very advantageous that implementing these two QoS-classes at approximately the cost required for implementing just one of them is possible.

In general, the selection of an appropriate coding scheme is guided by reliability and bandwidth constraints. Tunable parameters are the number of parallel encoded blocks and the interleaving degree. The trade-off for implementing *Guaranteed Integrity* and *Minimum Latency* on 102 wires is shown in Table 6.4. Depending on the code used, the application bandwidth and resilient

Code	Max. interl.	Avail. bandw.	used as SEC $\mathbf{P_{err,UC}}$	used as DED $\mathbf{P_{err,UD}}$
$1 \times (102, 95)$	1	95	$5.68 \cdot 10^{-8}$	$6.13 \cdot 10^{-9}$
$2 \times (51, 45)$	2	90	$3.77 \cdot 10^{-9}$	$1.71 \cdot 10^{-10}$
$3 \times (34, 28)$	3	84	$5.88 \cdot 10^{-10}$	$3.20 \cdot 10^{-12}$

Table 6.4. Error probabilities for different operation modes of SEC (or DED) codes using 102 physical wires

error probability vary. Out of these coding schemes, the appropriate one can be chosen at design-time to serve as bus encoding for *Guaranteed Integrity* and *Minimum Latency*.

The fourth mode, the *High Reliability* mode is based on a SEC-DED code. Hence, it will require different encoder/decoder logic in parallel to that used for the two previous modes. Note that SEC-DED codes offer one information bit less per block compared to SEC codes occupying the same amount of wires.

For the 128 bit bus of our example, we have finally selected the coding schemes from Table 6.5 to implement the four QoS levels. The selection of

Mode	Code / Application bandwidth		$\mathbf{P_{err,UC}}$	$\mathbf{P_{err,UD}}$
maximum bandwidth	none	102	$3.36 \cdot 10^{-7}$	$3.36 \cdot 10^{-7}$
guaranteed integrity	$4 \times (25, 20)$ DED	64	$9.82 \cdot 10^{-8}$	$1.59 \cdot 10^{-17}$
minimum latency	$4 \times (25, 20)$ SEC	64	$8.64 \cdot 10^{-11}$	$8.64 \cdot 10^{-11}$
high reliability	$3 \times (34, 27)$ SEC-DED	65	$5.88 \cdot 10^{-10}$	$3.20 \cdot 10^{-12}$

Table 6.5. Selected coding schemes for payload encoding

services and coding schemes a NoC supports is a design-time decision which depends on the intended application and environment, whereas each packet can be encoded with one of these codes during run-time. Since subsequent packets can be encoded differently, the information about the payload encoding scheme is transmitted as part of the header information.

Implementation. Figure 6.2 gives an example of how the input stage of a switch could work: the received packet is split into header and payload. After the decoding of the header information, the coding scheme of the payload is known and the appropriate decoder output can be selected. The decoded destination address is used to make the switching decision which finally leads to the selection of the output port the data is forwarded to.

It is important to note that it is not necessary to decode the data and to encode it again in a second block. The decoders can rather work in a way that their output is not the decoded and – if necessary – corrected information, but the associated codeword instead. This means that their input is the received data and their output a valid codeword which can be directly passed to the next bus.

6. Conclusion

We have presented various important aspects in implementing error-tolerant interconnect schemes which are crucial to a NoC since they encapsulate the low-level and implementation details of the NoC-platform. Thereby, they are helpful in offering services with defined characteristics. These services – which form the interface between platform and application – were discussed in detail using the Nostrum NoC as example. It provides a total of eight QoS-classes by combining both link-level and end-to-end error-tolerance. The ser-

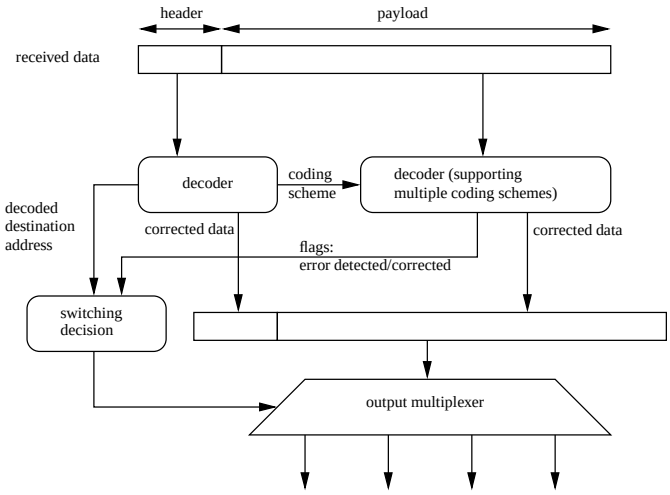

Figure 6.2. Applying error correction to incoming packets in a switch

vices offered by Nostrum were discussed in great detail showing that link-level (i.e., switch-to-switch) error-tolerance within the network in combination with end-to-end error-tolerance allows for efficient adaptive protection schemes.

Since error-tolerance profits from being implemented as low in the protocol stack as possible, we focused on switch-to-switch error protection, presenting a bus encoding scheme that allows for fast and simple decoding while supporting multiple QoS-characteristics. As described in the introductory sections, the development of this error-tolerance scheme was guided by specific constraints and it should be kept in mind that other constraints (especially if pipelining or packet retransmission was possible) could have lead to a different implementation approach.

References

[1] ITRS. International Technology Roadmap for Semiconductors, 2001.

[2] Michael Cuviello, Sujit Dey, Xiaoliang Bai, and Yi Zhao. Fault modeling and simulation for crosstalk in system-on-chip interconnects. *1999 IEEE/ACM International Conference on Computer-Aided Design*, pages 297–303, 1999.

[3] Michele Favalli and Cecilia Metra. Bus crosstalk fault-detection capabilities of error-detecting codes for on-line testing. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 7(3):392–396, September 1999.

[4] Luca Macchiarulo, Enrico Macii, and Massimo Poncino. Wire placement for crosstalk energy minimization in address buses. In *Proceedings of the 2002 Design, Automation and Test in Europe Conference and Exhibition (DATE 2002)*, pages 158 – 162, 2002.

[5] Tiehan Lv, Joerg Henkel, Haris Lekatsas, and Wayne Wolf. An adaptive dictionary encoding scheme for soc data buses. In *Proceedings of the 2002 Design, Automation and Test in Europe Conference and Exhibition (DATE 2002)*, pages 1059 – 1064, 2002.

[6] Davide Bertozzi, Luca Benini, and Giovanni De Micheli. Low power error resilient encoding for on-chip data buses. *Proceedings of the 2002 Design, Automation and Test in Europe Conference and Exhibition (DATE 2002)*, pages 102–109, 2002.

[7] Chinjie Duan, Anup Tirumala, and Sunil P. Khatri. Analysis and avoidance of cross-talk in on-chip buses. *Hot Interconnects 9*, pages 133–138, 2001.

[8] Y. Tosaka, S. Satoh, T. Itakura, K. Suzuki, T. Sugii, H. Ehara, and G. A. Woffinden. Cosmic ray neutron-induced soft errors in sub-half micron cmos circuits. *IEEE Electron Device Letters*, 18(3):99 – 101, March 1997.

[9] Marcello Lajolo, Matteo Sonza Reorda, and Massimo Violante. Early evaluation of bus interconnects dependability for system-on-chip designs. 14^{th} *International Conference on VLSI Design*, pages 371–376, 2001.

[10] Frederic Worm, Patrick Thiran, Paolo Ienne, and Giovanni De Micheli. An adaptive low-power transmission scheme for on-chip networks. In *Proceedings of the* 15^{th} *International Symposium on System Synthesis*, pages 92 – 100, October 2002.

[11] Kees Goossens, John Dielissen, Jef van Meerbergen, Peter Poplavko, Andrei Rădulescu, Edwin Rijpkema, Erwin Waterlander, and Paul Wielage. Guaranteeing the quality of services in networks on chip. In Axel Jantsch and Hannu Tenhunen, editors, *Networks on Chip*. Kluwer Academic Publishers, 2003.

[12] Tudor Dumitras, Sam Kerner, and Radu Mărculescu. Towards on-chip fault-tolerant communication. In *Proceedings of the 2003 Asia and South Pacific Design Automation Conference (ASP-DAC 2003)*, pages 225 – 232, January 2003.

[13] Shashi Kumar, Axel Jantsch, Juha-Pekka Soininen, Martti Forsell, Mikael Millberg, Johnny Öberg, Kari Tiensyrjä, and Ahmed Hemani. A network on chip architecture and design methodology. In *Proceedings of IEEE Computer Society Annual Symposium on VLSI*, pages 105 – 112, April 2002.

[14] Erland Nilsson. Design and implementation of a hot-potato switch in a network on chip. Master of science thesis, Royal Institute of Technology (KTH), Stockholm, Sweden, June 2002.

[15] Erland Nilsson, Mikael Millberg, Johnny Öberg, and Axel Jantsch. Load distribution with the proximity congestion awareness in a network on chip. In *Proceedings of the Design Automation and Test Europe (DATE)*, pages 1126–1127, March 2003.

[16] T.R.N. Rao and E. Fujiwara. *Error-Control Coding for Computer Systems*. Prentice-Hall International, 1989.

[17] Heiko Zimmer. Fault modelling and error-control coding in a network-on-chip. Master of science thesis, Royal Institute of Technology (KTH), Stockholm, Sweden, December 2002.

[18] Heiko Zimmer and Axel Jantsch. A fault model notation and error-control scheme for switch-to-switch buses in a network-on-chip. In *Proceedings of the Frist IEEE/ACM/IFIP International Conference on Hardware/Software Codesign & System Synthesis*, pages 188 – 193, October 2003.