Reliable Network-on-Chip Using a Low Cost Unequal Error Protection Code

Avijit Dutta and Nur A. Touba
Computer Engineering Research Center
Department of Electrical and Computer Engineering
University of Texas, Austin, TX 78712

Abstract

The network-on-chip (NoC) paradigm is seen as a way of facilitating the integration of a large number of computational and storage blocks on a chip to meet several performance and power constraints. However due to continued scaling of process technologies, the devices and interconnects have become more susceptible to single event upsets. This paper presents a low cost error correcting code based technique to protect NoC routers against single event upset induced soft errors. An unequal protection error correcting code based methodology is provided for the most commonly used store-and-forward routing strategy. The proposed codes have the same check bit overhead as the conventional single error correcting (SEC) code. The encoding/decoding overhead and latency are also similar to the conventional low cost SEC code. The proposed codes belong to the class of unequal error protection codes as they provide different levels of error correction capability for different portions of the same packet with more protection for the important parts of the data.

1. Introduction

The advent of nanometer technologies has facilitated huge numbers of transistors in a single die. Reduced feature sizes along with increasing transistor densities have transformed the on-chip interconnect into the deciding factor in meeting the performance and power consumption budgets of a design. Several interconnection schemes are currently in use, including crossbars, rings, buses, and network-on-chip (NoC's) [Krewell 05]. The bus and NoC based architectures are the most prominent and have been widely studied in the research community. However, buses suffer from poor scalability. As the number of cores increases, the performance of bus based architectures degrades dramatically. This has led to increased adoption of packet based interconnection networks known as network-on-chip. The NoC architectures offer a variety of advantages. A well designed NoC uses wires more efficiently and uses the same wires for multiple purposes. The reduced requirement of global wires improves power dissipation, signal integrity, less silicon area and better physical routability. The NoC topology can be tuned to the application leading to little arbitration, less wait states, and lower power utilization than a bus. The packet based architecture provides better scalability.

For NoC, the underlying network must meet quality of service requirements (such as reliability, guaranteed bandwidth/latency), and deliver energy efficiency [Park 06]. And all these should be achieved under the limitation of intrinsically unreliable signal transmission media. These limitations are caused by the increased likelihood of timing and data errors [Rossi 05], the variability of process parameters, crosstalk and environmental factors e.g., electro-magnetic interference (EMI) and radiation induced soft errors. The increased sensitivity to soft errors is caused by the reduction in transistor dimensions and the reduction of supply voltage. Ionizing radiation from high-energy neutrons and alpha particles can cause a *single-event upset (SEU)* [Nicolaidis 05] that may alter the state of the system resulting in a *soft error.* Radiation induced single event upsets can cause bit-flips in the sequential logic elements such as the router buffers, memories, and registers.

Some work has been done to protect on-chip interconnects against crosstalk [Rossi 05], [Nieuwland 05], [Bertozzi 02]. In [Nicolaidis 05], [Kastensmidt 05], [Bertozzi 02], and [Lajolo 01], several techniques were proposed to protect the on-chip sequential elements against single event upsets under the assumption that the links are not affected by soft errors. For NoC architectures, both the links as well as the router buffers can be affected by soft errors. In [Frantz 06], a new technique was proposed that can simultaneously deal with SEU and crosstalk effects in the NoC routers. A combination of error correcting codes (ECC) and hardware and time redundancy were used for this purpose. In [Park 06], another method was proposed to address

1550-5774/07 $25.00 © 2007 IEEE
DOI 10.1109/DFT.2007.20

IEEE computer society

simultaneously the problem of link errors due to crosstalk, capacitive loading, and SEUs in router buffers. A flit-based hop-by-hop retransmission scheme and corresponding retransmission architecture were proposed.

The routing mode influences the buffer size needed in the routers and the performance of the system, e.g., packet latency. In packet switching networks, data items have to be buffered at each router before it is sent over. There are two basic types of routing modes commonly used in NoC architectures: store-and-forward routing and wormhole routing. In wormhole routing, messages are sent as worms. The packet is split into flits and the flits are sent in contiguous fashion. The first flit contains the destination address and it reserves the channel through which the subsequent flits are sent. This routing architecture facilitates smaller router buffers but this routing strategy may lead to high data contention and consequently lead to higher message latency. The rest of the paper focuses only on the store-and-forward type of routing.

In store-and-forward routing, the entire packet is stored in the router buffer before it can be forwarded to the next router or the destination core. In this paper, we propose an unequal error protection code to protect data packets from SEUs and crosstalk induced link errors. Recent studies characterizing different bit errors arising from an SEU suggest that 1–5% of the SEUs can cause multiple bit upsets (MBUs) [Maiz 03]. Recent studies show that the most likely multiple-bit-upsets (MBUs) are the adjacent double-bit errors [Maiz 03], [Makihara 00]. Since the life span of a packet in the buffer is small, the likelihood of the same packet being affected by multiple independent SEUs is negligible. Hence the probability of random double-bit upsets is also negligible. The conventionally used codes such as SEC or SEC-DED codes can correct only single-bit errors and hence may result in data loss and data corruption in the presence of single event induced adjacent double-bit errors. The proposed code provides more protection for the bits in the header portion and therefore can prevent data loss. The proposed code can correct all single bit errors and detect all double adjacent bit errors in the entire codeword and additionally can correct all double adjacent errors in the header part. If the header is decoded correctly, then if there is uncorrectable error in the data part, a retransmission can be requested thereby preventing data loss.

2. Store-and-Forward Routing

The most active component of a network is the router and hence is key to achieving reliability and performance standards. The router can also be equipped with logic to send and receive retransmission requests upon detection of unrecoverable errors. The packet while residing in the router buffer can be affected by single event upsets. The packet can also be affected by link errors resulting from crosstalk, coupling noise, and transient faults during transmission.

In store-and-forward routing, the entire packet is stored in the router buffer. During the lifetime of the packet inside the router buffer, the data can be corrupted due to an SEU which can cause either a single-bit flip or double adjacent bit flips. The data can also be corrupted during transmission from one link to the next. This could be due to crosstalk, coupling noise, or transient faults. The error correcting code encodes the data before it is stored in the router buffer. Next it is decoded to enable router arbitration logic to fetch the destination address and port-id for the packet. Next the data is again encoded before it is transmitted to the next link. This ECC organization allows protection against link failures. The basic ECC scheme is shown in Fig. 1. In Fig. 1, the "D + E" block refers to the decoding and encoding of the incoming packet. This stage addresses link error induced data corruption. Note that if the packet is not modified by the routing arbitration unit then the additional encoder is not required. The check bit computed using the previous encoder in the same router can be simply reused.

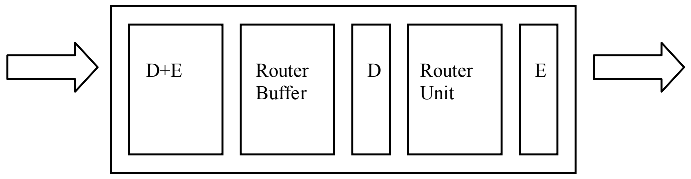

Figure 1. ECC scheme for NoC router

3. Proposed Code

The proposed code falls in the category of unequal error protection codes (UEP) [Masnick 67], [Morelos 94], [Hayashi 00]. These codes provide different levels of protection to different bits of the same word. This is achieved by conditioning the linear dependencies in the parity check matrix (H-matrix) of the code. In general, these codes have the property that some of the digits in a codeword will be decoded correctly only if J_2 or fewer errors occur and others will be decoded correctly only if J_1 or fewer errors occur where $J_1 > J_2$. In [Fujiwara 98], several two-level UEP codes were proposed. These codes were designed as to provide *b*-bit burst error correction capability in one *b*-bit portion of the codeword and either SEC or SEC-DED capability in the remaining portion of the codeword. Another variation of these types of codes was proposed in [Namba 01]. However the check-bit requirement for these codes is high. The proposed single-error-correcting, double-adjacent-error-detecting, selective-double-adjacent-error-correcting (SEC-DAED-SDAEC) code has the following properties:

1) All single-bit errors can be corrected.
2) All double adjacent bit errors can be detected.
3) All adjacent double-bit errors in the header of the packet and the one at the intersection of the header and data part can be corrected.

In a store-and-forward routing scheme, a data packet typically has a header portion followed by a data section which contains both the data and the check bits. Figure 2 shows the basic layout of a packet in the context of the store-and-forward routing.

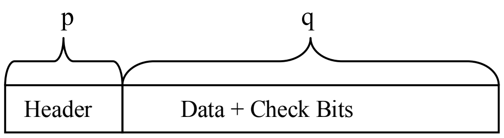

Figure 2. Packet structure for store-and-forward routing

For a $(p+q,k)$ SEC-DAED-SDAEC code, the codeword length $n = p+q$ and message length is k, and the number of check bits $r = p+q-k$. An upper bound on the maximum possible codeword length can be obtained for the proposed code as follows:

$$2^r - 1 \geq 2p + q$$
$$2^r - 1 \geq p + n$$
$$n \leq 2^r - 1 - p \qquad (1)$$
$$r = \lceil \log_2(n + p + 1) \rceil \qquad (2)$$

Equation 1 is derived from the fact that the least number of unique syndromes required for the proposed code is $(n+p)$, n for the single-bit errors, and p for the double adjacent errors in the header. Note that this is a lower bound on the number of unique syndromes required for the proposed code. If the number of check bits is r then the maximum number of unique syndromes is $2^r - 1$. This number should be more than or equal to the minimum number of unique syndromes required for the code. The characteristics of a linear block code are completely determined by its *H*-matrix. Table 1 shows the check bit requirement for different header and data sizes.

Table 1. Check bit requirements

Header, data, check bit	Bound on r	Min r
8,24,r	$r = \lceil \log_2(32 + r + 9) \rceil$	*6*
8,56,r	$r = \lceil \log_2(64 + r + 9) \rceil$	*7*
16,48,r	$r = \lceil \log_2(64 + r + 17) \rceil$	*7*

For the proposed systematic binary linear unequal error protection block code, the H-matrix can be viewed as follows: $H = [H_1 \mid H_2 \mid I]$ (3)

Where H_1 is a $r \times p$ sub-matrix, H_2 is a $r \times (q-r)$ sub-matrix, and I is a $r \times r$ identity matrix.

To detect and correct all single-bit errors, the corresponding error syndromes should be unique. Note that the syndrome for a single-bit error at the *i*-th bit position is the same as the *i*-th column of the *H*-matrix. To uniquely identify all the single-bit errors, all the columns of the *H*-matrix must be unique.

To detect all the adjacent double-bit errors, the corresponding syndromes should be different from all the single-bit error syndromes. The syndrome for a double-bit error is given by the exclusive-or (XOR) of the corresponding columns of the *H*-matrix. So there cannot be any 3-cycle involving adjacent columns in the *H*-matrix. A *k*-cycle refers to a set of *k* linearly dependent columns of the parity check matrix, i.e., when XOR-ed together, the output is an all-zero column. To be able to correct all the adjacent double-bit errors in the header portion (H_1 sub-matrix) the syndromes for the adjacent double-bit errors should be different from each other and also different from all the single-error syndromes as well as from all the double adjacent error syndromes in the H_2 part. Next we define the conditions that must be satisfied by the *H*-matrix for the proposed code:

1) No all 0 columns.
2) All columns are distinct.
3) No linear dependency involving columns C_i, C_j, C_k where $k>j>i$, such that $j=i+1$ or $k=j+1$ or both.
4) No linear dependency involving columns C_i, C_j, C_k, C_m where $m>k>j>i$ and $j<p+1$, such that $j=i+1$ and $m=k+1$ and p is the header size. This condition implies that the double adjacent error syndromes in the header portion are unique.

Condition 1 ensures that no single-bit error case matches the error-free case.

Condition 2 ensures that all the single error syndromes are unique. Every single error syndrome matches one of the columns of the *H*-matrix. Since all the columns of the *H*-matrix are distinct, the single-bit errors are uniquely identifiable and hence correctable. Additionally, this condition ensures that there are no pairs of double errors of the form (i,j) and (j,k) such that the corresponding syndromes are the same. Assume that such double errors exist, then $(C_i \oplus C_j) \oplus (C_j \oplus C_k)=0$, i.e., $(C_i \oplus C_k)=0$ but that contradicts the fact that all the columns of the *H*-matrix are distinct. This ensures that syndromes for adjacent errors of the form $(i,i+1)$ and $(i+1,i+2)$ are different.

Condition 3 ensures that the syndromes for all adjacent double-bit errors are different from that of the single-bit errors. Hence all the adjacent double-bit errors can be detected.

Condition 4 along with condition 2, ensures that a syndrome for an adjacent double-bit error in the header portion is different from all other adjacent double-bit error syndromes. If we assume that the only errors are single-bit errors or adjacent double-bit errors then with an *H*-matrix satisfying conditions 1 through 4, we can uniquely identify the syndromes for all single-bit errors and adjacent double-bit errors in the header portion. Hence we can correct all single-bit errors and detect all adjacent double errors in the whole codeword. Additionally all double adjacent bit errors in the header can be corrected.

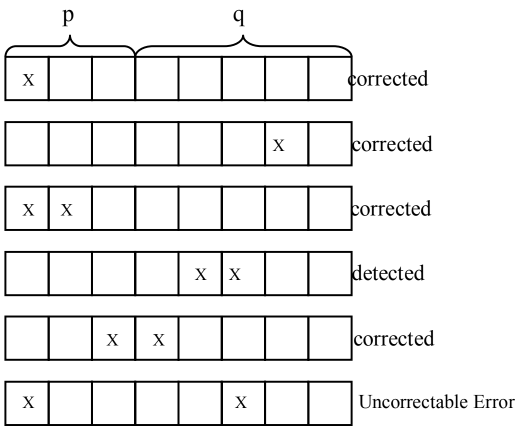

Figure 3. Error profiles

Figure 3 shows the bit-error profile and the error detection/correction capability of the proposed code with respect to those errors. Note that the non-adjacent errors may not always be detectable as they might alias with single-bit error or double adjacent bit errors and hence may lead to miscorrection. However the probability of non-adjacent error is negligible.

Note that in our discussion we are considering only the SEU induced soft errors and hence the only possible errors are either a single-bit error or an adjacent double-bit error. This ensures correct decoding of the header portion in the presence of such an error.

We call the 3-cycles of the type given by condition 3, *forbidden 3-cycles* (*3FC*). We call the 4-cycles of the type given by condition 4, *forbidden 4-cycles (4FC)*. While designing the *H*-matrix, additional constraints can be imposed to reduce the encoding and decoding overhead. This can be achieved by limiting the number of 1's in any row and column of the *H*-matrix.

4. Code Design Procedure

```
Input:  n(codeword length), maxIter, maxBacktrack, r(number of check bits), p(header size)
Output:  H-matrix
avail_col = Set of all non-zero columns of weight > 1
usedSyndromePool = {}
currentCol = r(starts after identity matrix I);
backtrack = 0
while ( currentCol < n-p ) {
   Iter = 0
   validColPool[currentCol] = {}
   while ( iter < maxIter ) {
      Iter++
      C = An untried column from avail_col
      Check for existence of forbidden 3-cycles
      if ( ! 3FCfound ) {
         validColPool[currentCol] =
                validColPool[currentCol] ∪ C
      }    }
   if ( empty(validColPool[currentCol]) ) {
      backtrack++
      if ( backtrack > maxBacktrack ) {
         return    // no code found
      } else {
         currentCol--
         if ( currentCol < 0 )  currentCol = 0;
         continue;
      }    } else {
 sCol=selectMinSynUsage(ColPool[currentCol]))
      add sCol to H-matrix
      add sCol and adjacent double error syndrome corresponding to sCol to usedSyndromePool.
      currentCol++
      backtrack=0;     } }
```

Figure 4. Algorithm to construct H_2

Figure 4 shows the outline of the algorithm used to construct the H_2 submatrix. All the columns of the H_2 matrix should be unique. While adding any new column forbidden 3-cycles (3FC) cannot be allowed as that will lead to an aliasing of a single error with a double adjacent error in the data part. At the same time, the algorithm tries to maximize the sharing of double adjacent error syndromes. This helps in reducing the number of used up syndromes and leaves more flexibility while designing the H_1 matrix. The algorithm maintains a list of syndromes for the single and the double adjacent errors in the constructed code space. A column is a candidate for the next position as long as it does not introduce a 3FC. From a list of candidates, the one that minimizes the number of double adjacent error syndromes in the constructed code space is chosen.

To construct submatrix H_1, whenever a column is added, a check is to be made so that it does not introduce any forbidden 3-cycles (3FC) and forbidden 4-cycles (4FC). This ensures that every single error and double adjacent errors in the header portion have unique syndrome and hence are correctable. This step is different from the H_2 submatrix construction algorithm. Also selectMinSynUsage should be replaced by selectRandCol because no forbidden 4-cycles are allowed in the H_1 submatrix.

Figure 5 shows an H-matrix for a (8, 24, 6) (header, data, check-bit) code. Here the only double adjacent error syndromes used for I and H_2 are (110000) and all its circular shifts, (100111), (011101), (101100), (001111) i.e.,10 distinct syndromes out of 29 (worst case) possibilities. This is achieved by minimizing the number of unique double adjacent error syndromes in the H_2 portion. This leaves more flexibility while searching for the H_1 matrix. After designing I and H_2, the number of available columns for the columns of H_1 and its adjacent error syndromes is (63-30-10) = 23 out of which 8 columns and 8 adjacent error syndromes have to be chosen while avoiding 3FCs and 4FCs.

$$\begin{array}{ccc}
H_1 & H_2 & I \\
\left(\begin{array}{l}
01010111 \\ 10111010 \\ 01110011 \\ 00110110 \\ 01101010 \\ 11000010
\end{array}\right. &
\begin{array}{l}
010100010000110110001111 \\ 100111111000010010010001 \\ 111111110101011100100010 \\ 001111001010111111000100 \\ 110001000001111111111000 \\ 001101100110000111111111
\end{array} &
\left.\begin{array}{l}
000001 \\ 000010 \\ 000100 \\ 001000 \\ 010000 \\ 100000
\end{array}\right)
\end{array}$$

Figure 5. *H*-matrix for proposed (8,24,6) code

$$\begin{array}{ccccc}
H_1 & & \overset{H_2}{C_I} & & I \\
\left(\begin{array}{l}
0111100101000101 \\ 1010110010101110 \\ 1001110101111001 \\ 1110011000010110 \\ 0010011101101000 \\ 1010001110110111 \\ 0010101001011011
\end{array}\right. &
\begin{array}{l}
10100011000110110001111 \\ 00111111000010010010001 \\ 11111111101011100100010 \\ 01111000010111111000100 \\ 10001001001111111111000 \\ 01101101110000111111111 \\ 00000000000000000000000
\end{array} &
\begin{array}{c}
1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1
\end{array} &
\begin{array}{l}
01010001100011011000111 \\ 10011111100001001001000 1 \\ 11111111110101110010001 0 \\ 00111100001011111100010 0 \\ 11000100100111111111100 0 \\ 11001001000111100000000 0 \\ 11111111111111111111111 1
\end{array} &
\left.\begin{array}{l}
0000001 \\ 0000010 \\ 0000100 \\ 0001000 \\ 0010000 \\ 0100000 \\ 1000000
\end{array}\right)
\end{array}$$

Figure 6. *H*-matrix for proposed (16,48,7) code

Table 2. Comparison of proposed SEC-DAED-SDAEC code with other codes

Header + Data	Codes	2-Input XOR Gates	Max Logic Depth	Forbidden 3-Cycles (3FC)	Forbidden 4-Cycles (4FC)	# Check bits
(8+24) = 32	SEC-DBED [Bodnar 03]	100	5	3	40	6
	SEC	114	5	30	143	6
	$(B_2EC)_8$-$(SEC)_{24}$ [Namba 01]	98	5	15	0	6
	DAEC [Abramson 59]	132	6	0	0	7
	Proposed SEC-DAED-SDAEC Code	104	5	0	0	6
(16+48) = 64	SEC-DBED (extended)	231	6	0	263	7
	SEC	204	6	68	316	7
	$(B_2EC)_{16}$-$(SEC)_{48}$ [Namba 01]	222	6	27	0	7
	DAEC [Abramson 59]	296	7	0	0	8
	Proposed SEC-DAED-SDAEC Code	240	6	0	0	7

Figure 6 shows the H-matrix for the (16,48,7) code. Note that the H_2 matrix of a higher dimensional code can be constructed from the H_2 matrix of a lower dimensional code. This can save considerable amount of search time. We illustrate this with the example of the H_2 matrix for the (16,48,7) code which can be obtained from the H_2 matrix of the (8,24,6) code. The H_2 matrix of the (8,24,6) code is a 6×24 matrix which is free of 3FC. In fig. 7, ${}^{32}H_2$ denotes the H_2 matrix for the (8,24,6) code where each R_i denotes each row of the matrix. The ${}^{32}H^c{}_2$ denotes the same matrix as ${}^{32}H_2$ except that the last row is complemented. An all zero row is added to ${}^{32}H_2$ and an all one row is added to ${}^{32}H^c{}_2$. A column vector C_I is introduced at the boundary of

${}^{32}H_2$ and ${}^{32}H^{c}{}_2$ to avoid any 3FC at the boundary. The resultant H_2 matrix is free of 3FC by construction. Note that the submatrix constructed this way has one extra column. Any column can be removed as long as no 3FC is introduced. The easiest choice for removal is the leftmost column. This method can be generalized to construct the H_2 matrix of any higher dimensional code from a lower dimensional code. Note that the H_2 matrix predominantly contains odd-weighted columns and the H_1 submatrix predominantly contains even weighted columns. The row complementation is performed to maintain this property of the H_2 matrix. Note that any row can be chosen for complementation but a good choice is the one where the resultant complemented row has the least number of 1's.

The number of the 2-input XOR gates required for the encoding/decoding can be computed from the H-matrix. It is equal to $\Sigma_{\#rows}\ (row\ weight - 1)$. The encoding and decoding delays are determined by the maximum logic-depth of the encoder and the decoder circuit which is equal to $log_2\,(max.\ 1\text{'s in any row})$.

$$ {}^{32}H_2 = \begin{pmatrix} R1 \\ R2 \\ R3 \\ R4 \\ R5 \\ R6 \end{pmatrix} \qquad {}^{32}H^{c}{}_2 = \begin{pmatrix} R1 \\ R2 \\ R3 \\ R4 \\ R5 \\ R6^{c} \end{pmatrix} $$

$$ \begin{array}{cc} & \qquad\quad C_I \\ {}^{64}H_2 = & \left(\begin{array}{c|c} {}^{32}H_2 & {}^{32}H^{c}{}_2 \\ \hline 00\ldots0 & 11\ldots1 \end{array}\right) \end{array} $$

Figure 7. Constructing H_2-matrix for proposed (16,48,7) code

Table 1 shows the number of XOR gates and maximum logic depth for the syndrome generator, number of forbidden 3-cycles and forbidden 4-cycles, and the number of check bits for a set of different relevant codes.

The first set codes are for packets with 8 bit header and 24 bit data. The SEC-DBED code proposed in [Bodnar 03] can correct all single-bit errors in the packet and can detect all the double adjacent errors in the 8-bit nibbles but cannot detect the double adjacent errors at the nibble boundaries. The 3FCs corresponds to these cases. This code cannot correct double adjacent errors in the header portion. It has a very large number of 4FCs. The SEC code can only correct single-bit errors. It has a large number of 3FCs and 4FCs. The $(B_2EC)_8$-$(SEC)_{24}$ code derived using the method proposed in [Namba 01] can correct all single-bit errors and additionally can correct adjacent double-bit errors in the header part. But it cannot detect all the double adjacent bit errors in the data part because it has some 3FCs. The hardware overhead is also larger than the proposed code. The proposed code, along with the DEC and DAEC codes are the only ones which meet all the requirements for the targeted application as they don't have any 3FC or 4FC. However the check bit overhead for the DEC and the DAEC codes are larger than the proposed code and hence they are less suitable for the router memories where memory area is a major performance constraint. For the DEC code the check bit requirements for protecting the 32-bit (8+24) and 64-bit (48+16) packets are 11 and 14 respectively which are almost twice the requirements for the proposed codes.

For a 64 bit packet with a 16-bit header, a minimum of 7 check bits are required. The SEC-DBED code proposed in [Bodnar 03] was extended for 64 bit and the derived code has a large number of 4FCs. The code cannot correct the double adjacent errors in the header portion. The SEC code also has a large number of 3FCs and 4FCs. The $(B_2EC)_{16}$-$(SEC)_{48}$ code derived from [Namba 01] has some 3FCs. The proposed code is free of all 3FC and 4FCs and hence meets all the requirements for the targeted application. One nice feature for the proposed code is that the overhead varies almost linearly with the header size. Next we discuss the encoding and decoding strategy for the proposed code.

5. Encoding/Decoding Algorithm

The proposed code is systematic. During encoding, the data bits can be directly copied and the check bits are generated using an XOR network corresponding to the *G*-matrix. The decoding algorithm is as follows:

1) Generate the syndrome using an XOR network corresponding to the *H*-matrix.
2) If the syndrome is the all zero vector, then no error is detected, otherwise one or more errors occurred.
3) If the syndrome matches any of the *H*-matrix columns then a single error is detected and the error position is the corresponding column position. The corresponding bit should be flipped to correct the error.
4) Else if the syndrome matches any of the *(header_size-1)* adjacent double error syndromes or the double error syndrome for the error at the boundary of the header and the data parts, then a double adjacent error is detected and the corresponding bit positions are generated using the error correction logic.
5) Else an uncorrectable error (UE) (i.e., a double non-adjacent error or more than two errors) has occurred.

If an uncorrectable error is encountered then it is assumed that the error occurred in the data part of the packet since the header errors are always correctable. When the UE signal is high a retransmission is requested. The header provides the address of the source and destination. Figure 8 shows the basic error detection and correction block diagram. If a non-zero syndrome is encountered, then the OR gate flags an error indication. If the syndrome matches any of the single error syndromes then the syndrome decoder generates a 1 in the erroneous bit position. Otherwise, if the syndrome matches any of the adjacent double error syndromes in the header portion, then the decoder generates 1's at the erroneous adjacent bit positions. Otherwise the output of the syndrome decoder is the all zero output. The syndrome decoder output (for the header part) consists of 3-input OR gates whose inputs are driven by outputs of *r*-input AND gates. The *i*-th output of the decoder is 1 if and only if a single error occurred at the *i*-th bit or a double-adjacent error occurred at $<i,i+1>$ bits or $<i-1,i>$ bits. For the remaining part, only an *r*-input AND gate is required to generate the *i*-th signal. The outputs of the decoder are used to generate the corrected word, by using *n* 2-input XOR gates. If the syndrome is non-zero and does not match any of the single or double-adjacent error syndromes, then an uncorrectable error (UE) is encounter and the UE signal is flagged.

6. Conclusions

The ECC methodology described in this paper provides the ability to correct all single-bit errors and detect all double adjacent error in a packet while correcting all adjacent errors in the header portion of the packet at a very little cost. The presented code provides a very low cost option to protect the packets against the most likely errors in the NoC environment by allowing different levels of protection to different parts of the packet.

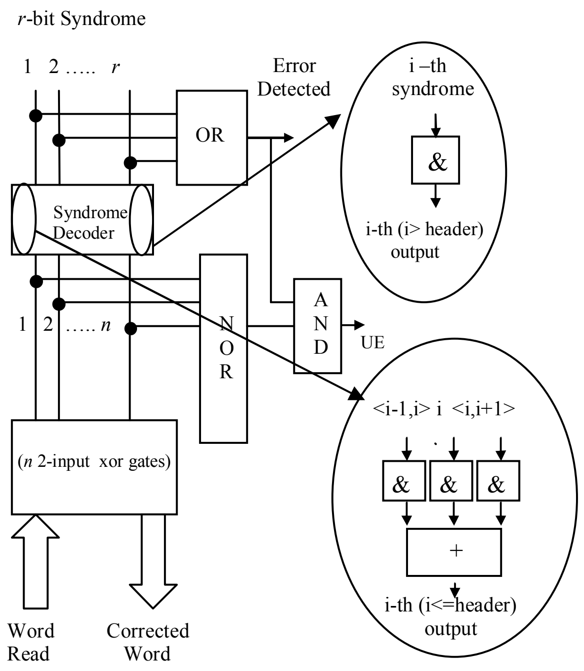

Figure 8. Error detection and correction block diagram

References

[Abramson 59] Abramson N. M., "A Class of Systematic Codes for Non-Independent Errors", *Proc. IRE Trans. on Information Theory*, Vol. IT-5, pp. 150-157, Dec. 1959.

[Bertozzi 02] Bertozzi, D. L. Benini, G. De Micheli, "Low power error resilient encoding for on-chip data buses", *Proc. of Design Automation and Test in Europe Conference and Exhibition*, pp. 102-109, 2002.

[Bodnar 03] Bodnar, L. and G. Chapelle,"A single error correctiondouble burst error detection code", *Proc. of Asilomar Conference on Signals, Systems and Computers*, Vol. 1, pp. 1118-1121, 2003.

[Frantz 06] Frantz, A. P., F.L. Kastensmidt, L. Carro, and E. Cota, "Exploiting ECC redundancy to minimize crosstalk impact", *Proc. of 19th Annual Symposium On Integrated Circuits and Systems Design*, pp. 202-207, 2006

[Fuziwara 98] Fujiwara, E., T. Ritthongpitak and M. Kitami,"Optimal Two-Level Unequal Error Control Codes for Computer Systems", *IEEE Trans. on Computers*, Vol. 47, No. 12, pp. 1313- 1325, Dec. 1998.

[Hayashi 00] Hayashu, T., and E. Fujiwara,"Bit and Byte Error Protection Codes with Two Protection Levels," *Trans. IEICEA*, Vol. J83-A, No. 2, pp. 196-207, 2000.

[Kastensmidt 06] Kastensmidt, F., L. Carro, R. Reis, *Fault-Tolerance Techniques for SRAM-based FPGAs*, Series: Frontiers in Electronic Testing, Springer, Vol. 32, pp. 180-185, 2006.

[Krewell 05] Krewell.,"Multicore Showdown", *Microprocessor Report,* Vol. 19, pp. 41-45,2005.

[Lajolo 01] Lajolo, M.; "Bus guardians: an effective solution for online detection and correction of faults affecting system-on-chip buses", *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, Vol. 9, Iss. 6, pp. 974–982, 2001.

[Maiz 03] Maiz, J., S. Hareland, K. Zhang, and P. Armstrong, "Characterization of Multibit Soft Error Events in Advanced SRAMs", *Proc. of IEEE Int'l Electronic Device Meeting*, pp. 519-522, Dec. 2003.

[Makihara 00] Makihara, A., et al., "Analysis of Single-Ion Multiple-Bit Upset in High-Density DRAMS", *Proc. IEEE Trans. on Nuclear Science*, Vol. 47, No. 6, Dec. 2000.

[Masnick 67] Masnick, B., and J.K. Wolf, "On Linear Unequal Error Protection Codes," *IEEE Trans. Inf. Theory*, Vol. IT-13, No. 4, pp. 600-607, Oct. 1967.

[Morelos-Zaragoza 94] Morelos-Zaragoza, R.H., and S.Lin," On a Class of Optimal Nonbinary Linear Unequal-Error-Protection Codes for Two Sets of Messages," *IEEE Trans. Inf. Theory*, Vol. IT-40, No. 1, pp. 196-200, Jan. 1994.

[Namba 01] Namba, K., and E. Fujiwara,"Unequal Error Protection Codes with Two-Level Burst and Bit Error Correcting Capability", *IEEE International Symposium on Defect and Fault Tolerances*, Vol. 47, No. 12, pp. 1313- 1325, Dec. 1998.

[Nicolaidis 05] Nicolaidis, M, "Design for soft error mitigation**",** *IEEE Trans. on Device and Materials Reliability*, Vol. 5, Iss. 3, pp. 405-418, Sept. 2005.

[Nieuwland 05] Nieuwland, A.K.; A. Katoch, D. Rossi, C. Metra, "Coding techniques for low switching noise in fault tolerant busses", *Proc. of IEEE International On-Line Testing Symposium*, pp. 183-189, 2005.

[Park 06] Park, D., C. Nicopoulos, J. Kim, N. Vijaykrishnan, and C.R. Das, " Exploring Fault-Tolerant Network-on-Chip Architectures", *Proc. of International Conference on Dependable systems and Networks (DSN 06)*, pp. 93-104, 2006.

[Rossi 05] Rossi, D., C. Metra, A.K. Nieuwland, and A. Katoch, "Exploiting ECC redundancy to minimize crosstalk impact", *IEEE Design and Test of Computers*, Vol. 22, Iss. 1, pp. 59-70, Jan 2005.