Self-Timed NoC Links Using Combinations of Fault Tolerance Methods

Teijo Lehtonen[1,2], Pasi Liljeberg[2], Juha Plosila[2,3]
[1]Turku Centre for Computer Science (TUCS), Joukahaisenkatu 3-5 B, FIN-20520 Turku, Finland
[2]University of Turku, Department of Information Technology
[3]Academy of Finland, Research Council for Natural Sciences and Engineering
E-mail: teijo.lehtonen@utu.fi, pasi.liljeberg@utu.fi, juha.plosila@utu.fi

Abstract

We propose link structures for NoC that have properties for tolerating efficiently transient, intermittent and permanent errors. The protection against transient errors is realized using Hamming coding and interleaving for error detection and retransmission as the recovery method. We introduce two approaches for tackling the intermittent and permanent errors. In the first approach, spare wires are introduced together with reconfiguration circuitry. The other approach uses time redundancy, the transmission is split into two parts, where the data is doubled. In both structures the presence of permanent or intermittent errors is monitored by analysing previous error syndromes. The links are based on self-timed signaling in which the handshake signals are protected using triple modular redundancy.

1. Motivation

When designing a fault tolerant on-chip link for a deep submicron chip, one should first consider the possible fault scenarios. The approach must be capable of tolerating multiple bidirectional errors as well as burst errors [1, 4]. For detecting multiple bidirectional errors the Hamming code is widely used. The standard version of it can detect two single errors and with one additional check bit the error detection ability can be extended to three. Cyclic codes can be used to detect burst errors [1]. Another approach for handling burst errors is interleaving [11].

The faults can be categorized into three classes: permanent, intermittent and temporary or transient faults [2]. Most of the failures (80%) are caused by transient faults [4]. The other way around, up to one fifth of all the failures are originating from permanent or intermittent faults. Thus, the fault tolerance approach must contain elements to not only tolerate the temporary errors but also the ones of more permanent nature. Most of the research has been concentrating on tolerating transient errors. The two methods for this purpose are forward error control (FEC) and automatic repeat query (ARQ), the latter of which is found to be more energy efficient [1]. The ARQ will not work in the presence of permanent errors and FEC with commonly used Hamming coding loses its effectiveness already in the presence of a single permanent fault. Fault tolerance methods besides coding include triple modular redundancy and the use of spare components together with error detection [6].

The idea in this work is to combine different fault tolerance methods to achieve a system that is efficient in tolerating transient, intermittent and permanent errors. For transient faults we use Hamming coding and interleaving to detect faults and ARQ as the recovery method motivated by its energy efficiency as reported in [1]. For tolerating permanent and intermittent errors we introduce two methods, one that uses hardware redundancy and the other based on time redundancy. In the first approach spare wires are introduced together with reconfiguration circuits. In the second approach, the data is split into two transmissions and in both of them the transmitted data is doubled. In the receiver the fault-free copy is chosen and the data of the two transmissions are again combined into a whole word. In both structures, the presence of a permanent fault is detected using the same Hamming code as for transient faults. A number of previous error syndromes are stored and if they equal, it indicates a permanent error. The exact error location can be determined by decoding the syndrome.

2. Link Structures

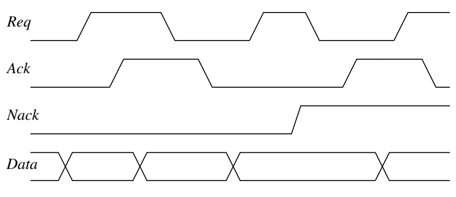
Communication between processing blocks in a NoC system typically needs synchronization which is error-prone. Also the clock distribution over a wide chip with low skew and jitter is problematic. A viable solution for this is the use of globally-asynchronous locally-synchronous (GALS) design approach, where the communication between processing blocks is done asynchronously. Therefore, we base our link on self-timed principles.

The width of the target link is set to 64 bits, which is split into four identical parts. Every part is encoded with Hamming (21,16) code and the parts are interleaved. Consequently, the system is capable of detecting error bursts affecting up to 8 adjacent wires and at least two simultaneous single faults extending to 8 simultaneous single faults if only two of them affect the same interleaving section.

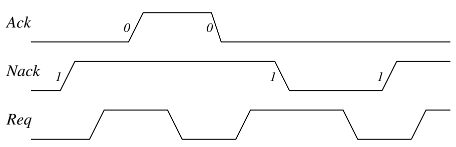
In the spare wire approach, four spare wires are added to the system, one for each interleaving section. This gives the system tolerance for permanent error bursts affecting up to 4 wires and maximum of 4 single faults if they affect separate interleaving sections. Thus, the total number of wires is 88, from which 64 are data, 20 check bits and 4 spares.

In the split transmission approach the data is split into two parts, two interleaving sections to each. The data in both parts is doubled, preserving the interleaving. Therefore, the error detection capability is 4-bit wide error bursts and two single errors. The minimum tolerance against permanent errors is determined by the wires located physically in the middle of the link, where the two doubled parts have the minimum physical distance to each others. Since there are four control signals between the parts (see below), the minimum permanent error burst tolerance is 6 bits extending up to 36 bits depending on which part of the link the errors affect. The minimum permanent single error tolerance is one, but a system having even 36 single errors works, if the errors occur only in two different interleaving sections.

The timing of the data transfer is realized using two-phase asynchronous bundled data signaling [10], illustrated in Figure 1(a), while internally the transmitter and receiver use 4-phase signaling. Two-phase signaling is chosen in order to minimize the control wire switching activity and the handshaking delay. Therefore, in addition to the data wires we need request (*req*) and acknowledgement (*ack*) signals. For signaling the incorrectness of a data transfer from the receiver to transmitter we use negative acknowledgement (*nack*) signal instead of *ack*. This makes the backward signaling delay-insensitive since there is no need for making timing assumptions. Finally, we need also a signal for indicating the reconfiguration (*reconf*). The actual reconfiguration data can be sent from receiver to transmitter serially by using self-timed dual-rail protocol presented in Figure 1(b) [10], where *ack*/*nack* signals are used for data (0/1) and *req* for acknowledgement. This way also the reconfiguration data exchange is delay-insensitive and no additional signals are needed.



(a) 2-phase bundled data.



(b) 2-phase dual rail (transmitting data "10011").

Figure 1. Self-timed signaling.

The timing signals are crucial for the correct operation of the link and therefore, they have to be protected against errors. For this purpose we use triple modular redundancy as proposed in [9]. Furthermore, the three instances of each control signal are physically dispersed to maintain the tolerance against burst errors. The proposed link structure with spare wires is presented in Figure 2 and the one with split transmission properties in Figure 4.

3. Realizations

The links were designed using the *Haste* design language and *Timeless Design Environment (TiDE)* toolset for asynchronous design by Handshake Solutions [5]. *Haste* has support only for 4-phase bundled data signaling so converter circuits were created to transform signaling to 2-phase and vice versa. The structures of the converters are explained in [7].

To find out the area overhead and the performance penalties the introduced fault tolerance causes, also a link without any fault tolerance and a design with ARQ but no reconfiguration circuitry were realized. The buffering capacity of these reference designs was set to the same as in the reconfiguration cases.

3.1 Link with Spare Wires

The structure of the link with spare wires is presented in Figure 2. The system is pipelined to increase the throughput. The transmitter contains first a latch stage (*L1*), which is followed by the Hamming encoder circuitry (*H*). After the encoding the data with check bits is stored in one of the two parallel output latches (*L2a* and *L2b*). These latches are used so that while one is connected to the output channel the next data word can be stored to the other. When *ack* is received to indicate a correct transmission, the output latch can be rapidly changed and a new transmission can begin. In the case of negative acknowledgement (*nack*) the same latch stays connected to the output channel and a retransmission is carried out (see also Figure 1(a)).

In the receiver the error syndrome for incoming data is calculated and the data word is stored to a latch (*L3*). If the syndrome equals zero, *ack* is sent and the data from the latch is forwarded to the receiver output. In the case of a nonzero syndrome, *nack* is sent and the syndrome is passed to the reconfiguration control unit, where it is stored into a 3-place ring buffer, so that the last three nonzero syndromes are found in this buffer. The structure and operation of the reconfiguration control unit is explained in Section 3.3.

When the receiver gets a request together with the reconfiguration vector from the reconfiguration control unit, it switches to the reconfiguration mode. An arbiter is used to guarantee that the mode change cannot occur in the middle of a receive operation. The reconfiguration data exchange between the receiver and the transmitter is illustrated in

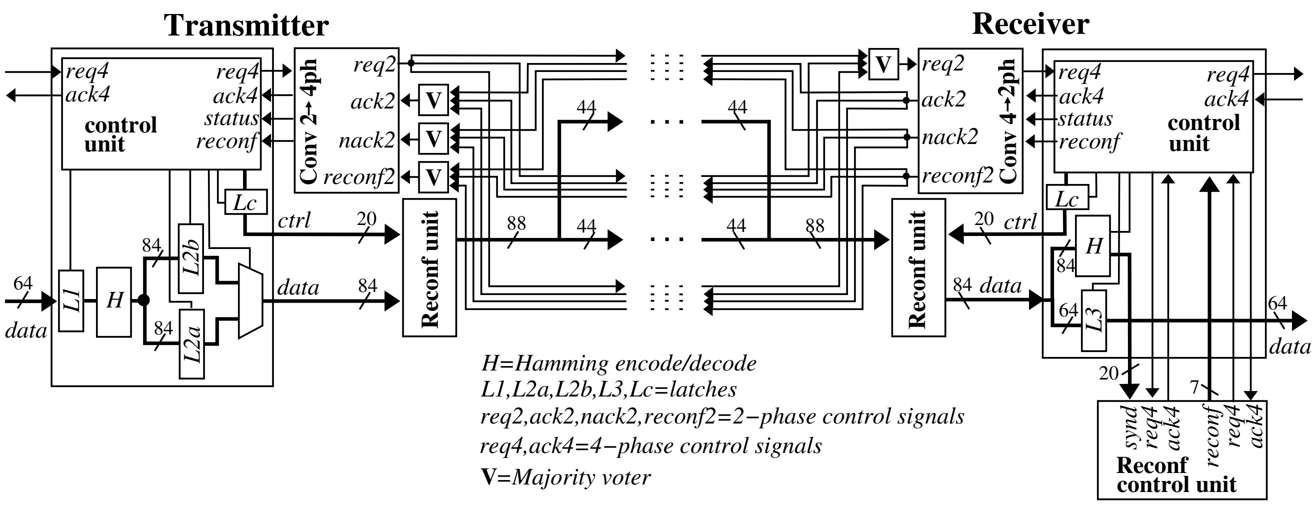


Figure 2. Structure of the link with spare wires.

Figure 3. The receiver sends *reconf* to indicate the mode change to the transmitter. The transmitter acknowledges this through the *req* line. Next, the reconfiguration information (reconfiguration vector, 7 bits) is transferred bitwise using *ack* and *nack* lines and every bit is acknowledged with the *req* line (see Figure 1(b)). After the acknowledgement of the last bit, the receiver sends *reconf* to indicate the mode change back to normal and the transmitter acknowledges it with *req*. Finally, receiver sends *nack*, so that the transmitter sends again the data it was sending when the mode change took place.

After the transmission of the control word both the receiver and transmitter store the error location into the control register of the correct interleaving section according to the reconfiguration vector. The structure of the reconfiguration unit is explained in [7].

3.2 Link with Split Transmission

The structure of the link with split transmission is presented in Figures 4 and 5. The system has many similar

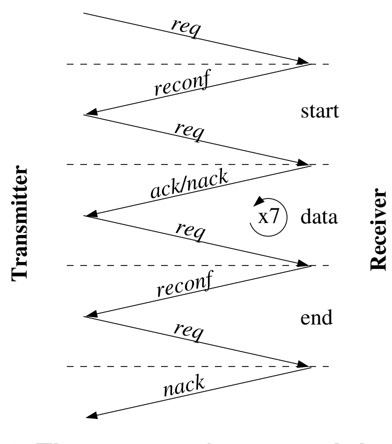


Figure 3. The protocol for transmitting the reconfiguration information.

parts to the one with spare wires. The transmitter differs after the multiplexer *M1*. In the normal mode the output of *M1* is connected to the output of the transmitter via the multiplexer *M3*. In the split mode, indicating that there is a faulty wire in the link, the output of the transmitter is obtained from the multiplexer *M2* (via *M3*). In the Hamming encoder circuitry the data is split into four interleaving sections (every fourth wire per section) and the five checkbits are calculated for each section separately and interleaved. In selector *Sel1* either first two or last two sections are chosen and doubled. Splitting the data according to the interleaving sections removes the need of two separate encoding and decoding units. The multiplexer *M2* chooses between the first and second split transmission parts. The part is changed when *ack* is received, in case of *nack* it stays the same and a retransmission is carried out.

In the receiver the error syndrome for incoming data is calculated and the data word is stored into a latch (*L3*) similarly as in the structure with spare wires. The data and check bits are in different parts of the data word depending on the operation mode. This selection is not shown in Figure 4. If the transmission is correct, *ack* is sent to the transmitter and the data from the latch is forwarded to the receiver output via the multiplexer *M5* in case of the normal mode and to the selector *Sel2* in the split mode. The correctness of the transmission is indicated in the normal mode when the syndrome equals zero and in the split mode when either copy of an interleaving section gives a zero-syndrome for both transmitted sections. This means that if the transmitted interleaving sections are marked as 1 and 2 and doubling creates instances a and b, the transmission is correct if one of the following combinations have zero-syndromes: $1a$ and $2a$, $1a$ and $2b$, $1b$ and $2a$, or $1b$ and $2b$. In the case of an incorrect transmission, *nack* is sent and the syndrome is passed to the reconfiguration control unit, which is done also during correct transmissions in the split mode. Based

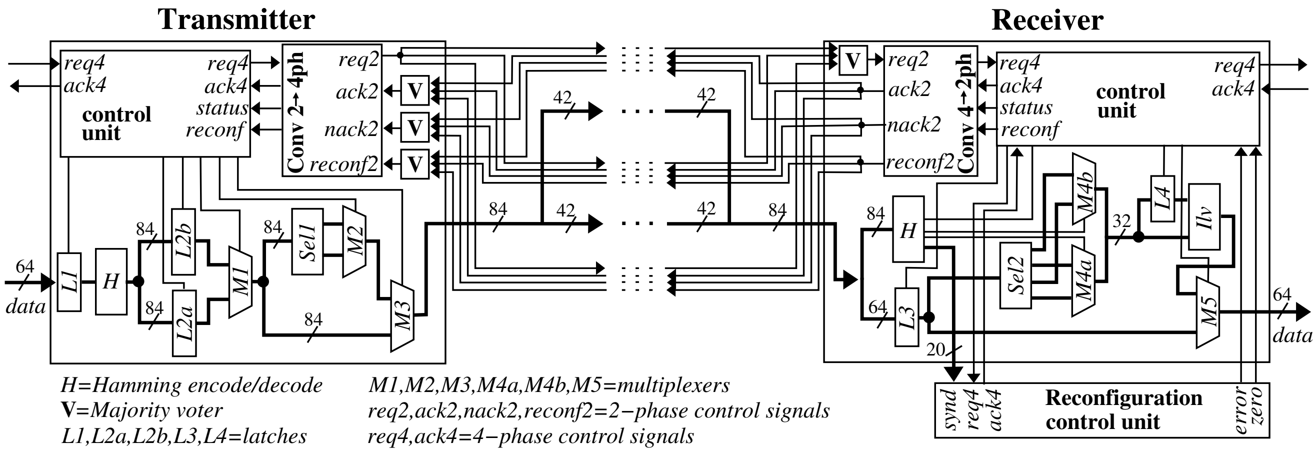


Figure 4. Structure of the link with split transmission.

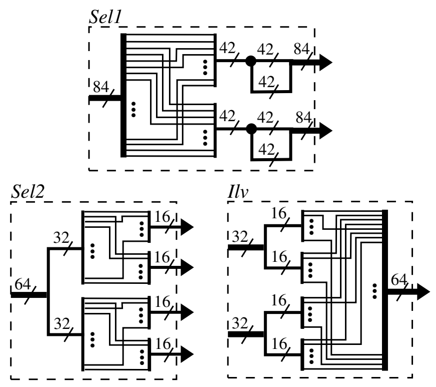


Figure 5. Select and Interleaver units.

on the syndromes the reconfiguration control unit detects permanent errors and controls the return to the normal mode if the errors have vanished (e.g. they were intermittent). The structure and operation of the reconfiguration control unit is explained in Section 3.3.

In the split mode, the selector *Sel2* separates the interleaved sections and using the multiplexers *M4a* and *M4b* the correct instances of both transmitted sections are chosen. The control for these multiplexers is obtained from the syndromes calculated by the decoding unit. The first split transmission part is stored to the latch *L4* and when the second part arrives they are interleaved (*Ilv*) and passed to the output via *M5*.

If the receiver is in the normal mode and an error is detected (signal *error* from the reconfiguration control unit) it switches to the split mode. The mode change takes place in the same way in the split mode if the signal *zero* is detected. An arbiter is used to guarantee that the mode change cannot occur in the middle of a receive operation or before a split transmission is completed. The receiver sends *reconf* to indicate the mode change to the transmitter. The transmitter acknowledges this through the *req* line, after which the receiver sends *nack*, in order to get the transmitter to resend the data it was sending when the mode change took place.

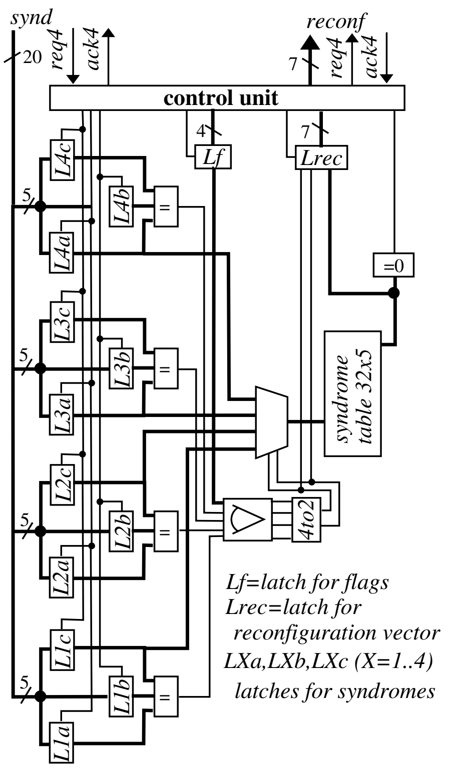
3.3 Reconfiguration control

The reconfiguration control unit for the spare wire design is depicted in Figure 6(a) and the unit for the split transmission design in Figure 6(b). Both reconfiguration units have a 3-place syndrome buffer which is divided into four 5-bit segments, one segment for each interleaving section (*LXa*, *LXb* and *LXc*, where *X* is the number of the segment).

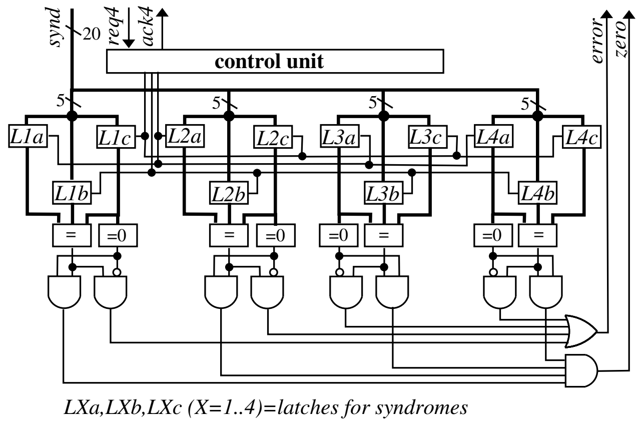
In the reconfiguration control unit for the spare wire design, the values of the three syndromes are compared individually in all the four segments and if they are equal and nonzero, a signal indicating a permanent error in that interleaving section is asserted. Based on these signals the reconfiguration process is commenced. Since there is only one spare wire for each interleaving section, a reconfiguration status flag (latch *Lf*) indicates if the spare has already been used and therefore the reconfiguration cannot take place. Arbitration between the error signals is needed to handle the situation, where a permanent error is detected in many sections at the same time (e.g. a burst error).

The reconfiguration procedure creates a reconfiguration control vector (latch *Lrec*) to be forwarded to the receiver. This contains 2-bit information indicating the correct interleaving section and 5 bits for pointing out the location of the corrupted wire. The location is fetched from a ROM memory according to the syndrome. In this memory there is a place for each of the 32 combinations of the 5-bit syndrome. Only 21 of these indicate a single error. The others are set to zero and the circuit checks if the fetched location equals zero. In this case no reconfiguration is done, because the exact error location cannot be solved.

The structure of the reconfiguration control unit for the split transmission design is simpler than the one explained above. An error is indicated via the signal *error* if in one of the segments the three syndromes are equal and nonzero. The signal *zero* on the other hand is asserted if all the syndromes in all the segments are zero.



(a) Spare wire design.



(b) Split transmission design.

Figure 6. Reconfiguration control units.

4. Results and Discussion

The designs were mapped to a 130 nm technology, synthesized and simulated using specific VHDL testbenches to find out the best and worst case throughputs and latencies. The interconnect delay was set to 500 ps, which was estimated to model the delay between two routers in a NoC structure including the drivers and possible repeaters. The same wire model was used in all simulations.

The throughputs, latencies and circuit areas are presented in Table 1, where *no FT* means the reference circuit without any fault tolerance structure, *ARQ* the one with ARQ but no reconfiguration properties, *spare* the presented structure with spare wires, and *split* is the structure with the split transmission properties. Latency (full) means the situation, where the buffer capacity of the link is totally occupied when the transmission begins and correspondingly latency (empty) means that there is no data in the buffers. The values listed in Table 1 for the split transmission structure are measured in the normal operation mode. The split transmission latency (full) in the split mode was measured to be 59.34 ns and the throughput 61.2 Mword/s. The reconfiguration procedure in spare wire design was measured to take in total 29.7 ns.

Table 1. Comparison of the link designs, with wire delay of 500 ps.

Design	no FT	ARQ	spare	split
Latency (empty) [ns]	2.67	7.71	8.48	9.65
Latency (full) [ns]	4.93	14.98	17.82	20.31
Throughput [MWord/s]	492.6	204.5	180.5	152.7
Area [$(\mu m)^2$]	6675	12386	25377	21602
- transmitter	4975	8348	12986	10518
- receiver	1700	4038	12391	11083

When comparing the designs with the reconfiguration properties to the ARQ design without them, it can be noticed that there is a performance penalty. In the spare wire design, the latency increases 10% / 19% (empty / full) and the throughput decreases 12%, while in the split transmission design the latency increase is 25% / 36% and the throughput decrease 25%. Thus, the spare wire approach has a smaller impact on the performance. On the other hand, the area overhead is larger in the spare wire approach. It uses 105% more area than the ARQ design without reconfiguration. The corresponding value for the split transmission is 75%. The area increase is mainly due to the reconfiguration control unit in the receiver.

The area overhead should be considered in the NoC context. Let us consider a simple mesh structure, where one router is assigned for each resource and the size of a resource is two times two millimeters [3]. There are three bididerctional links per each resource, thus, in total six transmitter/receiver pairs per resource. The area overhead caused by the introduced fault tolerance is only 2.2%-2.8% compared to the situation without any fault tolerance. The router area was not included in this calculation but it further decreases the relative area overhead.

The simulations were carried out using a 130 nm technology. As the dimensions have been scaled below 100 nm the delay of wires has not decreased as is the case with the delay of logic. The gap between the wire and logic delays is expected to increase as scaling deeper into nano regime [4]. Based on this scenario and because the bottleneck at the moment are the receiver and transmitter units, the performance of the presented link structures should increase as the dimensions are scaled down.

The main parts were created using a modelling language and synthesis tools which use a rather conservative approach to the problem. Based on our previous experience we predict that the performance could be significantly enhanced by careful manual design although still using standard gate libraries [8].

The tolerance against single permanent faults in the split transmission design is limited by the fixed location of the doubled data parts. If the location of the permanent fault was more exactly isolated, the placement of the doubled data could be controlled and a greatly improved tolerance against single faults could be achieved. The error location could be determined using the syndromes as is done in the spare wire design. This would probably increase the control logic and would also result in performance degradation. The presented approach has the assumption that if there are multiple errors, they most likely occur as bursts, which can be motivated by considering different manufacture defects.

The split transmission design has a property to return to the normal mode if the fault was intermittent. A similar property would be advantageous also to the spare wire design, where the reconfiguration at the moment is irreversible. The circuit is not able to tolerate permanent errors if there have been intermittent errors in the same interleaving section (but at a different wire) although the intermittent fault has vanished. However, the intermittent faults in practice repeat themselves in the same locations due to e.g. small anomalies in the manufactured chip, which turn into faults under certain environmental conditions.The advanced detection of different fault scenarios will be a part of our future research.

The number of spare wires in the presented design was set to four and they were assigned one for each interleaving section. The fault tolerance properties could be enhanced by increasing the number of spare wires. It would also be advantageous to have the spare wires assigned more flexibly to different interleaving sections. One interleaving section may have many erroneous wires while some other has none. The drawback of the increased flexibility is the growth of complexity which probably leads to a performance decrease and area overhead.

The analysis of the fault tolerance properties of the different structures can be done running them on a sophisticated error patterns. This analysis is presented in [7] as well as the impact to the power consumption.

5. Conclusion

We proposed link structures that have properties for tolerating efficiently transient, intermittent and permanent errors. The protection against transient errors was realized using Hamming coding and interleaving for error detection and ARQ as the recovery method. Two approaches were introduced to tackle the intermittent and permanent errors. Split transmission was an approach utilizing time redundancy while the other structure, which introduced spare wires, was a hardware redundancy approach. Communication in the links was based on asynchronous 2-phase signaling and the control signals for ARQ and reconfiguration were incorporated into these control signals. The control lines were protected using triple modular redundancy.

We presented designs for the different components of the links and proposed the needed reconfiguration control. The designs were implemented, simulated and compared against reference designs. The simulation results show that the performance decrease when comparing to a design with ARQ but no reconfiguration is larger for the split transmission design (latency 31%/throughput 25%) than for the spare wire design (15%/10%) while the area overhead is larger for the spare wire design (105%) as for the split transmission design (75%).

Our research shows that combining different fault tolerance methods a structure capable of tolerating all different types of errors is achievable. As is always the case with fault tolerance, this does not come for free. There is a clear area overhead, but on the other hand when comparing to the IP block areas in the NoC context, the overhead is only a couple of percents.

References

[1] D. Bertozzi, L. Benini and G. De Micheli, “Error Control Schemes for On-Chip Communication Links: The Energy-Reliability Tradeoff,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, June 2005, Vol. 24, No. 6, pp. 818-831.

[2] C. Constantinescu, “Trends and challenges in VLSI circuit reliability,” *IEEE Micro*, Vol. 23, Issue 4, July-Aug. 2003, pp. 14-19.

[3] W.J. Dally and B. Towles, “Route packets, not wires: on-chip interconnection networks”, *Proc. Design Automation Conference*, 2001, pp. 684-689.

[4] G. De Micheli and L. Benini, *Networks on Chips,* Morgan Kaufmann Publishers, 2006.

[5] Handshake Solutions, *http://www.handshakesolutions.com.*

[6] B. W. Johnson, *Design and Analysis of Fault-Tolerant Digital Systems,* Addison-Wesley, 1989.

[7] T. Lehtonen, P. Liljeberg and J. Plosila “On-line Reconfigurable Self-Timed Links for Fault Tolerant NoC.” *VLSI Design*, Journal of Hindawi Publishing Corporation, Apr. 2007.

[8] P. Liljeberg, J. Plosila and J. Isoaho, “Self-timed communication platform for implementing high-performance systems-on-chip,” *Integration, the VLSI Journal*, Vol. 38, Issue 1, Oct. 2004, pp. 43-67.

[9] S. Murali, et al., “Analysis of Error Recovery Schemes for Networks on Chips,” *IEEE Design & Test of Computers,* Sept.-Oct. 2005, Vol. 22, Issue 5, pp. 434-442.

[10] J. Sparsø, S. Furber, *Priciples of Asynchronous Circuit Design - A System Perspective,* Kluwer Academic Publishers, 2001.

[11] H. Zimmer and A. Jantch, “A Fault Model Notation and Error-Control Scheme for Switch-to-Switch Buses in a Network-on-Chip,” *IEEE/ACM/IFIP Int. Conference on Hardware/Software Codesign and System Synthesis,* Oct. 2003, pp. 188-193.