2009 15th IEEE Symposium on Asynchronous Circuits and Systems

Fault Tolerant Delay Insensitive Inter-Chip Communication

Yebin Shi, Steve B. Furber, Jim Garside and Luis A. Plana
School of Computer Science, University of Manchester
Oxford Road, Manchester, M13 9PL, UK
e-mail: shiy@cs.man.ac.uk, steve.furber@manchester.ac.uk, {jdg,lplana}@cs.man.ac.uk

Abstract**—Asynchronous interconnect is a promising technology for communication systems. Delay Insensitive (DI) interconnect eliminates relative timing assumptions, offering a robust and flexible approach to on- and inter-chip communication. In the SpiNNaker system - a massively parallel computation platform -a DI system-wide communication infrastructure is employedwhich uses a 4-phase 3-of-6 code for on-chip communication and a 2-phase 2-of-7 code for inter-chip communication. Fault-tolerance has been evaluated by randomly injecting transient glitches into the off-chip wires. Fault simulation reveals that deadlock may occur in either the transmitter or the receiver as handshake protocols are disrupted. Various methods have been tested for reducing or eliminating deadlock, including a novel phase-insensitive 2-phase to 4-phase converter, a priority arbiter for reliable code conversion and a scheme that allows independent resetting of the transmitter and receiver to clear deadlocks. Simulation results confirm that these methods enhance the fault tolerance of the DI communication link, in particular making it significantly more resistant to deadlock.**

I. Introduction

As technology shrinks, more IP cores are integrated onto a single chip to implement more complicated and efficient on-chip system solutions. To save wiring resource and power consumption and to increase communication performance, extensive research has been conducted into network-on-chip (NoC) systems [1][2]. The NoC approach particularly suits communication-dominant on-chip systems. Asynchronous NoCs have been proposed to eliminate the clock for global communication [3][4], providing better power efficiency and higher modularity compared to synchronous NoCs.

The robustness of asynchronous circuits to transient glitches or permanent faults is a growing challenge in the face of reducing feature sizes and increasing on- and off-chip communication speeds. Moreover, factors such as alpha particles, cosmic radiation, cross-talk and power bounce can cause soft errors in on- or inter-chip interconnects.

The behaviour of inter-chip link interfaces in the presence of transient faults is of particular interest in their role of communication in multi-chip systems and the vulnerability of asynchronous protocols to glitches. Moreover, inter-chip interfaces may incorporate complex conversion circuits between the different on-and inter-chip asynchronous protocols and delay-insensitive codes. The asynchronous interface circuits are smaller but more complex than the other components that form the asynchronous communication fabric, which generally has a regular architecture and a larger scale. The inter-chip wires are also more vulnerable as they reside in a noisy environment.

Previous work [5] investigated the impact of glitches on the inter-chip links. The common symptoms resulting from the glitches are missing and superfluous symbols causing corrupt packets and, in some cases, deadlock in different parts of the circuit. Though the causes of deadlock vary, they always result from failures in the handshake protocols. Some techniques were developed to increase the robustness of asynchronous interconnects to transient glitches.

This paper proposes various techniques, based on the original design of the interface circuits for the inter-chip links, which increase the resilience of the delay insensitive circuit to transient glitches. These techniques are designed to avoid deadlock and to minimize the impact of errors on the inter-chip links. The paper is organized as follows: after a brief introduction to the SpiNNaker system, the inter-chip link interface is described. Section IV shows examples of the error-tolerant circuits used and describes some of their evolution. Sections V and VI describe the simulation performed on two variants of the interfaces: the first is a 'robust' design from earlier work, the second is described and evaluated here for the first time. Section VII concludes the paper.

II. The SpiNNaker System

SpiNNaker [7] is a massively-parallel computation platform based on chip multi-processors and a packet-switched communications fabric, aimed at modelling large-scale systems of spiking neurons in real time. Each chip in this system integrates twenty microprocessor cores, various system-support peripherals and a communication infrastructure and is connected with other chips through six duplex links, as shown in Fig 1. The system employs three primary communication infrastructures: the AMBA AHB/AXI bus to connect each processor to its associated system blocks, the self-timed System NoC - built using the CHAINworks tool [3] - to connect each microprocessor subsystem to a shared off-chip SDRAM, and the

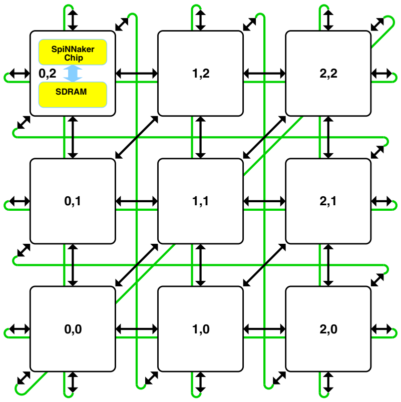

Fig 1 Mesh topology of the SpiNNaker system

1522-8681/09 $25.00 © 2009 IEEE
DOI 10.1109/ASYNC.2009.21

Communications NoC - a delay insensitive on-chip and inter-chip communication network that connects the synchronous microprocessor islands and is used primarily to carry neuron spike events. The last two communication infrastructures are both based on packet-switched networks.

This paper focuses on the interface circuit that bridges between on-chip and off-chip interconnects in the Communications NoC. This is a serial network transmitting short packets around and between chips. In SpiNNaker packets are short and come in two sizes, 5 and 9 bytes which are typically represented as 10 and 18 4-bit flits, respectively. As packet size varies, the packet length is indicated with an end-of-packet (EoP) marker.

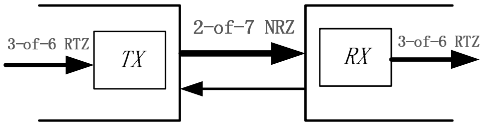

Fig 2 an off-chip link

The communication infrastructure comprises two parts: on-chip 4-phase or Return-To-Zero (RTZ) asynchronous interconnects and off-chip 2-phase or Non-Return-to-Zero (NRZ) interconnects as shown in Fig 2. To increase the power efficiency and throughput of the communication links, the off-chip interconnect employs the 2-phase asynchronous protocol as this reduces the number of transitions, saving half of the power consumed by the RTZ protocol and incurring only half the round-trip cycle delays per symbol. However, due to the higher complexity of pipelines based on the NRZ protocol, the on-chip interconnect uses the 4-phase protocol. Moreover, a 3-of-6 encoding is used for on-chip interconnects and a 2-of-7 code for off-chip interconnects. The two codes are both capable of transferring 4-bits of data per symbol, with the 3-of-6 code needing one fewer wire and the 2-of-7 code one fewer transition per symbol.

Table 1 2-of-7 and 3-of-6 flit coding

Decimal	2-of-7 code	3-of-6 code
0	001_0001	11_0001
1	001_0010	10_0011
2	001_0100	10_0101
3	001_1000	10_1001
4	010_0001	01_0011
5	010_0010	11_0010
6	010_0100	10_0110
7	010_1000	10_1010
8	100_0001	01_0101
9	100_0010	01_0110
10	100_0100	11_0100
11	100_1000	10_1100
12	000_0011	01_1001
13	000_0110	01_1010
14	000_1100	01_1100
15	000_1001	11_1000
EoP	110_0000	N/A

SpiNNaker employs two DI coding schemes, one 2-of-7 and the other 3-of-6 [6]. These are incompletely used: 2-of-7 coding can support 21 symbols, of which 17 are used here; 3-of-6 supports 20 codes, of which 16 are used. The extra symbol in the 2-of-7 code signifies end-of-packet (EoP) for the inter-chip communication; a dedicated channel is used to carry EoP symbols over the 3-of-6 on-chip links.

Table 1 lists the data symbols used. To design efficient completion detection (CD) circuits both codes can be grouped into smaller M-of-N codes with fewer wires, which leads to CD circuits with lower area and delay [6]. Their implementation details are ignored here, however it should be noted that CD circuits supporting incomplete DI code sets - while leading to efficient CD module designs - may be unable to detect invalid symbols which are not defined in the code set.

III. INTERFACE CIRCUITS

This section gives an architectural view of the inter-chip link interfaces. Some detailed circuit examples are developed in the subsequent section.

A. *Transmitter*

The function of the transmitter is to take 4-bit flits from a CHAIN link and output them on the off-chip connection. The input link comprises six wires conveying data as a 3-of-6 code and a separate, parallel 1-of-3 code which specifies the flit type, together with their acknowledge signals. Only two flit types are implemented here, namely 'normal' and 'end-of-packet' (EoP). The output has a single channel so seventeen symbols are carried, sixteen data symbols plus an EoP marker. The transmitter therefore needs to insert an extra symbol to mark EoP when the input flit has been tagged accordingly. This reduces the available external bandwidth slightly but reduces the pin count significantly, from 11 (10 used) wires from the CHAIN link to 8 on the chip boundary. As there are six duplex links this saves 36 pins on each chip.

The translation is pipelined (Fig 4): in the first stage latches hold the input values for data and control independently. In the subsequent stage the code translation is performed; normally this stage discards the control information but if the control indicates EoP then it is cycled again to generate the output EoP flit before its input is acknowledged. Up to this point the symbols are held as RTZ codes in simple C-element latches (Fig 3).

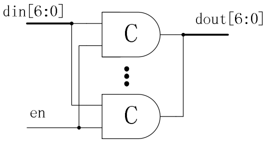

Fig 3 a simple C-element 1/2 latch

The final stage of the transmitter performs the two-phase conversion and outputs to the pins. This stage is kept as simple as possible because the off- and on-chip delays limit the interconnection bandwidth so circuit overheads need to be minimised.

The transmitter does not need many fault tolerant features as it is assumed that the majority of problems occur off chip.

The only signal considered to glitch is therefore the off-chip acknowledge ('ack2'). This is protected by its phase converter which expects a single acknowledge transition but is able to ignore any additional transitions until the next flit is sent. A glitch on this line may therefore result in the next flit being sent prematurely - with the consequence that it may not be received correctly - but will not deadlock the link providing this is tolerated by the receiver. As the transmitter has no way of discriminating a glitch transition from a genuine acknowledgment, this is unavoidable.

When the transmitter is reset it assumes that it has had all preceding flits acknowledged. Therefore it is able to transmit as soon as it receives an input flit. If this assumption is wrong and a spurious acknowledgment arrives it is simply ignored. In order to do this it is important that no assumption is made about the direction of the acknowledge transition; the phase converter solves this problem by converting arbitrary changes into whatever level is required internally. This facility means that the chip containing the transmitter can be reset (for whatever reason) without resetting the receiver chip. As all the chips are interconnected by the same asynchronous network this is an important consideration in the fault recovery of parts of the system in the event of, for example, a watchdog reset.

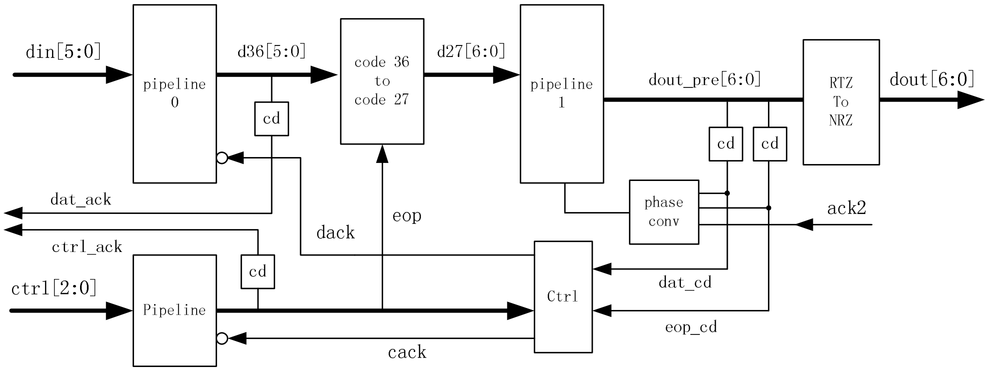

Fig 4 Transmitter block diagram

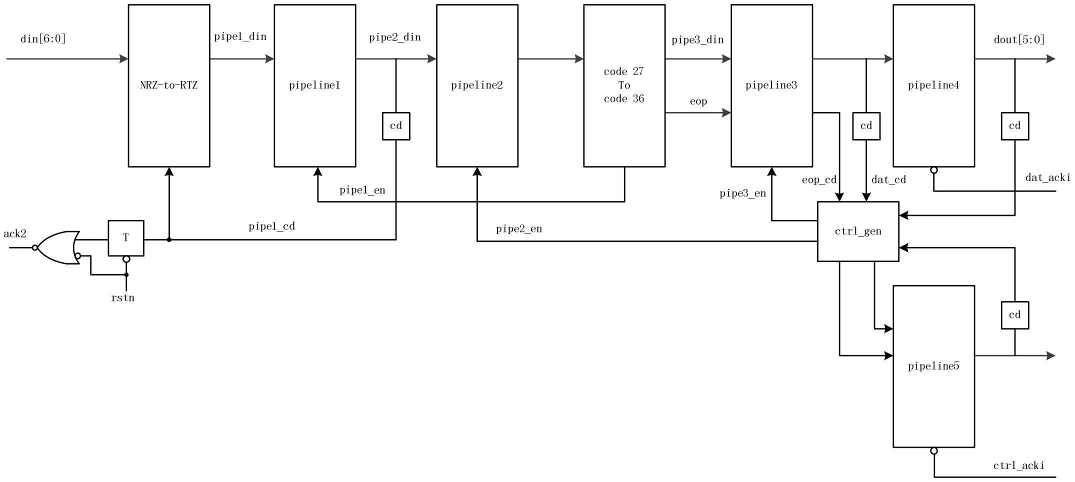

Fig 5 Receiver block diagram

A. Receiver

The receiver reverses the transmission process, converting the 2-of-7 NRZ symbols into 3-of-6 RTZ flits and stripping the 2-of-7 EoP symbols, replacing them with a parallel flit type-identifier for the downstream CHAIN link. Anticipating errors, extra transitions in the input symbols can be absorbed without causing deadlock.

Like the transmitter, the receiver (Fig 5) is pipelined to increase potential throughput. The first stage receives a symbol and, when it is complete, returns an acknowledge transition to the sender. In this stage a symbol is complete when transitions have occurred on at least two different wires. In the absence of glitches there should be exactly one transition on each of two wires.

The second pipeline latch provides some extra buffer space. In the implementation the latches are simple C-element latches (Fig 3) and thus hold either a data element or a null 'spacer'. The added capacity allows the receiver to hold two input flits simultaneously without impeding the operation of the time-critical external link. This is necessary for the protocol conversion back to the CHAIN link; before a flit can be dispatched it must be determined if it is a 'normal' or an 'EoP' flit; a flit is 'normal' if the following 2-of-7 input does not indicate EoP so the subsequent input is needed for the majority of outputs. The first flit of a packet is sent after the second is received and so forth with the last data being freed by the input EoP symbol.

The third pipeline stage converts the 2-of-7 input into a legal 3-of-6 output. For error tolerance the input is, in practice, a 2-or-more-of-7 code and this must be coerced into a code which will not deadlock the later flow. For simplicity the 2-of-7 code is first translated into a 'one hot' code by examining all possible legal pairs of inputs. Because the error model only adds transitions the receiver can rely on a legal combination being present before trying to process a symbol; however it is quite possible to have more than one legal pair of input bits in this stage. These are filtered using a priority encoder (Fig 10); the chosen 'priority' is arbitrary but ensures that only one code is chosen. The single, chosen 1-of-17 code can then be ORed into its appropriate 3-of-6 data symbol or signal EoP.

The last stage of the pipeline has already been alluded to. This retains a data symbol until pushed out by the subsequent symbol which is used to determine its type. Because EoP accompanies data on the output side the receiver does not stall between packets.

A final protective feature of the receiver (Fig 5) is that it produces a single acknowledge transition when its local reset is removed; this is injected after the acknowledge 4- to 2-phase conversion toggle flip-flop. This means that, if the receiver is reset during reception of a flit it will still appear to acknowledge it when reset is removed. This prevents the off-chip link deadlocking in the state where both the transmitter and receiver are each waiting for a signal from the other. If both sides of the link are reset - for example on power-on – then both transmitter and receiver start off 'ready'; the receiver immediately acknowledges an imagined input and the transmitter receives an 'erroneous' acknowledgment - which it is designed to ignore safely.

In operation it is conceivable that a transient fault may cause a single chip to crash; a watchdog recovery mechanism is in place to reset the faulty device which will probably disrupt the inter-chip handshaking protocol. However the link can recover from this in the same way. The glitch resistance therefore also aids significantly in recovering from larger system faults.

In the SpiNNaker system no packet can be more than 18 flits long; introducing a longer packet could cause problems in the on-chip network. By accepting glitches as potentially legal symbols extra flits may be introduced, producing a packet which could not be accommodated in a later buffer. To prevent this the receiver output is sent through a flit counter whose job is to insert an EoP marker if one has not been received within the last 18 flits. This may 'corrupt' a packet - actually it has already been damaged – but ensures that the input stream cannot deadlock units further downstream.

IV. Circuit Descriptions

The fault-tolerant circuits that make up the inter-chip interfaces have undergone considerable evolution as they were developed. This section gives a brief description of the most developed circuits, in some cases with some earlier incarnations used to illustrate particular problems.

A. Phase conversion

Phase conversion is performed in both interfaces because both have input and output wires. The phase converters are kept as close as possible to the chip boundary because handling 2-phase signals in CMOS logic is typically quite onerous. The major concerns are with the 2-to-4 phase conversion as this is the incoming interface and the one assumed to be prone to errors.

Fig 6 shows a simple phase converter for the transmitter. When a valid symbol is input the latch is closed via the toggle element and it remains so until an external acknowledge is received. The data phase conversion can be provided by a set of seven toggle flip-flops, which change as data bits arrive. This unit only expects valid input symbols so this suffices. The biggest potential problem with this circuit is that it relies on alternating input and acknowledge (ack2) stimuli, which cannot be guaranteed if the input acknowledge glitches, nor if the receiver is reset at some random time.

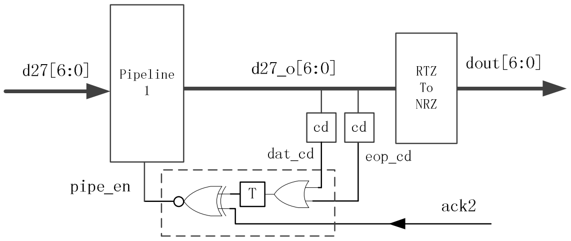

Fig 6 Transmitter phase converter

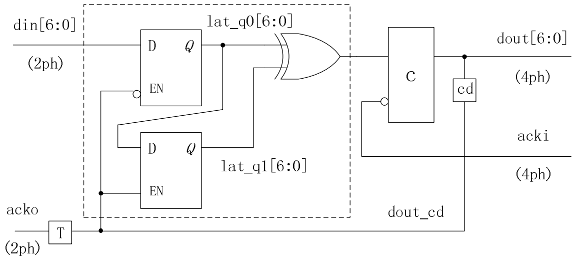

Fig 7 Receiver phase converter

Fig 7 shows an initial design for the receiver's phase converter. This comprised a master-slave latch in which the master is initially transparent to incoming transitions. In this design, when sufficient transitions have arrived the completion detector (cd) acknowledges the flit, closing the master latch and opening the slave latch, thus 'cancelling' the input via the exclusive-OR gate.

In a 'clean' environment this works satisfactorily; it will also tolerate the majority of glitches which will briefly alter the level of an input wire. However there can be a problem if an input glitch occurs coincidentally with the completion detector's firing.

Figure 7 illustrates a deadlock scenario; only three of the input wires are shown - the others are assumed to remain '0' and the buses are annotated with binary patterns, not the symbols' values. The input starts as '0' and two symbols, '6' and '5' are transmitted as transitions. The first symbol is successfully recognized by the subsequent C-gate pipeline stage. Unfortunately a glitch occurs on data wire din[0] at about the same time as the acknowledge signal acki arrives. The phase converter now holds the false pattern value '7' where three transitions have been captured. When the second symbol ('5') enters the converter it is interpreted as '4' by the XOR gate because the previous data held in the converter is '7' rather than '6'. The converter deadlocks because '4' has only a single transition so no Acknowledge token is issued.

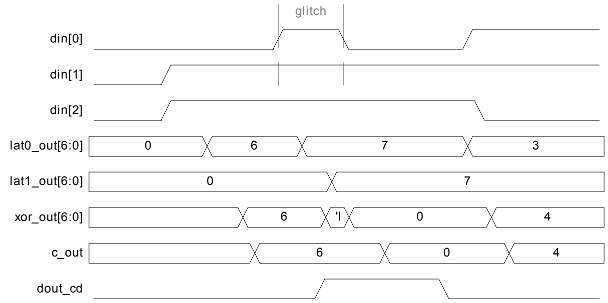

Fig 8 Deadlock timing in the receiver

The problem occurs because half a glitch is incorporated (as an error) in the first flit but the other half glitch appears in the subsequent flit. This does not deadlock if the next data transitions are on different wires, only if the glitched wire carries a valid transition in the next flit and this arrives first. The only recovery from this would be to await another glitch on a third data wire; it does not seem sensible to rely on this! Among others, this problem was highlighted by the intensive simulation, described later.

To combat this problem a different phase converter was designed which is purely sensitive to transitions (Fig 9). This comprises a parallel pair of RS flip-flops, with dominant reset inputs, whose outputs are logically ANDed. In operation the flip-flops are forced reset by an acknowledge pulse which is then released. At this time one of the flip-flops is set by the data input; the other remains reset. Any change on the data input will set the second flip-flop and the output will then go to, and remain, set until the next acknowledge pulse. The circuit is therefore truly phase insensitive responding only to the first input change.

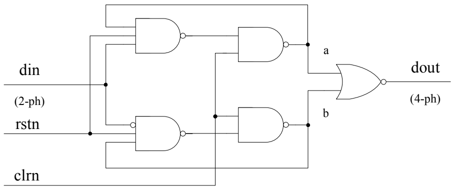

Fig 9 novel phase converter

In operation glitch edges can arrive at any time, including during the reset pulse. In this latter case the edges will be ignored. However data transitions always occur between reset pulses and are always a single level change, so they could, conceivably, be delayed by an ill-timed glitch but cannot be removed. Data transitions can never be lost, thus preventing deadlock from this cause, and the phase converter will filter out some of the spurious glitch transitions.

This style of phase converter can be used both at the receiver and for the acknowledgment entering the transmitter. It should not deadlock providing that its clear pulse has completed before the next genuine data transition occurs. This can be guaranteed by sending the acknowledgment (or data) on the falling edge of this pulse. In the first realisation this safety is compromised slightly - at least in theory - by sending the acknowledgment as the clear pulse is asserted. The clear pulse is self-timed locally whereas for an off-chip link the external pad delays are considerably larger and are incurred twice in the round-trip time. Pragmatically this is a very safe race condition and allowing an earlier output makes the interface cycle time – the slowest link in the chain - noticeably faster.

B. Symbol conversion

Transient glitches on the inter-chip wires may produce an illegal symbol in the receiver. Four of the possible 2-of-7 codes are not used and codes with more than two set bits are clearly illegal. In particular an invalid symbol must not be interpreted as both a data and EoP symbol. The output stream from this stage should consist of data packets, each accompanied by a type marker indicating they are 'normal' or 'EoP'. So, having obtained a 2-or-more-of-7 4-phase code this must be coerced into a legal symbol. This is done during the translation to a 3-of-6 code.

Fig 10 derived from [8] shows the key circuit in this stage. Inputs are fed to C-elements in pairs with the expectation that the first legal symbol to arrive will fire one element; this is akin to a DIMS circuit [9][10]. Assuming a correct input symbol this travels forwards to its corresponding output and is used to assert exactly three of the data output wires or the EoP indicator.

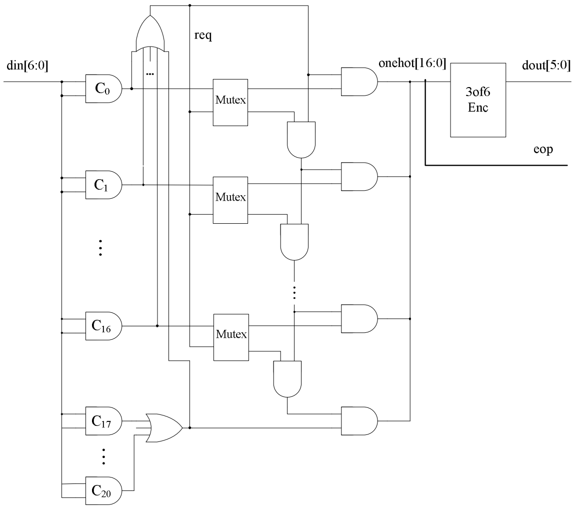

Fig 10 a priority arbiter

In the event of a corrupt input it is possible that two or more of the input C-elements may fire at about the same time. The first one to do so will begin the assertion of the req signal through the large OR gate - actually a tree structure owing to the considerable fan-in. The delay through this gate and its subsequent fan-out is exploited in the circuit operation in that the first DIMS term is assumed to reach its mutual exclusion element (mutex) before req has switched; it therefore wins that particular race. This input, by holding the mutex, blocks all the lower output paths in the daisy chain and waits to be output from the daisy chain when req is valid.

Later switching elements may also win through their respective mutex; however at some time later req prevents further changes. When req appears it begins to ripple down a chain, pausing when it finds an undecided mutex and quitting when it finds its first legal code signal. Although this is a serial search, and therefore quite slow, it is adequate to keep up with the off-chip cycle time. If higher performance was required it would be relatively simple to add a 'carry look ahead' style optimisation to this chain.

The output C-elements provide both an AND function on the forward cycle and prevent output glitches during the return-to-zero phase, where races from the inputs can otherwise - in principle - expose a 'lower priority' request briefly if the inputs are zeroed with significant skew.

C. *Flit counter*

In the presence of glitches, missed EoP symbols or additional false symbols in a packet can lead to framing errors. The communications NoC fabric may run out of buffer space if an over-length packet arrives and occupies all available buffer space, causing deadlock over the input links of the on-chip router. Packets in the SpiNNaker system have two valid lengths: 10 flits and 18 flits. The receiver needs to check the length of any received packets ensure an EoP is signalled at least every 18 flits and generate a framing error signal to tag the packet in cases where the EoP symbol is missed or superfluous data are received.

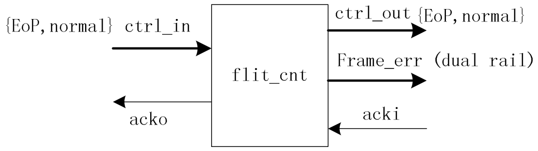

Fig 11 Flit counter

The flit counter (Fig 11) counts 'normal' input flits and is reset when an 'EoP' is received. If an eighteenth successive 'normal' input is received the output is changed to an EoP and the counter is reset. This results in splitting an input packet but that packet is already corrupt because extra flits must have been inserted. The 'truncated' packet can be marked as such to a downstream unit which could log and discard it. The counter does not have to be a high-performance unit it is counting at the flit rate. The chosen design (Fig 11) is based on van Berkel's handshake circuit modulo-N counter [11].

V. Fault Simulation

In the absence of a formal methodology to verify the effectiveness of a circuit's tolerance to transient glitches, extensive circuit simulation is essential. A fault simulation platform based on the injection of random glitches is necessary to observe the effects of errors on the asynchronous circuits in the communications NoC and to provide automatic detection of deadlock and verification of the results. The platform includes a sub-module to generate transient glitches with configurable duration and frequency, which are randomly injected onto the inter-chip wires (including the acknowledge wire). By specifying different glitch injection frequencies, a large number of random and densely-spaced glitches are injected onto the inter-chip wires during the transfer of one million packets. Whilst not a rigorous proof, extensive simulation based on this platform is sufficient to assess the impact of transient glitches on the asynchronous inter-chip links and to estimate effectiveness of glitch-resistant circuits. Such high density glitches are unlikely to occur in real circuits.

The (post-synthesis, back annotated) circuit has been simulated in the presence of a large number of high density random glitches (on average a glitch inserted roughly for each two packets) during the transfer of one million packets. Glitches are induced on both the forward data path and the acknowledge signal. As a result, invalid symbols - including incorrect end-of-packet (EoP) codes - can appear and acknowledge wires can transition unexpectedly, leading to link faults and, possibly, deadlocks as described previously. Flit insertion and corruption is unavoidable - glitches could contrive to emulate both a valid flit and its acknowledgment - and their correction can be left to a higher level protocol. The important issue for the asynchronous link is to avoid deadlock.

A mixture of 10- and 18-flit packets containing random data was transmitted. This size reflects the packets used within the SpiNNaker network although their contents are intended more as a test of the link than as operationally representative data. Their content was protected by a CRC and a packet was regarded as received successfully if its CRC was correct on reception. Whilst this is not a perfect test of 'successful' reception - occasional false positives are possible - it gives a good indication of the actual error rates.

In order to observe the robustness of the link, glitch duration was varied, as was the length of the external delays used to emulate the propagation time across a PCB. As there are some timing assumptions used in the design, it is a good idea to try to establish the limits of operation.

VI. Simulation Results

The first simulation results showed that most deadlocks occurred at the 2-phase to 4-phase converter in the receiver. This prompted the development of the phase converters described earlier in this paper.

The post synthesis fault simulation results in Table 2 confirm the effectiveness of the proposed interface circuits. With a similar density of injected glitches, the first generation circuits – using the conventional phase converter, code conversion and completion modules - have a moderate risk of deadlock. Through hardening the circuits with all the proposed techniques the new interface circuits exhibit a much better tolerance to transient glitches and, under anything approximating to expected operating conditions, are deadlock free.

Table 2 simulation results for the interface designs

Items \ Designs	Original I/F	Proposed I/F
Glitches	478,280	390,357
Successfully Received Packets	916,684	863,182
Deadlock	7,632	7
Performance (ns/symbol)	17	15
Area(um^2)	8219.7	8555.7

In some extreme circumstances simulation revealed an unpredicted deadlock model for the new circuit. This corresponds to a glitch ‘removing’ a genuine data (or acknowledge) transition and not, subsequently providing another transition on that wire. The mechanism for this is as follows (Fig 12): a glitch begins within a data symbol and injects a spurious edge which is accommodated, possibly corrupting a flit but not causing deadlock; the 'glitch' persists for a considerable time, its second edge corresponding closely with the following, valid transition; the glitches later edge and the data edge occur close together - and in opposing directions – effectively cancelling each other out.

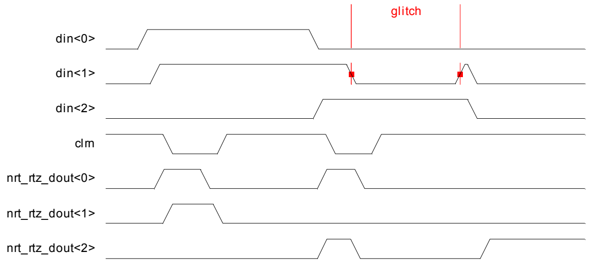

Fig 12 Timing diagram showing long-glitch fault

For this to occur a 'glitch' must be at least as long as the round-trip time on the data link, so that its leading edge can be filtered and its falling edge coincide with genuine activity. This must be a single glitch: two glitches on the same wire will expose transitions between them and, although these are erroneous, the link will continue to operate. As the anticipated cycle time is >10ns is seems unlikely that this failure mode will occur in reality; it was exposed only by simulations with artificially short delays and long glitch times.

Successful packet reception, as judged by a correct CRC, was monitored for the two designs. The earlier design had a notably higher number of packets received successfully than the one which avoids deadlock. This is probably due to its lower sensitivity to random edges; a transition which might corrupt a flit is readily accepted in the later design but it is prevented from causing a deadlock. The earlier design is less sensitive to corruption but will deadlock much more easily.

The reset mechanism was also verified by simulation by resetting both the transmitter and receiver at independent, random times. This did not caused corruption but not deadlock in the transmission link, as expected.

Flit throughput is important in system operation. The critical cycle time is the external link due to the on- and (especially) off-chip pad delays. These are minimised by using a 2-phase protocol but the circuit efficiency can also influence this noticeably. The new circuit cycles about 10% faster than its predecessor, some of which, however is attributable to 'conventional' circuit improvements.

Although the size of the interfaces is a small part of a SpiNNaker chip it is still desirable to keep overheads down as much as possible. Sample layouts have been constructed in a 130nm UMC process which indicates that the highly glitch resistant interface is about 4% larger than its predecessor. This is not a high price to pay for the considerable increase in reliability.

VII. Conclusions

This paper described some fault tolerant circuits intended to reduce the consequences of transient glitches on the asynchronous inter-chip interfaces in SpiNNaker. In particular the intent is to reduce the potential for deadlock; providing the link continues to run other error correction schemes may be used to verify data integrity.

By analyzing the reasons for deadlock and comparing different 2-phase to 4-phase conversion circuits, fault tolerant implementations of the inter-chip interfaces were devised. There are three major fault-trapping stages. The greatest benefit was conveyed by the phase-insensitive data and acknowledge converters; the second, code conversion, stage with priority arbitration ensured that only legal symbols can reach the on-chip network; finally a counter ensures that packet size is limited to within the size of later buffers.

The extensive simulation results show that the implementation has a very high resistance to deadlock and is able to not only keep running but successfully convey packets for a high proportion of the time despite artificially rather intense noise. Although the current circuit corrupts somewhat more packets than its predecessor this is due to glitches being translated into symbols rather than deadlocks. Deadlock avoidance is regarded as the most important consideration.

Under any circumstances reasonably close to the expected operating conditions it is believed that the links are now deadlock free. The only deadlocks that have been found were achieved by mixing artificially long 'glitches' with artificially short inter-chip delays. The probability of glitch durations approximating the link's cycle time (or greater) and then, coincidentally cancelling a real signal is regarded as exceedingly small. In such an event the link is still recoverably by a higher-level reset process.

The practical realisations of the circuits described here contain some delay assumptions. For example, the off-chip acknowledge is sent before the receiver's phase converter has completed clearing. This races the inter-chip cycle time with an on-chip pulse generation and, with the long, off-chip delays is very safe and speeds up the critical path of the slowest cycle in the network. It would be a simple matter to alter this to delay the acknowledge to the falling edge of the clear pulse if, for example, the 2-phase link had much lower latency, for example by being entirely on-chip. The 2-of-7 to 3-of-6 code converter also exploits a delay model; a delay insensitive circuit using the same principle could be produced but, owing to significant fan-in and -out would be larger and slower.

In addition to these 'in-line' processes a local reset scheme allows either end of the link to be reset independently at any time. Although this could corrupt a packet 'in flight' it - together with the inbuilt error tolerance - allows the link to recover normal operation. Thus, if some unpredicted fault mode causes a link deadlock a higher level process can detect the lack of activity and reset it. In practice it is thought more likely that this single chip reset may result from a watchdog following a software crash; either way it allows system recovery when a single, faulting chip is reset.

Initial realisation will be in 130nm CMOS as part of the firstSpiNNaker device and is scheduled for mid 2009.

References

[1] W. J. Dally and B. Towles, “Route Packets, Not Wires: On-Chip Interconnection Networks,” The 38th ACM Design Automation Conf., pp. 684-689, 2001.

[2] L. Benini and G. D. Micheli, “Networks on chips: a new SoC paradigm,” Computer, vol. 35, pp. 70-8, 2002.

[3] J. Bainbridge, and S. B. Furber, “CHAIN: A delay-insensitive chip areaInterconnect,” IEEE Micro., 2002, 22, (5), pp. 16–23.

[4] A. Lines, “Asynchronous interconnect for synchronous SoC design,” IEEE Micro, 2004, 24, (1), pp. 32–41.

[5] Y. Shi and S. B. Furber, ‘‘Error Checking and Resetting Mechanisms for Asynchronous Interconnect,’’ Proc. 18th UK Asynchronous Forum, University of Newcastle upon Tyne, 2006, pp. 24-27, http://async.org.uk/ ukasyncforum18/.

[6] W.J. Bainbridge et al., ‘‘Delay-Insensitive, Point-to-Point InterconnectUsing M-of-N Codes,’’ Proc. 9th IEEE Int’l Symp. Asynchronous Circuits and Systems (ASYNC 03), IEEE CS Press, 2003, pp. 132-140.

[7] L. A. Plana, S. B. Furber, et al., “A GALS Infrastructure for a Massively Parallel Multiprocessor,” IEEE Design & Test of Computers, Volume: 24 , Issue: 5, pp. 454 - 463, Sept.-Oct. 2007.

[8] T. Felicijan, W. J. Bainbridge, and S. B. Furber, “An asynchronous lowlatency arbiter for Quality of Service (QoS) applications”. 15th IEEE International Conference on Microelectronics (ICM'03), Cairo, Egypt, Dec 2003, pp. 123-126.

[9] D. E. Muller, “Asynchronous logics and application to information processing,” Proc. Symp. Application of Switching Theory in Space Technology, H. Aiken and W. F. Main, Ed. , pp. 289-297, 1963

[10] J Sparsø, S. B. Furber, "Principles of Asynchronous Circuit Design", Kluwer, pp.67-69, 2001.

[11] K. V. Berkel, “Handshake circuits: an asynchronous architecture forVLSI programming,”, Cambridge University Press, 1993, pp.35-38.