Proceedings of the ASME 2011 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference
IDETC/CIE 2011
August 28-31, 2011, Washington, DC, USA

DETC2011-48912

A JOINT-CODING SCHEME WITH CROSSTALK AVOIDANCE

Lei Zhou
College of Information Science and Technology
Nanjing University of Aerospace and Astronautics
Nanjing,Jiangsu,210016,China
Email: tomcat800607@126.com

Ning Wu
Fen Ge
College of Information Science and Technology
Nanjing University of Aerospace and Astronautics
Nanjing,Jiangsu,210016,China

ABSTRACT

The reliable transfer in Network on Chip can be guaranteed by crosstalk avoidance and error detection code. In this paper, we propose a joint coding scheme combined with crosstalk avoidance coding with error control coding. The Fibonacci numeral system is applied to satisfy the requirement of crosstalk avoidance coding, and the error detection is achieved by adding parity bits. We also implement the codec in register transfer level. Furthermore, the schemes of codec applying to fault-tolerant router are analyzed. The experimental result shows that "once encode, multiple decode" scheme outperforms other schemes in trade-off of delay, area and power.

1 INTRODUCTION

The rapid scaling of technology into the deep sub-micron regime has been accompanied by a dramatic increase in transistor densities. According to ITRS' prediction, up to 4 billion transistors will be integrated into one chip since 2010 [1]. However, the increasing densities also lead to increasing possibility of instantaneous fault because of more crosstalk noises and leakage current. To increase the reliability of system, the crosstalk avoidance and error detection scheme has become the critical issues in Network on Chip (NOC) design.

In recent years, there has been an evolving effort in error detection and correction mechanisms in the communication subsystem, and crosstalk avoidance codes (CACs) are considered as effective scheme to reduce the mutual inter-wire coupling capacitance and hence the energy dissipation of wire segments [2]. Yu *et al.* [3]. proposed an adaptive error control method for switch-to-switch links in a variable noise environment, to meet reliability requirements and achieve energy-efficiency. Srinivas *et al.* [4] proposed bus-encoding techniques that decrease crosstalk between wires and avoid adversarial switching patterns on the data bus. However, the data should be divided into groups according to the width. Ganguly *et al.* [5] proposed joint crosstalk avoidance and triple-error-correction/quadruple-error-detection codes, and their performance was evaluated in different NOC fabrics. Nevertheless this coding scheme can applied to any width of data, the code redundancy rate is larger than others.

In this paper, we propose a joint coding scheme which combines crosstalk avoidance coding with error control coding. The main idea of this scheme is to represent datawords into Fibonacci numeral system and add parity bits into coding, for providing the fault detecting capability and avoiding crosstalk noise simultaneously. Furthermore, the RTL level implementation of CODECs is offered. The codec is applied to fault-tolerant router based on End-to-End protocol and Point-to-Point protocol, and the performance of these error controlling schemes is also analyzed.

2 Error Control In Noc Links

The proposed coding scheme is based on the commonly used interconnect architecture Mesh, as shown in Fig. 1. Each router connects neighbors in four directions. Data exchange between the functional blocks takes place in the form of packets. This scheme divides packets into fixed-length flow control units (flits),as shown in Fig. 2, with buffers storing only a few flits.

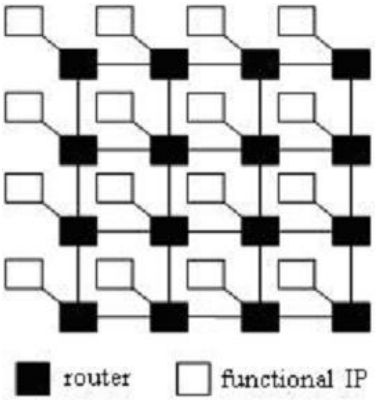


FIGURE 1. The Topology of 2D-mesh

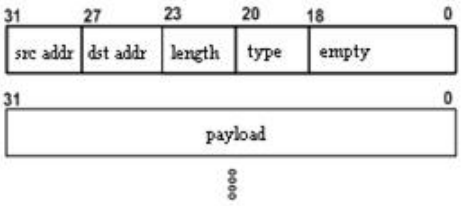


FIGURE 2. Structure of Packet

At most 8 flits compose a packet, including one header flit and 7 payloads. Header flit, which contained routing information, like source and destination address, packet length, etc, enables the switches to establish a path and subsequent flits simply follow this path in a pipelined fashion.

The purpose of the error control mechanism is to deliver the datawords over channel reliably. The mechanism can be classified into two ways: End-to-End protocol and Point-to-Point protocol. End-to-End protocol means the error control only executes once in the data transfer between functional blocks. In Point-to-Point protocol, the error control should be performed in every router the datawords pass through. In general, the hamming, parity code or CRC is applied to detect and the data retransmission is applied to correct the error. Note that the increasing possibility of instantaneous fault leads to the local congestion because of the increasing number of packet retransmission.

Crosstalk is the main course of instantaneous fault, therefore coding datawords by CAC based on error control code can reduce the possibility of error effectively. A few of CACs were proposed in literature. Here we consider Forbidden Pattern Condition (FPC) codes as the Crosstalk Avoidance scheme. It was first proposed in [6]. The forbidden patterns are defined as 3-bit patterns "101" and "010". For example, 1100110 has no forbidden pattern for there is no three consecutive bits. It has been shown in [7] that a code which contains no forbidden pattern experiences maximum crosstalk of no greater than $2 \cdot C$.

3 The Joint Coding

Although CAC and ECC address delay and reliability individually, the combination of CAC and ECC should satisfy the following conditions [4]. Firstly, CAC needs to be performed in first step because it involves nonlinear and disruptive mapping from data to codeword; Secondly, ECC needs to be systematic to ensure that the reduction in transition activity and the peak coupling transition constraint are maintained. And lastly the additional parity bits generated by ECC should be encoded by a linear CAC to ensure they do not suffer from crosstalk delay. According to the constraints of above, the construction of joint code is shown as Fig. 3.

Nonlinear CAC is used prior to other encodings, k bits data is encoded to l bits codeword. After that the additional m parity bits are added to the codeword to contain the error detection ability, then the m bits are further encoded by linear CAC for crosstalk avoidance to obtain m_c bits. Total $l + m_c$ bits are sent over the bus lastly.

Mutyam [8] proposed a bus encoding technique using a variant of binary Fibonacci representation as CAC scheme, which indicates that any n bits vector can be expressed by Fibonacci elements:

$$v = \sum_{k=0}^{m-1} d_k \cdot f^k \qquad d_k \in \{0,1\} \tag{1}$$

where d_k is the k-th bit in vector while f is the Fibonacci element. Here we define the Fibonacci Sequences as follow:

$$f_m = \begin{cases} 0 & (m=0) \\ 1 & (m=1) \\ f_{m-1} + f_{m-2} & (m \geq 2) \end{cases} \tag{2}$$

The literature [8] has proved that the data encoded by binary Fibonacci representation can prevent crosstalk delay; therefore we use binary Fibonacci representation as CAC. Due to the efficient multiple bits error detecting ability of CAC, the ECC only needs to detect one bit error by parity bits. So the encoding algorithm is expressed as Fig. 4.

As the whole codewords satisfy FPC, it has the crosstalk avoidance ability definitely referring to [7]. Here we only need to prove the one bit error detection ability of additional parity bits.

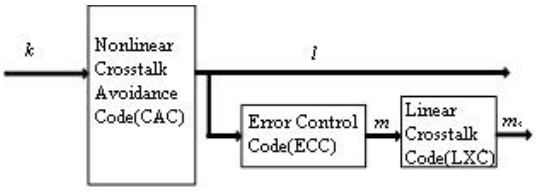


FIGURE 3. Construction of Joint Code

```
Procedure Joint-Code-Algorithm (v)
/* v is the original dataword , the length of CAC is
m */
    if   v >= f_{m+1}  then   //CAC encode
        d_m = 1
        r_m = v - f_m
    else
        d_m = 0
        r_m = v
    endif
    for k = m -1 to 2 do
        if r_{k+1} >= f_{k+1} then
          d_k = 1
        elseif r_{k+1} < f_k
          d_k = 0
        else
          d_k = d_{k+1}
        endif
        r_k = r_{k+1} - f_k · d_k
    end for
    d_1 = r_2
    p = 0                  // get parity code
    for k = 1 to m do
        p=p^d_k
    end for
    if p = 0 and d_m = 0 then          // LXC encode
        d_{m+2}d_{m+1} = 00
    elseif p = 0 and d_m = 1
        d_{m+2}d_{m+1} = 11
    elseif p = 1 and d_m = 0
        d_{m+2}d_{m+1} = 10
    else
        d_{m+2}d_{m+1} = 01
    endif
    return (d_{m+2}d_{m+1}d_m...d_1)
```

FIGURE 4. Joint Code Algorithm

Theorem 1. *The additional parity bits of codeword possess one bit error detection ability.*

Proof. ECC shields datawords by generating two additional parity bits, which refers to the principle of even or odd parity. When ECC gets codeword d_m, d_{k+1}, d_k generated by CAC, it created parity bit value p through bitwise XOR operator. Then the parity bit value is extend to two parity bits d_{m+1}, d_{m+2} to satisfy FPC.

TABLE 1. Truth table of parity bits

p	d_m	d_{m+2}	d_{m+1}
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	1

According to FPC, the first bit d_{m+1} should equal to the last bit of codeword of CAC, so the parity ability is guaranteed by d_{m+2}. For example, if we use even parity, the parity bits should equal to 00 or 11 when $p = 0$. Otherwise the parity bits is 01 or 10 when $p = 1$. The truth table of relationship between p, d_m and parity bits d_{m+1}, d_{m+2} is shown in Table 1.From Table 1 we can deduce that $d_{m+1} = d_m, d_{m+2} = p \bigoplus d_m = p \bigoplus d_{m+1} = d_1 \bigoplus d_2 \ldots \bigoplus d_{m+1}$; it is equivalent to that the last bit d_{m+2} is even parity bit of the whole codeword. The derived process is consistent when using odd parity principle.

4 Implementation of Codec

According to the algorithm of joint code, encoder transforms the n-bit binary data into m-bit Fibonacci code, and adds two additional parity bits to the tail of it. n and m satisfy $2^n < f^{m+2}$, therefore the original 32 bits binary data is mapped to 46 bits Fibonacci code. Adding 2 parity bits, the total width of codeword is 48. The encoder based on algorithm can be implemented using the structure illustrated in Fig. 5.

The original 32 bits data is transferred into encoder, compared with f_{47} to determine the value of d_{46} and d_{47}. In the next stage, the rest of the input r_{46} is compared with f_{45} and f_{46} to generate d_{45}, and then the remaining is transferred to next continuously until d_1 is left. Lastly $d_{47},\ldots d_2 d_1$ is performed by XOR gate to generate d_{48}. The combinational logic depth of encoder is too large, so circuit is divided into 8 sections to generate the part of codes, then merges them. The process of each section consumes 1 cycle, the encoding process is completed in 9 cycles totally.

Fig. 6 depicts the structure of decoder. When received 48 bits codeword, decoder firstly uses bitwise XOR to $d_{47}, \ldots d_2 d_1$ for recreating the parity value p. If $p \neq d_{48}$, there may occur data corruption during data transfer, the error flag e will be marked and routers will request retransmission. Otherwise the circuit transforms the Fibonacci code into binary data according to formula (1) and transfer to the next router.

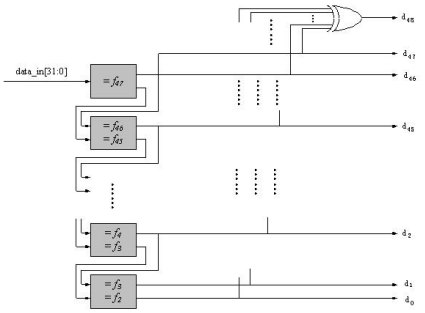


FIGURE 5. The implementation of encoder

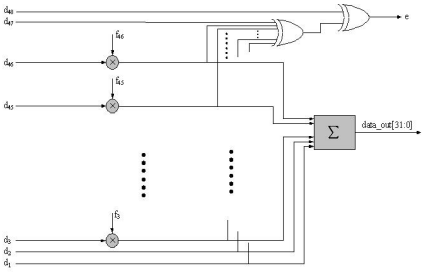


FIGURE 6. The implementation of decoder

5 Experiment Result And Analysis

5.1 The Experiment Scheme

To evaluate the complexity of the CODECs, we implemented the fault-tolerant router which applied to the joint code and constructed 4×4 2D-mesh network. Packet injection follows a uniform distribution. According to default bit error rate (BER), we injected error bits in links between routers to simulate the occurrence of instantaneous fault. We propose three types combination of CODEC and router:

1. Once encode, once decode: The combination follows End-to-End protocol, that dataword is only encoded in the router which connected to source function block and decoded by destination router.
2. Multiple encode, multiple decode: The combination follows Point-to-Point protocol, that dataword is encoded and decoded by every router the dataword passed in the routing path.
3. Once encode, multiple decode: dataword is only encoded in source router, the routers in routing path decode and parity the codeword, and transfer the original codeword directly if codeword has no bit error.

The experiment implements the three error-control schemes mentioned above using Verilog HDL, and then synthesizes them in Design Complier of Synopsys. Based on it, the performance of average delay, power and area is discussed as follows.

5.2 Delay

The delay of CODEC is fixed, therefore the extra delay suffered in three types can be calculated as follow:

$$\Delta Delay1 = Delay_{encoder} + Delay_{decoder} \tag{3}$$

$$\Delta Delay2 = \alpha(Delay_{encoder} + Delay_{decoder}) \tag{4}$$

$$\Delta Delay3 = Delay_{encoder} + \alpha Delay_{decoder} \tag{5}$$

Where $Delay_{encoder}$ and $Delay_{decoder}$ are the delay of the encoder and decoder respectively, the value are 9 cycles and 1 cycle respectively according to the implementation in section III. α expresses the average hops of 2D-mesh network, this implies that:

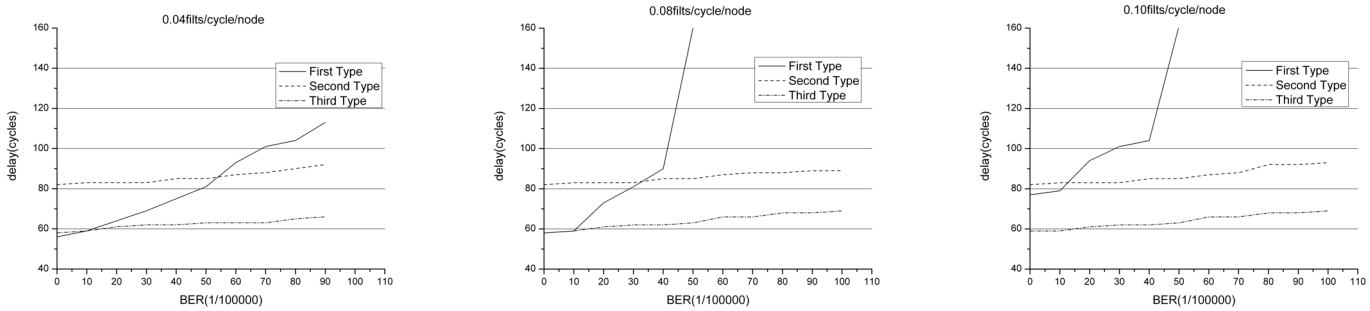
$$\alpha = \frac{\sum_{j=1}^{m}\sum_{i=1}^{n} hop_{ij}}{2\times C_{m\times n}^{2}} \tag{6}$$

where m and n indicate the number of nodes in horizon and vertical direction of the network, and hop_{ij} is hops from node i to j. We use 4×4 2D-mesh network as the experimental network, $m=4, n=4$. In experiments the injection rates follow uniform distribution, therefore $\alpha = 2.67, \Delta Delay1 = 17, \Delta Delay2 = 45.39, \Delta Delay3 = 18.67$. Except the fixed delay of CODEC, the delay suffered from error control also includes the consumption of data retransmission. In networks we assume the BER is 0, 1/100000, 1/10000, 2/10000, 3/10000, 4/10000, 5/10000, 6/10000, 7/10000, 8/10000, 9/10000 and 1/1000 under the injection rate of 0.04flit/cycle/node, 0.08flit/cycle/node and 0.10flit/cycle/node, Fig.7 plots the average delay versus BER.

The gap between delays of three types is slight when BER and injection rates are low. When BER and injection rates increase, the delay of first type promotes distinctly. Although the delay of codecs in first type is smallest, this type involves more packet retransmission which must pass through each router in routing path. For second type, the increasing of delay is limited because that the retransmission happened when router finds error in the routing path, this method shortens the path of retransmission effectively. Note that in low BER and injection rates, the delay in this type is larger than others for this type suffers more delay penalty in encoding and decoding process. Therefore this type is not fitted to low load system. The third type avoids encoding during routing process compared to the second type, therefore the delay outperforms than others.

TABLE 2. Power and Area of Codec

	Power			**Area** (μm^2)		
	Dynamic(mW)	**Leakage**(μW)	**Total**(mW)	**Combinational Logic**	**Sequential Logic**	**Total**
Decoder	1.9959	32.1384	2.0280	365897.35	73483.50	439380.85
Encoder	0.8106	1.7212	0.8124	29069.40	2767.56	31836.96



(a) (b) (c)

FIGURE 7. BER versus Delay

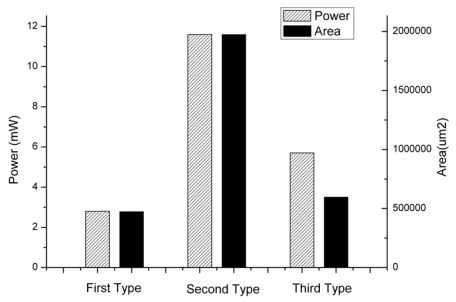


FIGURE 8. The comparison of power and area

5.3 Power and Area

The codecs are synthesized using a SMIC 0.18-μm CMOS standard cell library in Design Compiler of Synopsys, The results of power and area are shown in Table 2.

To simplify the comparison, we only evaluate the consumption of power and area caused by codecs. Fig.8 plots the consumption of power and area under three types.

In first type, encoding and decoding process in source router and destination router respectively, therefore codecs are placed in local port of each router. In second first type, encoding and decoding happened in every hop in routing, codecs should be placed in ports of four directions. In third type, encoding is processed when datawords enter the network and the routers decode in routing path, so one encoder is placed in local port and four decodes are placed in ports of four directions. It is obvious that the second type consumes more power and area than others. Note that the consumption of the third type is larger than the first type, but the gap between them is slight because the consumption of decoder is far less than encoder.

6 Conclusion

In this paper we proposed a joint coding scheme which combines crosstalk avoidance code with error detection code, to solute the reliable problem of NoC in deep sub-micron regime. We mapped the dataword into Fibonacci numeral system to avoid crosstalk, and added parity ability based on it. The implementation of codec in low complexity is showed. Further we analyzed three schemes of codec applying to NoC, the experimental results show that the "Once encode, multiple decode" type outperforms than others from the view of delay. Although the power and area of this type is increasing slightly compared to the best one, it still is the most appropriate scheme in the three types which satisfies the requirement of error control.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (No. 61076019), the Jiangsu Natural Science Foundation (No. BK2008387) and the NUAA Research Funding (No. NS2010115).

REFERENCES

[1] M.S.Wu, C., 2005. "Using a periodic square wave test signal to detect cross talk faults". *IEEE Design and Test of Computers,* ***22***(2), March-April, pp. 160–169.

[2] H.Zimmer, A., 2003. "A fault model notation and error-control scheme for switch-to-switch buses in a network-on-chip". *First IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and Systems Synthesis*, pp. 188–193.

[3] Q.Yu, P., 2009. "Adaptive error control for noc switch-to-switch links in a variable noise environment". *IEEE International Symposium on Defect and Fault Tolerance of VLSI Systems*, pp. 170–174.

[4] S.R.Sridhara, N., 2005. "Coding for system-on-chip networks: A unified framework". *IEEE Transaction on VLSI Systems,* ***13***(6), pp. 655–667.

[5] A.Ganguly, P.P.Pande, B., 2009. "Crosstalk-aware channel coding schemes for energy efficient and reliable noc interconnects". *IEEE Transaction on VLSI Systems,* ***17***(11), March-April, pp. 1626 – 1639.

[6] S. R. Sridhara, N. R. S., 2005. "Efficient on-chip crosstalk avoidance codec design". *Proceedings of IEEE International Conference on VLSI Design*, Jan, pp. 417–422.

[7] C.Duan, V.H.Cordero Calle, S., 2009. "Using a periodic square wave test signal to detect cross talk faults". *IEEE Transaction on VLSI Systems,* ***17***(4), pp. 551–560.

[8] Mutyam, M., 2004. "Preventing crosstalk delay using fibonacci representation". *International Conference on VLSI Design,17th*, pp. 685–688.