A PREFETCHING SCHEME FOR AUTOMATIC REPEAT-REQUEST FAULT-TOLERANT ON-CHIP NETWORK

WEN-CHUNG TSAI¹, DENG-YUAN ZHENG², SAO-JIE CHEN³, YU-HEN HU⁴

¹Department of Information and Communication Engineering, Chaoyang University of Technology, Taichung, Taiwan, ROC
²Novatek Microelectronics Corp., Hsinchu, Taiwan, ROC
³Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, ROC
⁴Department of Electrical and Computer Engineering, University of Wisconsin, Madison, WI, USA
E-MAIL: azongtsai@cyut.edu.tw, r98943094@ntu.edu.tw, csj@cc.ee.ntu.edu.tw, hu@engr.wisc.edu

Abstract:
In a fault-tolerant on-chip network, overhead due to the Automatic Repeat-reQuest (ARQ) operation may significantly impact the performance. In this work, an intelligent prefetching scheme that can effectively mitigate the performance hits from the ARQ overhead is proposed. Specifically, a packet-based retransmission scheme is incorporated to improve payload utilization while using Error-Detecting Code (EDC). A Virtual Cut-Through (VCT) switching is implemented to alleviate drastic performance degradation in dealing with the damaged data caused by noise upsets during packet transmissions. Potential performance advantages in terms of packet latency and transmission throughput were carefully analyzed to validate the practice of our proposed prefetching scheme in the designed network on-chip systems.

Keywords:
Automatic Repeat-reQuest; Fault-Tolerance; On-Chip Network; Prefetching; Virtual Cut-Through

1. Introduction

In the near future, performance requirement and fault-tolerant capability of conventional network architecture will not satisfy the diverse communication requirements in the next-generation on-chip network [1], [2]. To deal with the damaged data caused by noise upsets during packet transmissions, Store-and-Forward (SAF) and Automatic Repeat-reQuest (ARQ) are commonly used in fault-tolerant computer networks [3]. However, the related performance issues have not been thoroughly studied in an on-chip communication environment.

The upsets in a network could lead to faulty communications. That is, the information carried by the transmitted packets could be damaged since the data bits could be changed when they pass through noisy channels. A fault diagnosis using pioneering flits to investigate the network condition was proposed in [4]. Moreover, a number of Error-Detecting Codes (EDC) such as parity bit and Cyclic Redundancy Check (CRC) have been thoroughly studied for detecting various kinds of transmission errors by checking the accuracy of the propagated data packet [5]. In most of the EDC-based network systems, the sender usually adds generated redundant data to the transmitted packet. Then the receiver decodes the whole transmitted packet with the redundant data by an implemented algorithm, and determines whether the received information is correct or not. In other words, the system can estimate whether there exists a fault in a faulty packet.

On-chip communication over a network fabric may be disrupted due to transient faults in the interconnection devices [6]. The faults could injure the data packets which reside in a physical channel between routers. In order to cope with those faults, many fault-tolerant methods have been investigated and developed [7], [8], [9], [10], [11]. In which existing fault-tolerant schemes have been focused on the enhancement of fault coverage, but they may incur significant processing overheads dealing with the faulty conditions and can be inefficient in term of transmission performance.

Therefore, in this paper, we propose an intelligent prefetching scheme to enhance the communication performance of a traditional fault-tolerant network system. A retransmission method is presented for resolving the infected data packets caused by transient faults in the network. This retransmission mechanism is packet-based and can reduce the downgrading of payload utilization due to the use of EDC. Besides, a forwarding of unchecked packets using Virtual Cut-Through (VCT) switching is carried out to prevent drastic performance degradation.
Correspondingly, a trace and flush mechanism is also proposed to handle the erroneous packets that have been propagated to the network. In brief, we provide a fault-tolerant on-chip architecture which could detect faults, tolerate faults, and retransmit the packets infected by faults. Finally and most importantly, the performance degradation of our designed system is graceful when compared with the previous arts.

2. Background

An on-chip network consists of many routers and channels to organize data communication. However, the communication system could be collapsed while some of the data packets are damaged. Therefore, related fault-tolerant theory and implementation will be introduced as follows.

2.1. Fault Detection and Error Control

Preventing the whole system from erroneous communication caused by faulty data packets, the system has to be fault-tolerant. The CRC (Cyclic Redundancy Check) is a common fault-detection technique for detecting errors in digital data processing without the capability of error correction. It is used primarily for checking the accuracy of data after transmission. In a CRC algorithm, a certain number of redundant bits, usually called checksum, are appended to the transmitted packet by the sender. Then the receiver can check whether the carried information agrees with the checksum to determine whether an error occurred in transmission under a certain degree of probability. If a transmission error occurs, the receiver would send a Negative Acknowledgement (NACK) signal back to inform the sender of the fault, and wait for the fault-tolerant processing to deal with such fault condition.

2.2. CRC Theory

The CRC is based on polynomial arithmetic, which computes the remainder of a polynomial divided by another. For example, the carried information 10100101 is represented by the polynomial \( x^7 + x^5 + x^2 + 1 \), where the polynomial takes a single variable \( x \) which coefficients are 0 or 1, and the arithmetic of addition and subtraction are the same as an exclusive-or operation (done in modulo 2). Moreover, multiplication of such polynomials is quite straightforward; that is, the product of one coefficient by another is the combination through the and operator, and their partial products are summed by the exclude-or operator. Division can be done as a long division of polynomials over integers, but follows the arithmetic rules mentioned above. Figure 1 shows an example on the detailed arithmetic process of dividing \( x^5 + x^4 + x^2 + x \) by \( x^2 + x + 1 \).

For achieving the CRC protocol, the transmitted packet contains the original \( m \)-bit information followed by a \( r \)-bit checksum. To compute the \( r \)-bit CRC checksum, a certain fixed generator polynomial of degree \( r \) on which both sender and receiver agree is primarily demanded. First, the sender appends \( r \) 0-bits to the \( m \) bits of information, and the resulting polynomial of degree \( m + r - 1 \) will be divided by the generator polynomial. After the division, the quotient polynomial is discarded, but the remainder polynomial of degree \( r - 1 \) (or less), having \( r \) coefficients, will serve as the CRC checksum. Accordingly, the receiver can inspect the accuracy of the transmission by the following two methods. Firstly, it can divide the first \( m \) information bits of the received data by the same generator polynomial to get the checksum, and verify if it agrees with the last \( r \) bits. Alternatively, it can divide all the \( m + r \) received bits by the generator polynomial and check whether the \( r \)-bit remainder is 0 to determine the data accuracy.

\[
\begin{align*}
\frac{x^7 + x^5 + x^2 + 1}{x^2 + x + 1} &= \frac{x^5 + x^4 + x^2 + x}{x^2 + x + 1} + \frac{x^3 + x + 1}{x^2 + x + 1} \\
&= \frac{x^5}{x^2 + x + 1} + \frac{x^2 + x + 1}{x^2 + x + 1} \\
&= \frac{x^3 + x^2 + x + 1}{x^2 + x + 1}.
\end{align*}
\]

Figure 1. A long division example showing the detailed arithmetic process

2.3. CRC Implementation

For the fault-tolerance purpose, every on-chip channel has to be monitored on the occurrence of faults. Therefore, each channel is equipped with a set of CRC channel encoder/decoder in our design to check the data transmission, looking for the erroneous transmission. To inspect the data transmission between neighboring routers, the following mechanism as shown in Figure 2(a) is needed. (1) Encode the data with a CRC channel encoder at the sender to compute the checksum, and (2) Decode the data with a CRC channel decoder at the receiver to confirm the data accuracy. Once the receiver detects an error by the disagreed checksum, it would send a NACK signal back to the sender. However, appending the CRC checksum to the data would reduce the payload utilization; that is, hurting the traffic throughput. In an improper design where a packet consists of flits, the transmitted data quantity would at most
be doubled if the checksum is appended to each flit. In other words, the valid throughput could be reduced by 50% since the checksum which is propagated by the data transmission channel follows its own flit. Thus, the communication performance would suffer drastic downgrade just for fault detection. To avoid the drastic downgrade in performance, we treat the whole packet as a unit of inspected message instead of a flit. Then, we will append a checksum only after each packet size of transmitted data to reduce the lost payload utilization, as shown in Figure 2(b). Accordingly, the fault-detecting implementation in our scheme is a packet-based CRC mechanism whose payload utilization would increase with the packet size, thus we can make a better balance between fault detection and network performance.

<table>
<thead>
<tr>
<th>Name</th>
<th>Generator Polynomial</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRC-12</td>
<td>$x^{12} + x^{11} + x^3 + x^2 + x + 1$</td>
</tr>
<tr>
<td>CRC-16</td>
<td>$x^{16} + x^{15} + x^7 + 1$</td>
</tr>
<tr>
<td>CRC-32</td>
<td>$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$</td>
</tr>
</tbody>
</table>

3. Methodology

At the moment that the network suffers from the disturbance of permanent faults or transient faults, the transmitted data flits could be damaged by these faults, leading to communication errors. By using a channel error-detection code (e.g., CRC), such infected data packets can be discovered and usually be discarded in case where faulty data is propagating. Therefore, retransmitting the original discarded data flits (using ARQ) must be carried out to compensate the disappeared communication information and then to cope with the faulty data transmission. Store-and-Forward (SAF) switching is commonly used to prevent propagating corrupt packet before data error being detected. However, the support of fault-tolerance ability leads to network performance degradation. To compensate that performance loss, an intelligent prefetching scheme cooperating with Virtual Cut-Through (VCT) switching is introduced in following sections.

3.1. Traditional Store-and-Forward Switching Router

As Figure 3 shows, a traditional router design (Router B) with SAF switching starts to transmit data flit (denoted D) of a packet to the next router (Router C) only when the CRC checksum in the tail flit (denoted C) is received and checked with no error (i.e., Acknowledgement; ACK) at time 5 (T5). As a result, a packet transmission delay in each router is at least the flit number of the packet (3 cycles of delay in Router B from the 3 flits in a packet).
3.2. Prefetching Virtual Cut-Through Switching Router

As Figure 4 shows, to reduce the packet latency in a router, our proposed prefetching router design (Router C) with VCT switching starts to pre-fetch the data flits of a packet in the upstream router, which are detected and corrected respectively at time 3 (T3) and time 4 (T4). Consequently, at time 5 (T5), Router C buffers two more flits than those of Router C in Figure 3.

3.3. Error Handling with Prefetching Scheme

The major design issue of the prefetching scheme is that, as Figure 5 shows, when the checksum flit is received in Router B and detected with error (i.e., NACK) at time 5 (T5), the corrupt data flit had been propagated to Router C. Therefore, Router C must be noticed with the error condition with a sideband control signal (Flush packet) from Router B to discard the prefetching packet flits.

3.4. Router Architecture and Sideband Control

Figure 6 illustrates the designed router and the sideband control signals PKT_END and EXT_CRC_NACK.
3.5. Performance Issues in Packet Retransmissions

In order to support packet retransmissions, some buffer spaces are occupied by the transmitted packets until the corresponding ACKs are received as shown in Figure 3, Figure 4, and Figure 5. Obviously, the communication performance might be inferior to a network system that needs no support fault-tolerance. This is a performance-reliability tradeoff and needs to consider carefully.

4. Experimental Results

In this section, simulations were run in Register Transfer Level using Cadence NC-Verilog. Next, performance analyses of experimental results are discussed in the following sub-sections.

4.1. Simulation Environment

Each channel bandwidth was set to one flit (32 bits) per cycle. Thus, four cycles are required for switching a header flit from an input port to an output port in a pipelining fashion. We assigned to each input port a buffer of 1024 bits as that used in [12]. We used XY [13] and Odd-Even (OE) [14] as the deadlock-free routing methods. Synthetic traffic performance analyses in terms of latency and throughput were carried out on an 8x8 mesh network. The sizes of packets were randomly distributed between 4 to 16 flits (10 flits averagely). In each run of the simulations, all performance metrics were averaged over 30,000 packets after a warm-up session of 10,000 packets. A common traffic pattern, namely uniform, was considered in our experiments. In uniform traffic, a node transmits a packet to any other node with equal probability. In hotspot traffic, uniform traffic is applied, but 20% of packets change their destinations to one of the following four selected nodes [(7, 2), (7, 3), (7, 4), (7, 5)] with equal probability.

4.2. Performance Analyses

In Figure 7, BFT_NoC indicates the performance of a Network-on-Chip (NoC) router without supporting retransmission. Since it is free of overheads (additional checksum flit and buffer space) in implementing the retransmission mechanism, BFT_NoC acts the best performance metrics in both latency and throughput. Besides, RE_FT_NoC_V2 represents the traditional SAF router and RE_FT_NoC_V3 behaves our proposed prefetching VCT router. Both RE_FT_NoC_V2 and RE_FT_NoC_V3 are designed to support retransmission mechanism for the corrupt packet recovery. As shown in Figure 7(a) and Figure 7(b), using XY routing method and compared with the traditional SAF router implementation, our proposed prefetching VCT router achieves about 40% and 8% performance enhancements in terms of packet latency and transmission throughput, respectively.
Using an OE routing method will deliver more routing adaptivity than that of XY routing scheme [14]. We performed the simulation to confirm and validate such superior result. As Figure 8(a) and Figure 8(b) display, our proposed prefetching VCT router still overcomes traditional implementation about 30% and 5% performance enhancements in terms of packet latency and transmission throughput, respectively.

**Figure 7.** Performance variations in (a) latency and (b) throughput under XY routing

**Figure 8.** Performance variations in (a) latency and (b) throughput under OE routing

4. Conclusions

In an on-chip network, not only communication fault-tolerance has to be guaranteed, but also communication performance related to the adopted fault-tolerant mechanism shall be well taken care of. As such, we proposed an intelligent prefetching scheme that can reduce the performance regression of the conventional Automatic Repeat-reQuest (ARQ) based retransmission implementation. In experiments, with the prefetching scheme, the designed on-chip communication system and its operations can be both fault tolerant and performance efficient.

Acknowledgements

This work was partially supported by National Science Council, ROC, under grants NSC-102-2218-E-324-001 and NSC-102-2218-E-002-011.

References


