International Journal of Electronics and Communication Engineering (IJECE)
ISSN 2278-9901
Vol. 2, Issue 4, Sep 2013, 119-130
© IASET

ENCODING ALGORITHM FOR CROSSTALK EFFECT MINIMIZATION USING ERROR CORRECTING CODES

SOUVIK SINGHA[1] & G. K. MAHANTI[2]

[1]Department of Computer Science & Informatics, Bengal Institute of Technology and Management, Santiniketan, West Bengal, India

[2]Department of Electronics and Communication Engineering, National Institute of Technology, Durgapur, West Bengal, India

ABSTRACT

In this paper we present an analysis of crosstalk effects on buses implementing error correction codes. Crosstalk between adjacent wires on the bus may create a significant portion of delay with increasing die size and shrinking wire dimensions. Wires are becoming longer and more resistive and at the same time clock frequencies are rising. If two adjacent wires have simultaneous rising transition, both transitions speed up and a hold violation is possible. Similarly a rising transition on one wire can cause a neighboring wire to falsely transit and lead to a logic fault. In particular, we present an implementing error correcting codes for analysis of crosstalk effects on buses using hamming single error correcting codes. We give algorithms for generating optimal bus encodings and provide a general construction method for a practical class of codes using extra one even parity bit. Encoding and decoding circuits are given for specific codes. We first report the results of an analysis and then using hamming single error correcting codes to reduce the interference in DSM buses.

KEYWORDS: Bus Encoding, CODEC Algorithm, Error Correction, Parity Check Code, Single Error Correction

INTRODUCTION

With the development of VLSI technology day by day, the crosstalk avoidance code reduces the coupling capacitances and hence results in minimization of crosstalk. Crosstalk induces delay, power dissipation, and improvement in signal integrity. When crosstalk avoidance codes are combined with error detecting and correcting codes then signal integrity is improved. With dramatic shrinking of feature size, global buses in deep submicron system-on-chip designs are suffering from high power consumption and large propagation delay. In particular, the propagation delay through long cross-chip buses is already proving to be a limiting factor in the speed of some designs. It has been shown that the delay through a long bus is strongly a function of the coupling capacitance between the wires [1, 2, 3 and 4]. When the cross-coupling capacitance is comparable to or exceeds the loading capacitance on the wires, the delay of such a transition may be twice or more that of a wire transitioning next to a steady signal. We call this delay penalty the “crosstalk” delay. In some high-speed designs where cross-talk delay would have limited the clock speed, the technique of shielding was used. Other methods include increasing wire spacing to reduce the coupling capacitance or increasing wire widths to reduce the ratio of coupling to ground capacitance [4, 5, 6]. The shielding methodology used today is passive in that shield wires are tide statically to ground. The self- shielding codes can be further divided into two categories, memory-less and memory-based [1, 7 and 8]. The memory-based coding approaches generate a codeword based on the previously transmitted code and the current data word to be transmitted. On the receiver side, the data is recovered based on the received codeword. The memory less coding approaches use a fixed code book to generate a codeword to transmit based on the input data. A more useful approach would guarantee a best-case switching scenario for a wire. The concept of active shielding uses shields on either side of the wire that helps to speed up signal propagation delay [8, 9, 10]. To satisfy the delay constraints and guarantee signal integrity several techniques have been proposed, the most common practices consist in inserting repeaters [7, 9 and 16] and shielding the bus wires [13], both active and passive shielding [13, 14] , signal encoding to minimize conflicts [13, 15]. Other techniques have been proposed that rely on bus encoding to prevent crosstalk -induced delay [15]. To face the problem of possible logic errors due to crosstalk, a fault-tolerant bus can be employed. Alternately, error correcting codes can be implemented [12, 13 and 17]. Crosstalk avoidance codes can be used to reduce the effective coupling capacitance of a wire segment. It is needed to incorporate error correcting codes. There are a few joint crosstalk avoidance codes (CAC) and single error correction (SEC) codes [17, 18]. The single error correcting codes are Duplicate- add- parity, Modified Dual Rail (MDR) codes used to reduce the crosstalk [18, 19].In some recent research bus coding technique has been focus. DAP (Duplicate Add Parity), MDR (Modified Dual Rail), DAPBI (Duplicate Add Parity Bus Inverter), and BSC (Boundary Shift code), are all various cross talk avoidance single error correction in on- chip DSM technology in VLSI interconnects [1,12,13]. The multiple error correcting coding techniques are used in network on- chip methodology and NoC can be incorporated multi core SoC [13]. Bus encoding techniques to eliminate crosstalk classes 5 and 6 have been proposed in [3,10]. Recently, delay and energy efficient bus encoding schemes have been proposed in [2,17]. Conversely, other bus encoding techniques have been used to prevent crosstalk but do not correct errors [3,10, 17]. Another cross- talk avoidance in DSM busses detect and correct error that overhead large number of additional wires causing high delay (32- bit line requires 65 lines).

In this paper we analyze the ability of the error correcting codes to limit the crosstalk fault impact on system performance while maintaining the same error correcting ability. We provide algorithms for generating optimal bus encoding codes which is more efficient and minimizing encoding and decoding circuits are given for specific codes. We recall the general scheme for fault-tolerant buses and some basic properties of error correcting codes [1]. First we analyze the ability of single error correcting codes to alleviate crosstalk effects on bus wires, obtained by simply adding one extra even parity check bit. In the next phase we have provided a (7, 4) optimal Hamming code for error detection and correction. In section 2 an overview has been done for error detecting and boundary shift codes. Section 3 presents a new single error correcting coding algorithm using an extra even parity. Section 4 extends error detection and correction by using optimal (7, 4) Hamming code. In section 5 an efficient encoder/decoder circuit has been proposed that is being minimized.

ERROR DETECTING CODES (EDC) AND BOUNDARY SHIFT CODES (BSC)

This section provides error detecting codes and boundary shift codes for single error correction.

Error Detecting Codes

Self-checking circuits are one of the basic building blocks of fault tolerant systems. Fault tolerance on a bus can be implemented using error detecting or correcting codes .Hamming code can be successfully used to detect and correct bidirectional errors affecting bus lines. A code is self- shielding if it does not allow invalid transition. This is possible if the codeword in the code book have a large enough Hamming distance between them. In general, if the minimum hamming distance between any two codeword is *d*, then we can either correct up to $\left\lfloor \frac{d-1}{2} \right\rfloor$ errors, or detect up to [*d-1*] errors [1]. A binary code is linear if the bitwise sum (mod 2) of any two codeword is also a codeword. Victor et al. [2] used a graphical model to represent these constraints. All *n* bit words are represented by a vertex in the graph with edges between two vertices if transitions between the corresponding words are valid. A memory less self- shielding code consists of a set of vertices forming a clique that is a set of vertices with edges between every pair in the set. Properties of the graphs for these codes can be used to obtain a formula for the size of the largest clique for general *n* [2]. The encoding and decoding operations introduces a delay, for instance estimate in terms of numbers of gates, which depends on the code structure and on its error correction capability, which influence the encoding /decoding circuitry complexity.

Boundary Shift Codes

Boundary Shift Code is an invalid transition technique, which represents a transition from one codeword to another so the adjacent bits are changed in opposite directions. This results in the worst case coupling capacitance [1,2]. For example code 01100011 and code 11011010 would be an invalid transition.

A code is self-shielding if it can avoid transition [2]. The dependent boundary is place where two adjacent bits differ and are denoted by the leftmost bit position in the codeword. For example the code 11001110 and 01100111 are having dependent boundary of (2,4,7) and (1,3,5) respectively. The BSC is based on the fact that no overlapping dependent boundary sets form an invalid transition and one bit circular shift converts the code with odd dependent boundary and vice versa.

So by alternation self- shielding codes between the codes is generated. Even dependent boundary is used to duplicate the data word in BSC and then to send the duplicate codeword along with the parity bit with self- shielded transmitted data. The decoder is similar to the DAP decoder with additional multiplexer array to generate the non- circulated code word [1, 7, 15].

If a codebook has codeword only even dependent boundaries, then performing a 1- bit circular right shift yields a new codebook with no even dependent boundaries. Since the two codeword share no dependent boundaries, we can alternate between the two to obtain a self- shielding code. We call this a boundary shift code.

CODING ALGORITHM

The proposed coding algorithm uses a fixed code book in memory less self- shielding code. This results a valid transition between any two codeword. This has been modeled graphically which is shown in figure 2. The model easily generalizes for the additional constraints. For example, the following codeword contains an invalid transition by bit 1 and bit 2:

$C_1 \rightarrow 1\ 0\ 1$

$C_2 \rightarrow 0\ 1\ 1$

In our approach we would like the minimum Hamming distance between any two codeword to be at least *d*, *(d = 2)*. We can add the constraints to the model by only placing edges between any two vertices of corresponding word satisfying the Hamming distance constraints in addition to the valid transition condition. Here *n = 2,* so we add extra one even parity bit to satisfy minimum Hamming distance constraints. Table 1 depicts this.

Table 1

Dataword	Codeword
00	000
01	011
10	101
11	110

Figure 1 shows such a graph for a single error detecting *(d = 2)*, self – shielding code with three wires. Here we use an algorithm that determines if a clique of size *k* exists in the graph.

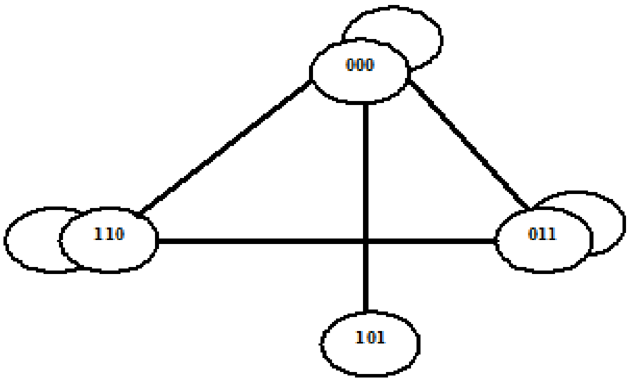

Figure 1: Graphical Representation of Single Error Detecting 3- Bit Self Shielding Codes

Algorithm 1: Solving *k* size clique in a graph

```
Clique _K (G, K)
{
        $V \leftarrow$ set of all vertices in G.
        if (G is complete && $|V| \geq K$) then
          return 1
        else
          for ( $v \in V$ )
          {
            remove vertex u / degree from G and v
            if ($|V| < K$) then
                return 0
            else
                S_G ={ sub graph U / vertex , where $(u, v) \neq 1$}
            if clique _ k (S_K , K ) = = 0 ) then
                remove v from G and V
            else
                return 1
        }
    }
```

Algorithm for Code with Memory

In code with memory, the code book can be a function of previously transmitted codeword. For this case we use two graphs.

Step 1: Consider any vertex of graph G_1 having *n* bit word and an edge between two vertices forming a valid transition.

Step 2: Consider any vertex of graph G_2 having *n* bit word and an edge between two vertices having Hamming distance greater than *d-1*.

Step 3: To find self- shielding minimum distance *d* code with rate $log_2 M$, each codeword must be able to transit a size *M* subset of codeword that are at least a Hamming distance *d* apart from each other. We can determine if such a code exists by the following steps.

Step 4: To determine maximum clique of the graph a binary variable is assigned with each vertex in the graph to denote an edge is the member of clique or not.

Step 5: Find the sum of all binary numbers in clique and the greatest sum will become the maximum clique.

CODING TECHNIQUE AND ERROR DETECTION

In this section, we analyze the impact of single error correcting codes on crosstalk induced bus delay. For this analysis, we have considered one method for transforming 4 bit of data into a 7 bit Hamming codeword, i.e. we have to use 7 × 4 generator matrix *G*. The generator matrix maps the data bits to codeword. In previous section the proposed algorithm finds maximum- rate codes but not encoders or decoders for them. Here we construct a class of practical codes with necessary circuits.

We first define the 4-bit data *D* to be 1 × 4 vector [D_1 D_2 D_3 D_4]. Then we create a 4 × 7 generator *G* such that the product (modulo 2) of the *D* and *G* is the desired 1 × 7 Hamming codeword. Each data bit is represented as given below:

$$D_1 = \begin{pmatrix}1\\0\\0\\0\end{pmatrix} \quad D_2 = \begin{pmatrix}0\\1\\0\\0\end{pmatrix} \quad D_3 = \begin{pmatrix}0\\0\\1\\0\end{pmatrix} \quad D_4 = \begin{pmatrix}0\\0\\0\\1\end{pmatrix}$$

Our (7, 4) Hamming code defines parity bits p_1, p_2, and p_3 as

$P_1 = D_1 \oplus D_3 \oplus D_4$

$P_2 = D_1 \oplus D_2 \oplus D_4$

$P_3 = D_1 \oplus D_2 \oplus D_3$

We represent each parity bit with a column vector containing a 1 in the row corresponding to each data bit included in the computation and a zero in all other rows. This is given below:

$$P_1 = \begin{pmatrix}1\\0\\1\\1\end{pmatrix} \quad P_2 = \begin{pmatrix}1\\1\\0\\1\end{pmatrix} \quad P_3 = \begin{pmatrix}1\\1\\1\\0\end{pmatrix}$$

Now to create a generator that produces code words with the bits ordered D_1, D_2, D_3, D_4, P_1, P_2, P_3 (4 data bits followed by 3 parity bits) use the vectors from the previous steps and arrange them into the following columns [D_1 D_2 D_3 D_4 P_1 P_2 P_3]

The result is following 4×7 generator matrix:

$$G = \begin{pmatrix}1 & 0 & 0 & 0 & 1 & 1 & 1\\0 & 1 & 0 & 0 & 0 & 1 & 1\\0 & 0 & 1 & 0 & 1 & 0 & 1\\0 & 0 & 0 & 1 & 1 & 1 & 0\end{pmatrix}$$

Thus, we can encode any data value to create the codeword using the Hamming code defined by the matrix *G*. To decode the Hamming code defined by matrix *G* the first step is to check the parity bits to determine if there is an error. For this a 3×7 parity check matrix *H* is constructed such that row 1 contains 1s in the position of the first parity bit and all of the data bits that are included in its parity calculation. Row 2 contains 1s in the position of the second parity bit and all of the data bits that are included in its parity calculation. Row 3 contains 1s in the position of the third parity bit and all of the data bits that are included in its parity calculation. The parity check matrix is given below:

$$H = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Multiplying the 3×7 matrix *H* by a 7×1 matrix representing the encoded data produces a 3×1 matrix called the syndrome. If the syndrome is all zeros, the encoded data is error free. If the syndrome has a non-zero value, flipping the encoded bit that is in the position of the column in *H* that matches the syndrome will result in a valid codeword. The following figure 3 depicts the parity check circuit for all the data bits and parity bits that are included in the parity calculation.

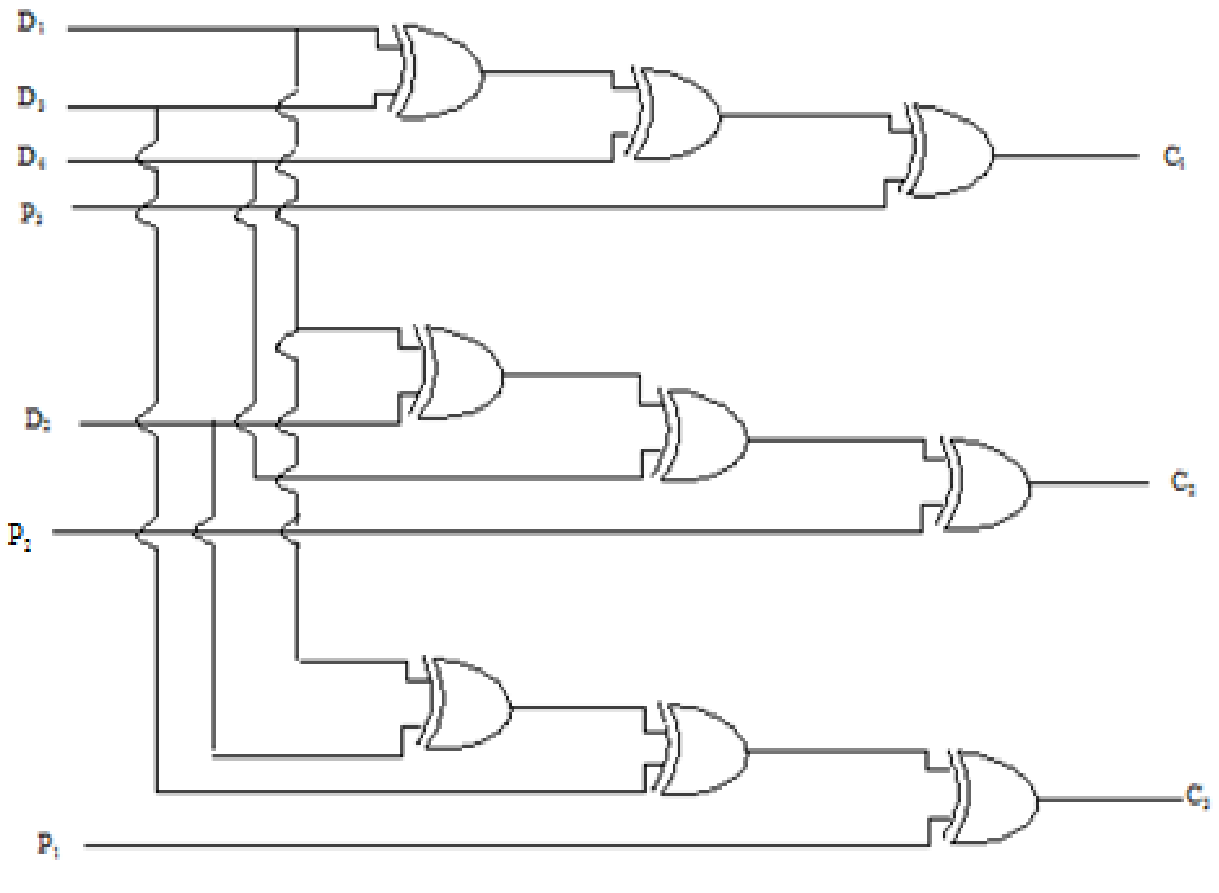

Figure 2: Parity Check Circuit

ENCODER AND DECODER CIRCUIT MODELS

Figure 3 and Figure 4 shows a circuit diagram of an encoder and decoder that can implement error correcting code. The implementation of encoder and decoder is done by using boundary shift code. Two adjacent bits are same and they are denoted by the left most bit of the boundary. Two code words 01100111 and 11001110 have dependent boundaries {1, 3, 5} and {2, 4, 7}, respectively. Since there is no overlap transition must be valid. Design of encoder and decoder is based on boundary shift coding is shown in figure 1. A 4-bit data X_0, X_1, X_2, X_3 , when encoded using boundary shift coding method duplicates all the bits, shifts and adds parity bits to the data bits, and hence the 9-bit codeword {Y_0,Y_1, Y_2, Y_3,Y_4, Y_5, Y_6, Y_7, Y_8} is generated. This 9-bit code is transmitted and at the receiver end decoder is used to decode 9-bit code into 4-bit data just opposite to encoder. Patel et al. [1] use a clock circuit with an inverter to take the select line;

there is no requirement to include the clock circuit. From the result of input/output characteristic we found that output Y_0 and Y_1 both are giving same results, which is similar to input X_0. So in encoder circuit we take S to be always 0 and Y_0, Y_1 are directly connected to X_0 without multiplexer. In decoder circuit diagram here we take S = 1 for getting the X_0 there is required only one multiplexer. So we conclude that our proposed encoder - decoder circuit diagram is minimized to previous work and get same results.

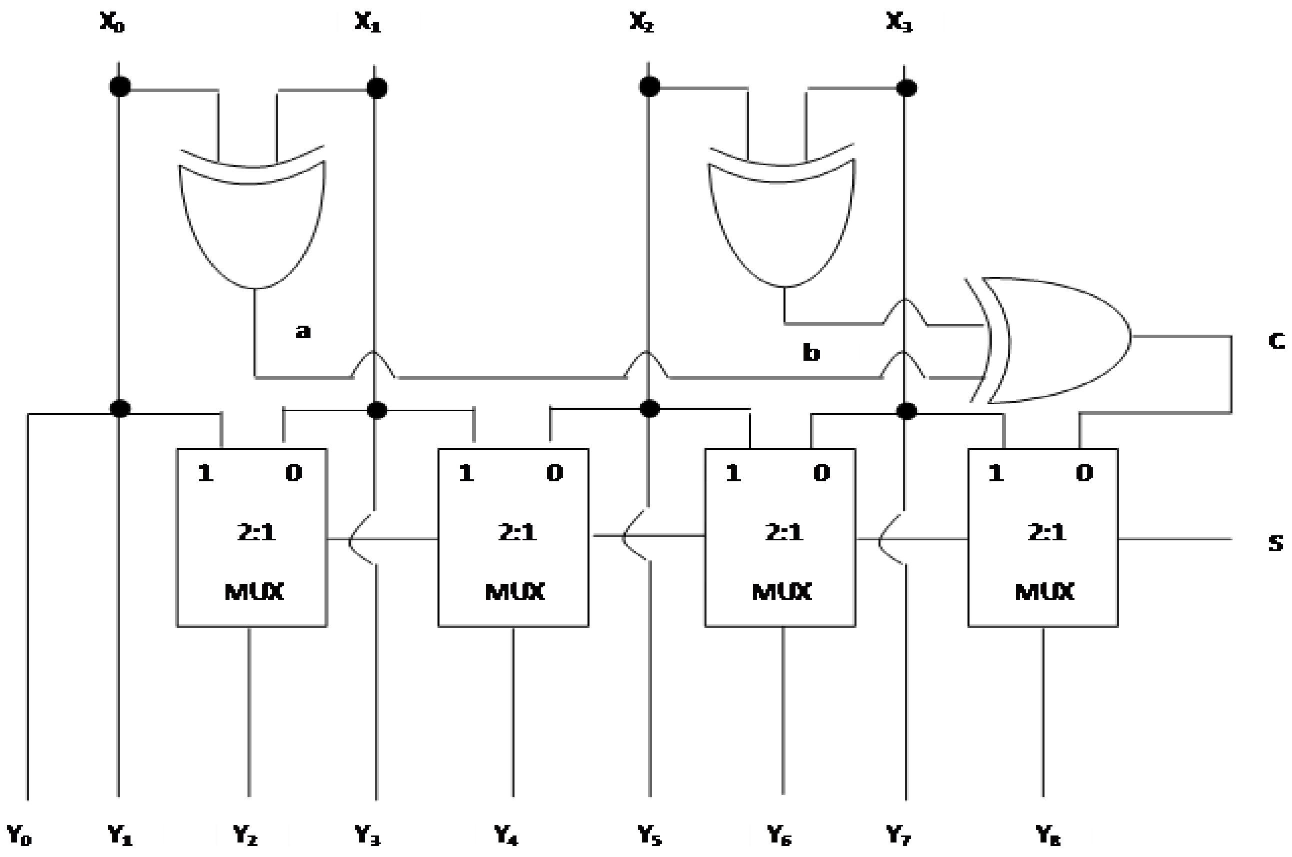

Figure 3: Encoder Circuit Based on Error Correcting Coding Method

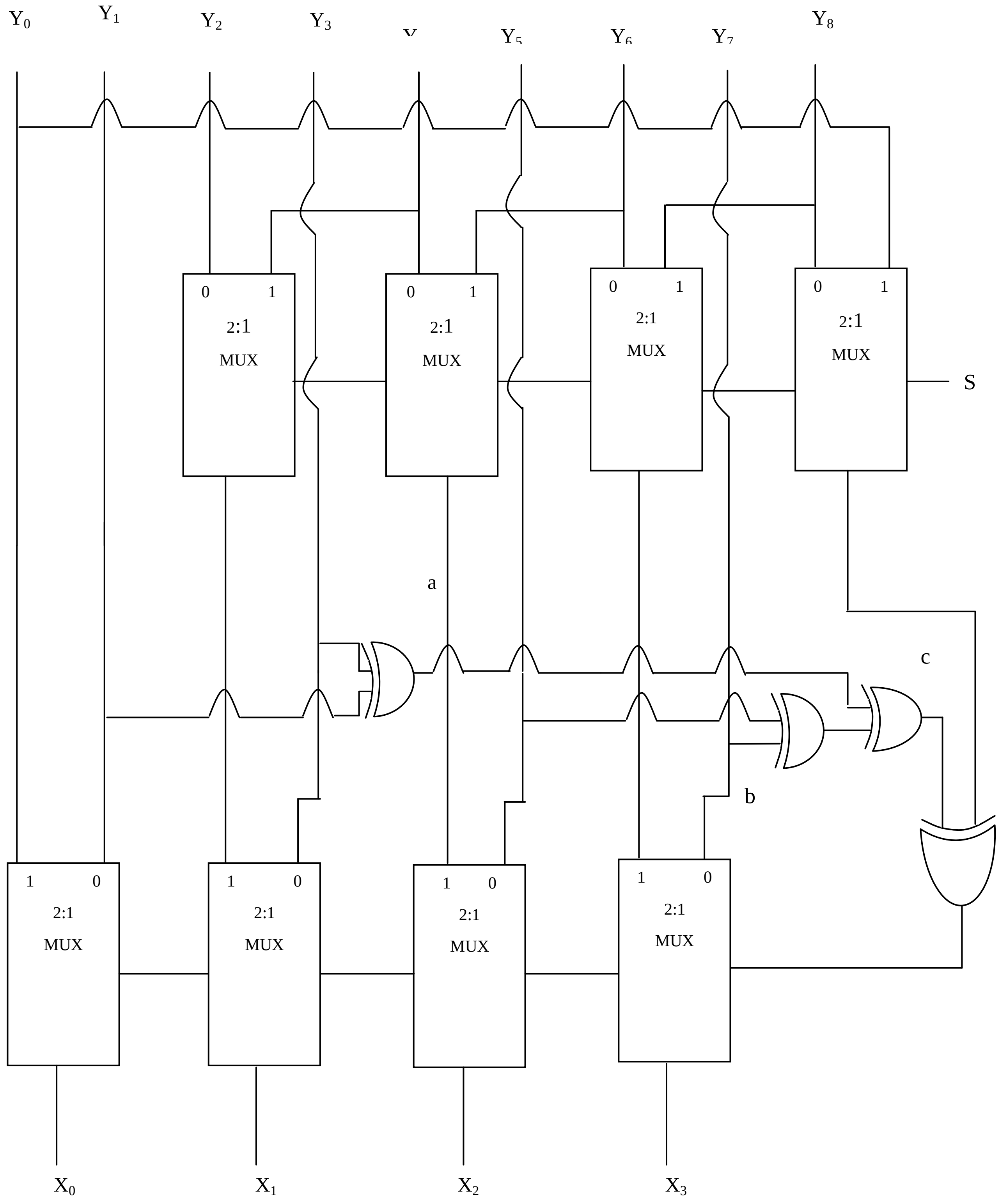

Figure 4: Decoder Circuit Based on Error Correcting Coding Method

Using an extra one bit even parity check code in the above construction gives an infinite class of single error correcting codes. In Table 2 and Table 3 we compare the rates of these codes to the optimal rates. This performance is better than that of the optimal memory-less code. Since the codes constructed are based on very simple error-correcting codes, they can be encoded and decoded efficiently.

Table 2: Single Error Correcting Encoder Output

Input				Output								
x_0	x_1	x_2	x_3	y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8
1	0	1	0	1	1	0	0	1	1	0	0	0
0	1	1	1	0	0	1	1	1	1	1	1	1
1	0	0	0	1	1	0	0	0	0	0	0	1
0	1	0	0	0	0	1	1	0	0	0	0	1
0	1	0	1	0	0	1	1	0	0	1	1	0
1	0	1	1	1	1	0	0	1	1	1	1	1

Table 3: Single Error Correcting Decoder Output

Input									Output			
y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	x_0	x_1	x_2	x_3
1	1	0	0	1	1	0	0	0	1	0	1	0
0	0	1	1	1	1	1	1	1	0	1	1	1
1	1	0	0	0	0	0	0	1	1	0	0	0
0	0	1	1	0	0	0	0	1	0	1	0	0
0	0	1	1	0	0	1	1	0	0	1	0	1
1	1	0	0	1	1	1	1	1	1	0	1	1

SIMULATION RESULTS

The simulation has been done on Xilinx ISE 8 tools and the result is verified. Simulated results show that when both the circuits are simulated in same environment then proposed technique gives results without cross talk interference. This is also demonstrated in Table 2 and Table 3.

We found reduction in crosstalk delay and also reduction in crosstalk noise in Figure 4 and Figure 5 in comparison with Reference [1]. This method also reduces chip area by reducing number of logic gates and multiplexer in comparison with Patel et al.

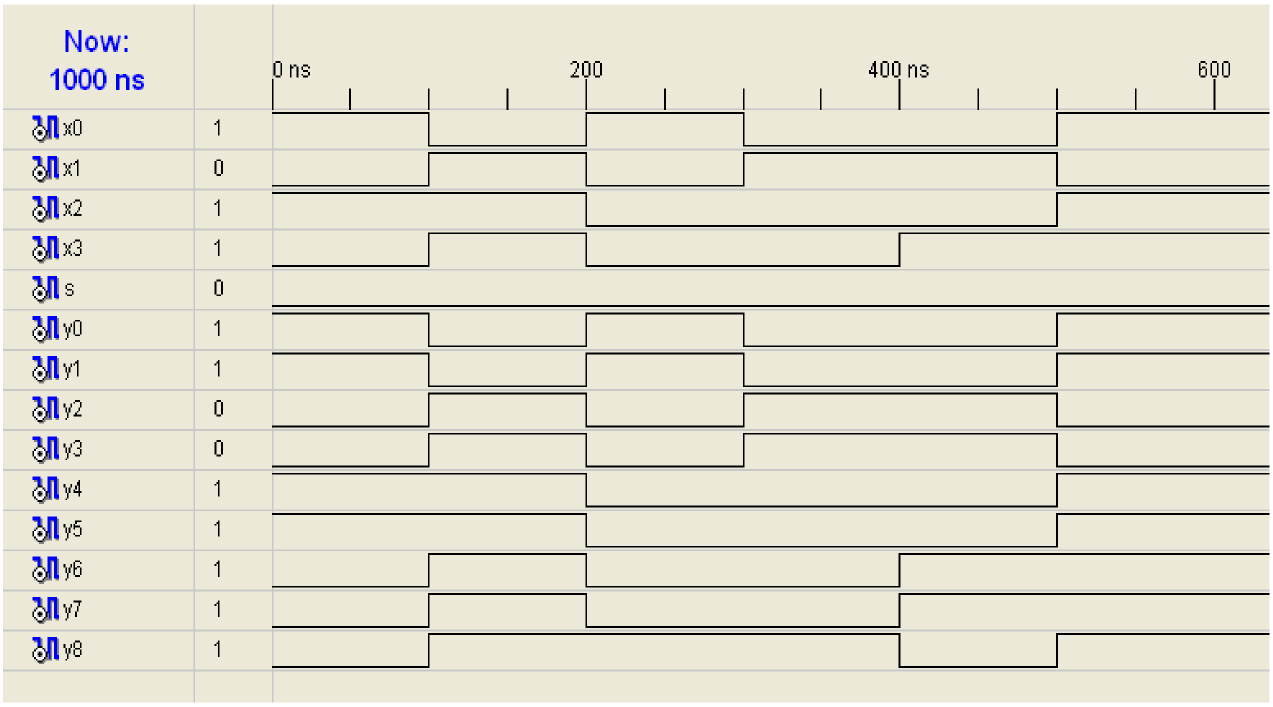

Figure 5: Simulation Results of Proposed Encoder

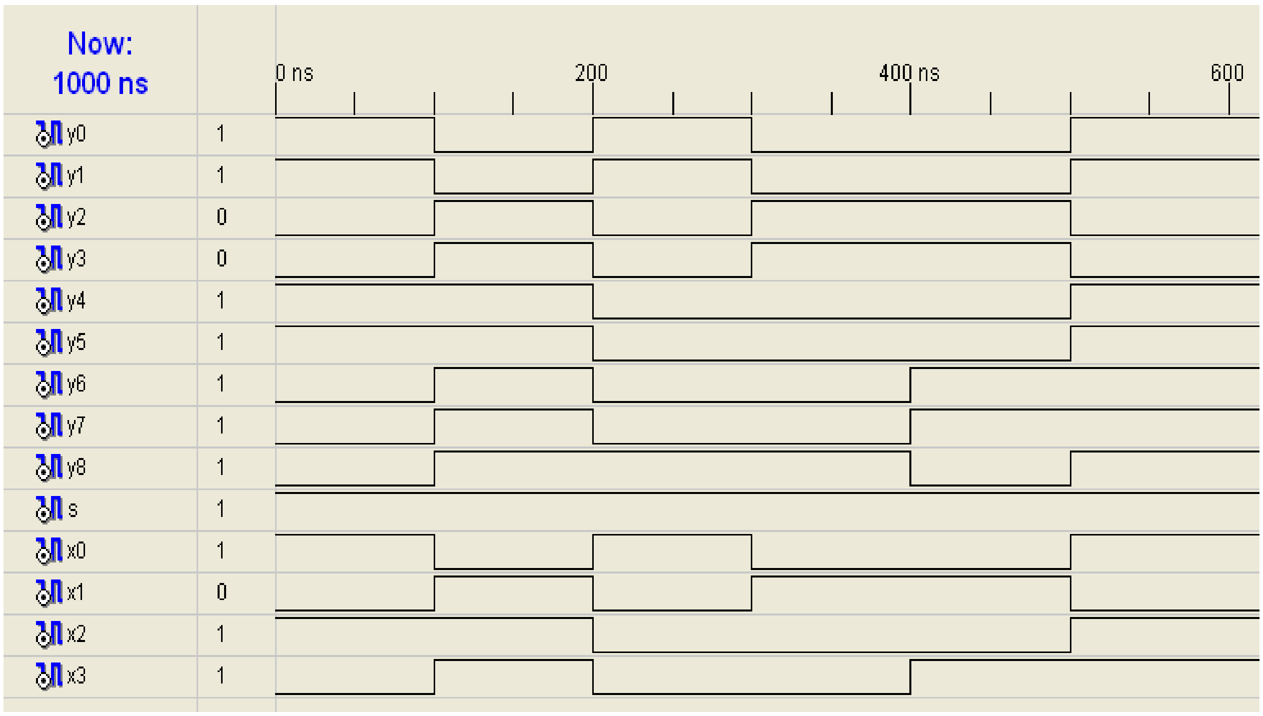

Figure 6: Simulation Results of Proposed Decoder

CONCLUSIONS

In this paper we have presented an analysis of crosstalk effects implementing error correcting coding technique. We have shown that the redundancy introduced by error correcting codes can be exploited in order to avoid the worst case crosstalk induced delay. The proposed method almost removes the worst-case inductive crosstalk using boundary shift encoding method. These codes are derived from conventional error-correcting codes, for the specific case of single error correcting boundary shift codes we give gate level encoding and decoding circuits. This method also reduces the chip area, propagation delay and the circuit is very simple to implement.

REFERENCES

1. Patel, K.N, Markov, I.L, " error- correction and and crosstalk avoidance in DSM busses," in IEEE transactions on very large scale integration (VLSI) systems, Oct. 2004, Volume 12, Issue10, pp 1076-1080.
2. C. metra, M. Favalli, and B. Ricco. " online detection of logic errors due to crosstalk , Delay, and Transient Faults. In proc. of IEEE int. Test Conf., pp 524-533, 1998.
3. B. Victor and K. Keutzer. " Bus encoding to prevent crosstalk delay". Proc. IEEE/ACM Intl. Conf. On Computer Aided Design, Nov. 2001, pp.57-69.
4. A. Vittal and M. Marek-Sadowska, " Cross talk Reduction for VLSI," IEEE Trans, Computer-Aided Design, Vol. 16, no. 3, 1997.
5. M. Favalli and C. Metra, "Bus crosstalk fault-detection capabilities of error-detecting codes for on-line testing". IEEE Trans. On VLSI Systems, sept. 1999. Pp 392-396.
6. P.K.Lala, "Self-Checking and Fault-Tolerance Degital Design". Academic Press, Inc, 2001.
7. K. Hirose and H. Yasuura, "A bus delay reduction technoque considering crosstalk". Proc. Design, Automation and Test in Europe Conf. and Exhibition 2000, pp. 441-445.

8. D. Bertozzi, L. Benini, and B. Ricco,"Energy-efficient and reliable low-swing signaling for on-chip buses based on redundant coding". Proc. IEEE Intl.Symp. on circuit and systems, 2002. pp. 93-96.

9. Duan. C, Cordero. V and Khatri. S.P, "Efficient On-Chip crosstalk avoidance CODEC Design"' IEEE Transactions on VLSI Systems, April 2009, pp 551-561.

10. C. Duan, A. Tirumala, and S. p. Khatri, " Analysis and avoidance of cross-talk in on-chip buses. Hot Interconnects 9, Aug 2001. Pp. 133-138.

11. T. Gao and C. L. Liu. " Minimum crosstalk channelrouting". Proc. IEEE/ACM Intl. Conf. On computer Adided Design, Nov. 1999. pp. 692-696.

12. P. Sotiriadis and A. Chandrakasan, 'Bus energy minimization by transition pattern coding (TPC) in deep sub-micron technologies", In proceedings of IEEE/ACM International Conference on Computer-Aided Design, Nov 2000. pp 322-327.

13. J. F. Wakerly, " Error correcting codes, self- checking circuits and applications". Elsevier north-holland, Inc., New York, 1978.

14. A. Ganguly, P.P. Pande, and B. belzer, " Crosstalk aware chammel coding schemes for energy efficience and reliable NOC interconnects," IEEE Trans. Very large scale integr. (VLSI) syst., vol. 17, no. 11, Nov. 2009, pp. 1626-1639.

15. C. Metra, M. Favalli, and B. Ricco. " Self- checking detection and diagnosis scheme for transient, delay and crosstalk faults affecting bus lines. IEEE trans. Comput., pp 560-574, june 2000.

16. D. Pamunuwa and H. Tenhunen. " Repeater Insertion to Minimise delay in Coupled Interconnects". In Fourteen international conference on VLSI design, pp 513-517.

17. Madhu et al., " Delay and Energy –Efficient Data Transmission for On-chip Buses," In ISVLSI'06, pp 355-360, 2006.

18. D. Rossi, S. Cavallotti, C. Metra. " Error Correcting Codes for Cross talk Effect Minimization", Processing of the 18th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems.(DFT' 03).

19. D Rossi, V.E.S van Dijk, R.P Kleihorst, A.H. Nieuwland, C. Metra, " Coding Scheme for Low Energy Consumption Fault- Tolerant Bus", Processing of the Eight IEEE International On-line Testing Workshop (IOLTW' 02).