On-Line Detection of the Deadlocks Caused by Permanently Faulty Links in Quasi-Delay Insensitive Networks on Chip

Wei Song, Guangda Zhang and Jim Garside
School of Computer Science at the University of Manchester
Manchester M13 9PL United Kingdom
{songw, zhangga, jdg}@cs.man.ac.uk

ABSTRACT
Asynchronous networks on chip (NoCs) are promising candidates for supporting the enormous communication needed by future many-core systems due to their low-energy and high-speed. Similar to synchronous NoCs, asynchronous NoCs are vulnerable to faults but their fault-tolerance is not studied adequately, especially the quasi-delay insensitive (QDI) NoCs. One of the key issues neglected by most designers is that permanent faults in QDI NoCs cause deadlocks, which cripples the traditional fault-tolerant techniques using redundant codes. A novel detection method has been proposed to locate the faulty link in a QDI NoC according to a common pattern shared by all fault-related deadlocks. It is shown that this method introduces low hardware overhead and reports permanently faulty links with a short delay and guaranteed accuracy.

Categories and Subject Descriptors
B.8.1 [Performance and Reliability]: Reliability, Testing, and Fault-Tolerance

General Terms
Design, Reliability

Keywords
Deadlock, permanent faults, detection, asynchronous, quasi-delay-insensitive, network on chip

1. INTRODUCTION
Current semiconductor technology has enabled tens even hundreds of cores to be integrated on a single chip. The enormous amount of information exchanged on-chip cultivates the demand for energy-efficient, scalable and reliable networks on chip (NoCs). Most existing NoCs are synchronously built. Custom calibration of the global clock tree is usually required in high performance designs [18]. As an alternative, asynchronous NoCs divide a chip into synchronous islands, which enables individual frequency/voltage control and simplifies chip-level timing closure [3]. They also convey data in a speedy and energy-efficient fashion [5].

The increasing size of NoCs brings reliability issues. Compared with designs using earlier technologies, current sub-micron designs demonstrate reduced yield, worsened process variation, increased rate of faults and shortened device life-time [4]. Future NoCs are expected to operate reliably in the presence of faults.

Fault-tolerance issues have been extensively studied in synchronous NoCs but rarely in asynchronous NoCs, especially in quasi-delay insensitive (QDI) NoCs [16, 6, 5, 15]. The natural tolerance to delay variations of QDI circuits provides the innate protection from delay-related faults [10]. However, they are extremely vulnerable to permanent faults because these faults break the handshake protocol and cause deadlocks. Unlike synchronous circuits, where redundant coding schemes can be employed to detect permanently faulty links or switches [12, 9, 7], the deadlock in QDI NoCs prohibits the propagation of data and therefore disables any data checks. Although off-line scan techniques can locate permanent faults [17], detecting them on-line remains difficult. Timeout is a common way to detect the existence of deadlocks [8]. However, it is difficult to locate the source of a deadlock because the congestion caused by the deadlock will spread over the whole network, leading to timeout in multiple locations.

This paper tries to resolve this problem by on-line detecting the deadlocks caused by permanently faulty links in QDI NoCs. A common pattern has been found in all deadlocks caused by a single permanent fault. Using this pattern, a simple detection circuit is able to report the exact location of a faulty link in the presence of a deadlock. Utilizing the location, further research can be done to retain the network functionality by isolating the faulty links [21, 7, 8].

2. RELATED WORK
Fault-tolerance is a well-studied topic in synchronous NoCs. Transient faults can be dynamically detected using redundant coding schemes. If a data packet is found corrupted, it can either be corrected using the redundant code or re-transmitted [19]. On the other hand, permanent faults can be checked off-line using self-test mechanisms, such as scan chains [21], or detected on-line by searching persistent errors using certain coding schemes [12, 7]. Although permanent faults cannot be recovered, a network may survive with reduced resources. The faulty components can be replaced with spares [12] or isolated using adaptive routing algorithms [21, 7].

Providing fault-tolerance for self-timed NoCs is considered easier compared with QDI NoCs. Since self-timed pipelines have clock-like latch triggers and directly use binary-encoded data, most synchronous techniques can be directly adopted to cope with transient and permanent faults [11, 8].
In QDI NoCs, several unordered redundant coding schemes have proposed to detect and correct transient faults [13, 1, 20]. However, all of them cease to work when the network is deadlocked due to permanent faults. To our best knowledge, there is no code scheme able to detect permanent faults in QDI NoCs. The most effective method existed so far is to detect permanent faults off-line using scan chains [17] while no method has been proposed for on-line detection.

The most related work is the timeout mechanism proposed by Imai and Yoneda [8] in a self-timed NoC using QDI inter-router links. A delay line is used to detect the abnormal data skew among the data wires of the same pipeline stage. However, this method does not work in a pure QDI NoC. Since self-timed pipelines are used inside the router of Imai’s NoC and they do not latch incomplete data, the fault-caused partial data (which leads to the large skew) is isolated in the QDI inter-router link. In a pure QDI NoC, this partial data will propagate to all downstream stages as long as they are ready (see Section 3.2). As a result, all the downstream stages are timeout and multiple faults are reported.

### 3. Deadlocks Caused by Permanent Faults

#### 3.1 Pipeline model

Since most QDI NoCs use 4-phase pipelines [6, 5, 15], this paper is focused on detecting faulty 4-phase pipelines. A 4-phase 1-of-N pipeline is shown in Figure 1. A wide pipeline is usually built from multiple 1-of-N pipelines, shown as slices in Figure 1. Each slice contains several C-elements acting as data latches. A wide OR gate is used to detect the data-completion in each slice. A common ack signal is then generated using a multi-input C-element to synchronize the data transmission of all slices. Although this paper uses 4-phase 1-of-N pipelines, the proposed method is applicable to all M-of-N pipelines.

To provide an easy way for analysing the deadlocks caused by faulty links, Figure 2 simplifies a general 4-phase pipeline into a sequence of places where each place stores a data token or a space (token). In an empty pipeline (Figure 2a), all stages store space tokens, also denoted by the low ack signals. When a pipeline is full of data (Figure 2b), data tokens are stored alternatively because data must be separated by space in 4-phase pipelines. The pipeline stages with data tokens return positive ack to their predecessors.

#### 3.2 Fault analyses

Most permanent faults on wires can be transformed into stuck-at faults [2]. A stuck-at-0 fault denotes a faulty wire that outputs 0 regardless of its input while a stuck-at-1 fault denotes a faulty wire that always drives its output high. According to the types and the locations of stuck-at faults, Figure 3 depicts all the four possible patterns of deadlocks caused by a single permanent fault:

- **Data stuck-at-0** (Figure 3a) With a data wire is stuck at 0, the pipeline is blocked when a 1 occurs on that bit. As a result, all downstream stages eventually go into a temporary state waiting for the withdrawal of the faulty 1.
- **Data stuck-at-1** (Figure 3b) If a data wire is stuck at 1, the pipeline is blocked when it should reset to a space. Consequently, downstream stages will gradually enter another temporary state waiting for the withdrawal of the faulty 1.
- **ACK stuck-at-0** (Figure 3c) Assuming the ack is stuck at 0, its receiving stage will not withdraw its data. As a result, soon all downstream stages will return positive ack and wait for new data.
- **ACK stuck-at-1** (Figure 3d) Assuming the ack is stuck at 1, the receiving stage of the faulty ack would fail to store any incoming data. Consequently all downstream stages will be empty and wait for new data.

It can be observed from the four cases of deadlocks that...
they share a common pattern: the ack of all downstream stages have the same value while they are alternatively valued in upstream stages. Detecting such pattern, a 4-phase pipeline can tell when and where there is a deadlock.

4. DEADLOCK DETECTION IN ROUTERS

A QDI NoC can locate a faulty link by searching the common pattern. In this paper, the detection technique is applied to a 5-port router [15] but it can be used in any routers using 4-phase asynchronous pipelines.

4.1 Router structure

As presented in Figure 4a, the router has five bidirectional ports. Every port is 32 bits wide using 4-phase 1-of-4 pipelines. Each buffer contains one to several pipeline stages depending on its configuration. A switch allocator controls the connection between input and output ports through the central crossbar. Packets in this network are divided into flits. Address information is stored in the head flit, which is followed by a sequence of data flits. The packet is terminated by a tail flit which contains a single positive bit on the end-of-frame (EOF) wire.

The structure of, and the connections between, the input and output buffers of adjacent routers is illustrated in Figure 4b. Other relevant information about this router can be found in Song and Edwards [15].

An output buffer contains only a few pipeline stages directly connected to the crossbar. An input buffer comprises several pipeline stages, an XY router and a controller. When a new packet arrives, its head flit is blocked in the second stage (stg 2) awaiting the XY router to produce a request (rt_r) to the switch allocator. After an output buffer is allocated, denoted by a positive rt_ack, the controller enables the pipeline until a tail flit is noticed through the EOF wire. Working in a QDI fashion, the controller ensures that no data is sent to the crossbar before a path is allocated and the path is not deallocated until the tail flit is safely delivered. For this purpose, the controller controls the enable/reset of the XY router and the ACK ipa0 to the first stage (stg 1).

4.2 Deadlock detection

To detect deadlock faulty links, extra circuits are added to the wormhole router as shown in Figure 5a and 5b. An error detector (Figure 5b) is added in each input buffer to detect deadlocks in the connected link. Since the common pattern deduced in Section 3.2 is valid only when the status of the pipeline is stable (deadlocked), a timeout mechanism is utilized to monitor the activities in the input buffer. If the buffer is inactive/paused for a certain timeout period, the detector is assumed safe to sample the ACK values. To achieve this timeout, an external clock (clk) is introduced. The timeout period is controlled by the counter as shown in Figure 5a.

The connection and the internal structure of the error detector are shown in Figure 5b and 5c. It is a sync/async hybrid module where the asynchronous-related signals and circuits are coloured in grey. The three flip-flops (start, err_r, err_ack)
and \( \text{erpt} \) forms a state machine which controls the timeout process. Its state transition graph is revealed in Figure 5e with cross-references in Figure 5c. Several motion detectors (MDs) are used to monitor the activities on certain asynchronous signals. As shown in Figure 5d, if there is any transition on sig when start is high, act is set positive.

Bearing in mind that the input buffer is downstream to the possibly faulty link (Figure 5b), the timeout process can be described as follows:

**Idle**: After reset, the error detector enters the Idle state and all registers are low.

**Start**: After a timeout period (necessary for the reset of MDs), register start is set high (state transits to Timeout) to enable all MDs to monitor the activities of \( \text{ipa0} \) (new data transmission), \( \text{ipa1} \) (new incoming data) and \( \text{rt}_{\text{ack}} \) (new path allocation).

**Timeout**: Monitoring for a whole timeout period, if any MD outputs high (case 1), the input buffer is not deadlocked and the state returns to Idle. Otherwise, the buffer may be deadlocked due to a fault if one of the following three cases is true: 2 the buffer is transmitting a packet (denoted by the path allocation \( \text{rt}_{\text{ack}} \) and the pipeline stage 1 downstream to the fault) has equal input/output ACKs \( \text{ipa0} = \text{ipa1} \), matching the downstream part of the shared pattern of deadlocks; 3 an ACK stuck-at-1 or a data stuck-at-0 fault may cripple a head flit causing an empty input buffer (unallocated \( \text{rt}_{\text{ack}} \)– and negative ACKs \( \text{ipa1} \), \( \text{ipa0} \)); \( \text{ipa1} \) an unallocated input buffer (\( \text{rt}_{\text{ack}} \)– and \( \text{ipa0} \)) with an arriving tail flit (\( \text{ipof} \)) can be caused by a fake tail flit produced by a data stuck-at-1 fault on the EOF wire. In the presence of any of the three cases (2, 3 or 4), err\(_r\) is set high (state transits to Test) to check the upstream pattern in the adjacent output buffer (Figure 5b). If none of the three cases appears (case 5), the input buffer is paused due to congestion and the state returns to Idle.

**Test**: In this state, the error detector waits for the adjacent output buffer (upstream pipeline stages) to verify the upstream part of the deadlock pattern. The C-element in the output buffer (Figure 5b) is enabled (err\(_r\)). It returns a positive err\(_r\)ack if sig 1 has equal ACKs (\( \text{opa} = \text{oa} \) violates the deadlock pattern). Consequently the state machine is reset through the asynchronous reset shown in the bottom of Figure 5c (case 6). If err\(_r\)ack remains low for a whole timeout period, denoting the deadlock pattern has been stable for a long time, the link is confirmed deadlocked due to a permanent fault. In this case, the error detector reports the deadlock through \( \text{erpt} \) (state transits to Confirm).

**Confirm**: The link is confirmed deadlocked. If the fault is intermittent and recovered, err\(_r\)ack would be positive afterwards, which then resets the state to Idle (case 6).

### 4.3 Other technical issues

The mixture of sync/async circuit is one of the difficulties of this design. The sync/async interface in the error detector requires no synchronizer and metastability does not affect the detection accuracy. The asynchronous MDs and the central AND gate (highlighted with slash lines) in Figure 5c act as a shield to ensure that the values of the three cases (2, 3 and 4) are read only when they are assumed stable as they have been inactive for a whole timeout period. Even under extreme conditions when the input buffer suddenly resumes after congestion and the state mistakenly enters Test, the C-element in the upstream output buffer (Figure 5b) will correct the mistake by setting err\(_r\)ack to high, which then asynchronously resets state machine (the bottom of Figure 5c, case 6). Moreover, the error detector ensures its accuracy by reporting a deadlock only when the deadlock pattern has been found and remained stable for a whole timeout period (in state Test).

The total latency \( T_f \) of reporting a fault from its appearance has two parts:

\[
T_f = T_d + T_r
\]

where \( T_d \) is the time used to form a deadlock and \( T_r \) is delay of reporting this deadlock. \( T_d \) is out of the control of the deadlock detection. It is short on a busy link but infinite on an idle wire. Nevertheless, no damage is made if no deadlock is formed. The report latency \( T_r \) is a more important factor in evaluating the detection speed. The upper and lower bounds of \( T_r \) can be described as:

\[
2T_i \leq T_r \leq 4T_i
\]
5. PERFORMANCE ANALYSES

The asynchronous routers with and without deadlock detection have been implemented using a 130 nm standard cell library and synthesized by the Synopsys Design Compiler. Both routers share the same configuration of five input/output ports, one pipeline stage in output buffers and two in input buffers. The post-synthesis netlist, annotated with gate latency, is simulated in a SystemC/Verilog mixed environment where network interfaces and processor elements are implemented in SystemC.

Table 1: Overhead of deadlock detection

<table>
<thead>
<tr>
<th></th>
<th>Original</th>
<th>Detection</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area (µm²)</td>
<td>33,881</td>
<td>34,629</td>
<td>2.2</td>
</tr>
<tr>
<td>Throughput (MByte/s/node)</td>
<td>503.2</td>
<td>483.9</td>
<td>-3.8</td>
</tr>
<tr>
<td>Latency (µs)</td>
<td>61.5</td>
<td>64.1</td>
<td>4.2</td>
</tr>
</tbody>
</table>

![Figure 6: Toggle rates with various traffic loads](image)

![Figure 7: Power with various traffic loads](image)

![Figure 8: Detection power with various timeout frequencies (a) and clock frequencies (b)](image)

mittent faults. When a fault disappears, the resumed link will trigger err_ack and reset the state machine, which then withdraws the fault report (errt –).

Figure 6: Toggle rates with various traffic loads

The asynchronous routers with and without deadlock detection have been implemented using a 130 nm standard cell library and synthesized by the Synopsys Design Compiler. Both routers share the same configuration of five input/output ports, one pipeline stage in output buffers and two in input buffers. The post-synthesis netlist, annotated with gate latency, is simulated in a SystemC/Verilog mixed environment where network interfaces and processor elements are implemented in SystemC.

Table 1 reveals the performance overhead of adding deadlock detection. The extra logic leads to a marginal area increase of 2.2%. Its speed overhead is also negligible. Collected from 4x4 networks injected with uniform random traffic (using 128-byte fixed-length packets), the saturation throughput per node drops by 3.8% while the average minimum packet transmission latency is prolonged by 4.2%.

The extra wires added by the error detector are unprotected as they are assumed significantly less active than data wires. Using two y-scales (linear and Log10), Figure 6 reveals the average toggle rates of the extra wires (Error) and the data wires (Data) in a NoC injected with various loads. The timeout and clock frequencies are set to 2 MHz and 100 MHz respectively. It is shown that the extra wires have the highest toggle rate of 1 MHz in an idle network and the activity decreases with the increased traffic. The highest rate is exactly half of the timeout frequency because the detector is constantly checking ACK stuck-at-1 deadlocks (case 3 in Section 4.2), which may cause empty input buffers, and this check always turns out negative in 2Tc. With increased traffic, the state machine in the error detector rarely enters the Test state and the toggle rate of the extra wires thus drops significantly until it is nearly zero in a saturated network. In a network with 26% load (128 MByte/s/node), the toggle rate of the extra wires is only 1.2% of that of data wires. If this rate is still considered high or the rate in idle networks is unacceptable, they can be further reduced by increasing the timeout period Tc.

The power overhead of the deadlock detection is also examined and revealed in Figure 7 using the same configuration of Figure 6. It is shown both the router and deadlock detection circuit consume more energy with increased traffic. The power of the detection logic is 18.6 µW (29.7%) in an idle network while it increases to 43.4 µW (0.7%) in a saturated network, however the rate compared to the total router power decreases from 29.7% to only 0.7%.

It is found that the power overhead of the deadlock detection is linear with the timeout and the clock frequencies. Figure 8a shows the detection power with various timeout frequencies in a saturated network (clock frequency set to 100 MHz) and Figure 8b reveals the increased power with increased clock frequency also in a saturated network (timeout frequency set to 2 MHz). According to the results, as long as the clock has a higher frequency than timeout, a slow clock should be used for low power consumption.

To test permanent faults, a fault injector is attached to each link wire in the NoC. It randomly generates a permanent fault (stuck-at-1 or stuck-at-0) according to a pre-configured mean time between failures (MTBF). Note that the error detector is able to detect a fault only when a deadlock is caused, the faults on unused wires are not detectable and the faults on infrequently used wires are slow to detect. However, as long as a fault causes a deadlock, it is found and accurately reported soon after. In the simulation, the location accuracy is 100% and no flawless links have been reported faulty. Setting the MTBF to 5 ms in a saturated network, Figure 9 shows the average report latency, which is the delay between the occurrence of a deadlock and the corresponding report. The average report latency approximately increases proportional to 3Tc. The latency variation also matches the estimation in Equation 2.

As mentioned in Section 4.2, the error detector is able to withdraw its report when an intermittent fault dissipates. The disappearance of an intermittent fault always triggers the C-element in the output buffer (Figure 5b) when a fault is reported. This consequently activates err_ack which then asynchronously resets the state machine and withdraws the report. In one simulation, the faults are set to fade away in

![Table 1: Overhead of deadlock detection](image)

![Figure 7: Power with various traffic loads](image)

![Figure 8: Detection power with various timeout frequencies (a) and clock frequencies (b)](image)
6. CONCLUSION

Permanent faults are a significant threat to QDI NoCs as they cause deadlocks. This paper proposed a novel deadlock detection method which reports the accurate location of the permanent faults occurring on inter-router links of a QDI NoC. The detection circuit is lightweight which introduces low area, speed and energy overhead. As long as a deadlock is caused by a faulty link, the location of the faulty link is reported in a maximum of four timeout periods. Thanks to the dedicated common deadlock pattern, the faulty location is accurately identified and no flawless link is mistakenly reported. It is shown that an SDM NoC can use this method to survive from multiple permanent faulty virtual circuits.

Acknowledgement

The authors would like to thank the various grants from the National Natural Science Foundation of China (61272144), the China Scholarship Council, and the Engineering and Physical Sciences Research Council (EP/I038306/1).

7. REFERENCES