School of Electrical, Electronic & Computer Engineering

UNIVERSITY OF
NEWCASTLE UPON TYNE

Fault Tolerant Techniques to Minimise the Impact of Crosstalk on Phase Encoded Communication Channels

Basel Halak, Alex Yakovlev

Technical Report Series
NCL-EECE-MSD-TR-2006-115

2006

Contact:
basel.halak@ncl.ac.uk
alex.yakovlev@ncl.ac.uk

Supported by EPSRC grant EP/C512812/1

NCL-EECE-MSD-TR-2006-115

School of Electrical, Electronic & Computer Engineering,
Merz Court,
University of Newcastle upon Tyne,
Newcastle upon Tyne, NE1 7RU, UK

http://async.org.uk/

Fault Tolerant Techniques to Minimise the Impact of Crosstalk on Phase Encoded Communication Channels

Basel Halak, Alex Yakovlev

May 2006

Abstract

A communication scheme in which symbols are encoded by means of phase difference between transitions of signals on parallel wires is considered. A significant decrease in the reliability of such a channel is caused by capacitive crosstalk between adjacent wires. A more robust high-speed phase encoded channel can be designed by minimising its vulnerability to crosstalk noise. This paper investigates the impact of crosstalk on phase encoded transmission channels. A functional fault model is presented to formulate the problem. Three fault tolerant schemes are introduced which are based on information redundancy techniques and the partial order coding concept. These schemes are simulated with CADENCE using AMS CMOS 0.35µm process. Area overheads, performance and fault tolerant capability of those methods are compared. It is shown that a substantial improvement in the performance can be obtained for four wire channels when using the fault tolerant design approach, at the cost of 25% of information capacity per symbol.

Index Terms-Asynchronous operation, crosstalk, communication channels, error-checking, fault tolerance, information redundancy, simulation, performance, reliability and VLSI.

1. Introduction

On-chip global buses are increasing in length with increasing die sizes, resulting in large propagation delays [1], [3]. The delays of those buses have two impacts, namely:

- They limit the system performance in many high-speed microprocessors [2], [3].
- They lead to an increase in the clock skew, which makes it difficult to accurately distribute a single global clock across the entire system [4].

This trend is anticipated to exacerbate in the future due to the increasing gap between gate delays and interconnect delay brought about by shrinking feature sizes [5]. Globally asynchronous locally synchronous (GALS) electronic system design is a methodology that addresses these problems. In GALS functional modules are designed using conventional design techniques. Each module is complemented by its own local clock generator and a self-timed wrapper that enables the modules to communicate using asynchronous handshake protocols [7], [8]. It is expected that 40% of the electronic design in 2020 will be driven by handshake clocking [6]. Although the issue of self-timed communication protocols has been intensively

investigated in terms of power efficiency, speed, area overheads and reliability in [9], [10], [11], [12], [13], [14], the effect of transient errors on self-timed channel reliability was first addressed in [15]. Transient errors caused by cross-talk, cross-coupling, ground bounce or environment interference, become more prominent as integration increases. Thus signal integrity is put at risk [16], [17]. This motivates the fault tolerance approach to design. Multi-rail phase encoding is a novel fault tolerant, self-timed signalling protocol, it was proposed in

[15]. The data is sent using the phase relationship between differentially delayed copies of a reference signal. Mutual Exclusion elements [18] are employed as phase detectors for data recovery. Figure 1 illustrates how this protocol can be used to send data on a two wire channel. This technique outperforms many of the existing self-timed communication methods such as m-of-n, e.g.1-of-4, Return to Zero (RTZ) in terms of information capacity [15]. It also exhibits high robustness against Single-Event Upsets (SEUs), hence it is more reliable. However, capacitive crosstalk between adjacent wires may deteriorate their phases, which will strongly affect the integrity of the data being sent [15]. This problem can create a bottleneck to the system and may prevent the use of this technique to send data on long wires and/or at high frequencies. Crosstalk can be defined as a disturbance, caused by electromagnetic interference, along a circuit or a cable pair. A telecommunication signal disrupts a signal in an adjacent circuit or wire and can cause the signals to become confused and cross over each other.

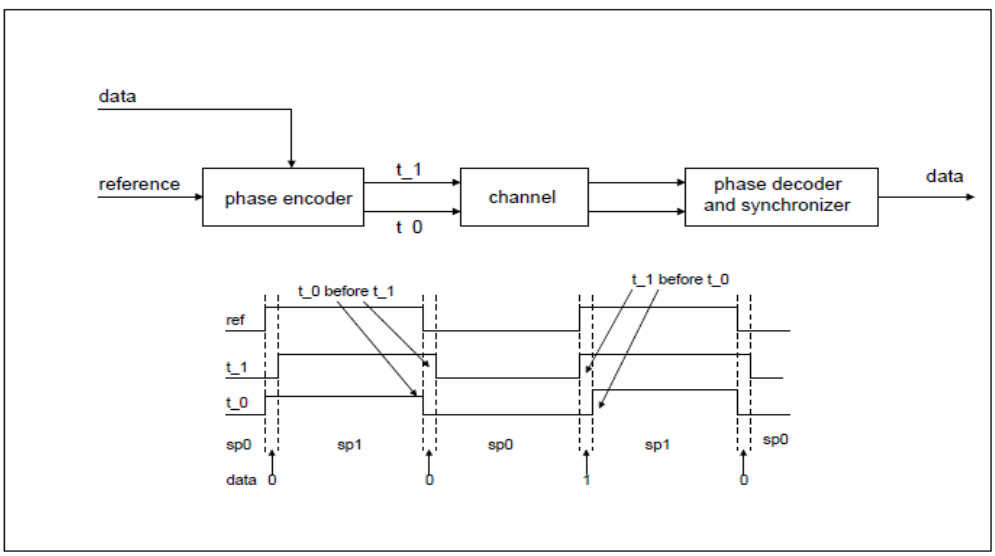


Fig. 1: Phase encoded Transmission Protocol

In the case of adjacent wires the crosstalk noise effect can be explained as follows; when two wires are placed close together, the current flowing down one (which we will call “aggressor”) induces a current in the other wire “victim”. The electric field causes a current in the victim that flows both ways, backwards and forwards. For example if a single electron was at a point along the aggressor, it will tend to repel electrons in the victim in both directions away from that point. This type of coupling is often called “capacitive” coupling. The aggressor wire also generates a magnetic field, which in turn generates a current in the reverse or backward

direction in the victim wire. This type of coupling is often called “inductive” coupling. So crosstalk is a direct result of the electromagnetic field radiated from the aggressor wire, therefore its coupling effects attenuate with distance. This means the crosstalk between non-neighbouring wires is less than that between neighbours. This fact is going to be exploited in the fault tolerant techniques introduced in this article.

There are two types of Crosstalk noise: Functional Noise and Delay Noise. A functional noise occurs when a transition on a wire (A) leads to a glitch on a neighbouring wire (B) as shown in figure 2. A delay noise occurs when two neighbouring wires switch simultaneously, which causes transition slowdown or speedup in the victim wire. This leads in both cases to a reduction in the original time distance between the two transitions as can be seen in figure 3.

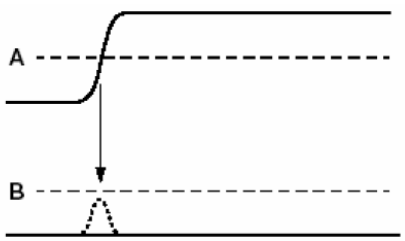


Fig. 2: Glitch due to Crosstalk

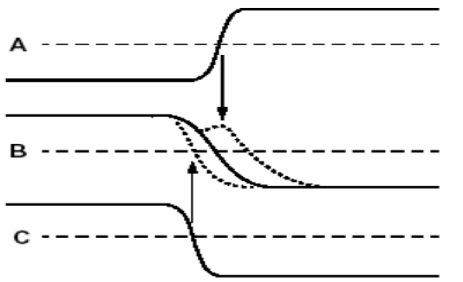


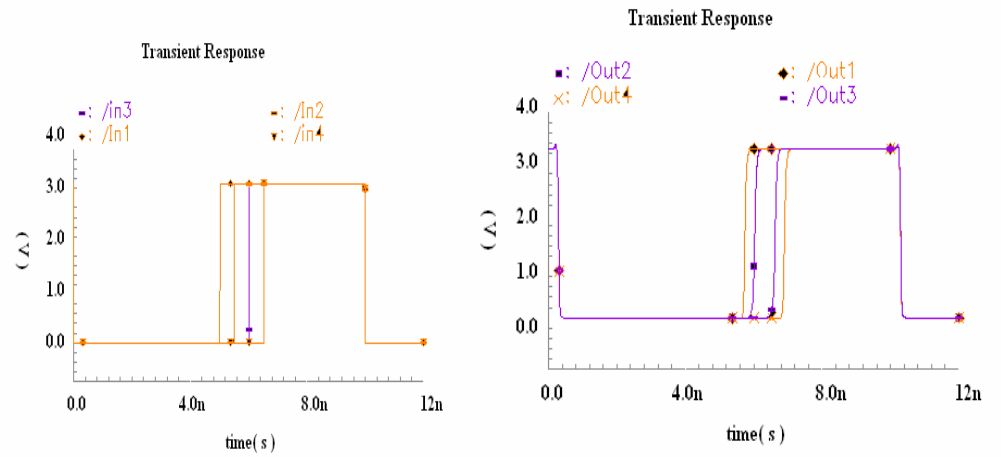
Fig. 3: Transition Slowdown or Speedup due to Crosstalk

In the case of phase encoded channels all wires switch close in time and in the same direction, they can be considered allies, i.e. they reinforce each other. This reinforcement is manifested as a reduction in the original phase between transitions. Figure 4 shows the transient response of the inputs (4a) and the outputs (4b) of a four wire phase encoded channel whose length is 2mm. As can be clearly seen the time distance 500ps between the first and the second transition (in1 & in2) was reduced at the outputs (out1 & out2) to 230ps. This also applies for (in3 & in4). This phase corruption does not generate errors as long as the mutex elements at the receiver side can decide which transition was the first one. However, our simulations showed that the crosstalk noise can in some cases lead to phase conversion i.e. the transitions are received in an order different to their original one. Errors can also occur if glitches (see figure 2) are perceived as transitions by the mutex elements. These errors are filtered out if they happen outside the event window; otherwise they cause faults [15]. This problem is fatal to the multi-phase encoding technique.

The impact of crosstalk noise on communication channels has been addressed in many papers. Researchers have proposed several techniques aimed at reducing the crosstalk induced delays. The insertion of repeaters and shielding of bus wires are the most common methods [19], [20]. The selection of a proper global wire configuration has also been proved to significantly minimise the impact of crosstalk [21]. Furthermore, research has recently shown that fault tolerant techniques can be employed to increase communication channels reliability in the presence of crosstalk [22]. Other techniques rely on crosstalk avoidance codes [23], [24].

This article provides a comprehensive study on the use fault tolerant techniques to minimise the effect of crosstalk noise on multi-phase transmission protocol.

For deep submicron circuits the capacitive coupling is more prevalent, and the delay is dominated by the capacitance and the resistance [25]. Therefore our focus will be only on the impact of capacitive crosstalk.
Note that most of the ideas in this paper will be presented for a four wire channel, which will allow us sufficient level of generality and yet to avoid complex mathematical, algorithmic and circuit solutions.
The organisation of the paper is as follows, Section 2 introduces a functional fault model which formulates the impact of crosstalk noise on multi-phase transmission protocol. Section 3 defines some essential concepts which are used to explain the theory of the fault tolerant techniques presented in this paper. The background theory, implementation and simulation results of each technique are presented and analysed in section 4. Finally the conclusions are drawn in section 5.



(a) (b)

Fig. 4: Crosstalk Impact on Four Wire Phase Encoded Channel

2. Transient Fault Model

As explained previously the information is encoded as a differential phase between signal transitions on wires. The crosstalk noise may corrupt this phase leading to systems failure. In order to tackle this problem, a fault model based on a functional abstraction of the crosstalk errors is introduced in this section. It consists of three types of faults, namely:

- Type 1 is a result of the conversion of phase between transitions on adjacent wires.
- Type 2 is a result of the phase conversion between transitions on non-adjacent wires
- Type 3 which includes both type 1 and 2, i.e. it is a result of the phase conversion between transitions on all wires.

Consider the case of a four wire phase encoded channel. The first wire is denoted *a*. The second is *b*, the third is *c* and the forth is *d*. Each combination of transitions (information symbol) will be represented by those four letters whose order indicates the order of signal transitions in time. For example *badc* means the second wire *b* transition first then the first wire *a* follows, and then the fourth wire *d*. Finally the last transition occurs on the third wire *c*. This notation will be used throughout the article.

Let us now assume that the combination *bacd* was sent. When the combination *bacd* is received, i.e. the transition order was preserved, no errors are said to have occurred. If the combination *abcd* is received, type 1 fault is said to have occurred, i.e. a conversion in phase between transitions on two adjacent wires (a, b). If the combination *bcad* is received, a type 2 fault is said to have occurred, i.e. a conversion in phase between transitions on two non-adjacent wires (a, c). Finally, if the combination *cbad* is received, a type 3 fault is said to have occurred, i.e. a conversion in phase between transitions on two adjacent wires (b, c) and on two non-adjacent wires (a, c).

3. Background Definitions

The first concept to be introduced is *the complete set of conditions (CSC)* for a set of wires which can be defined as follows: the *complete set of conditions* is a group of conditions which describe the order of signal transitions on the wires for each information symbol. For an *n* wire channel the number of these conditions is equal to the combinatorial number $\binom{n}{2}$. In order to form the complete set of conditions, we define a function called *Mutex Function or* M (W1, W2) as follows:

$$M(W1,W2) = \begin{cases} 1: \text{When transition on w1 occurs before that on w2 (w1 > w2)} \\ 0: \text{Otherwise (w1 < w2)} \end{cases}$$

For example in the case of a four wire phase encoded channel, six conditions are needed to identify each combination of transitions (information symbol). They can be written as follows:
CSC (4) = {M (a, b), M (a, c), M (a, d), M (b, c), M (b, d), M(c, d)}.
The second concept to be introduced is *the general condition matrix* for a set of wires which can be defined as follows: *General Condition Matrix* for an *n* wire channel is a matrix which has a number of rows equal to the number of possible combinations of transitions on wires, i.e. *n!*, and a number of columns equal to the number of conditions which identify each combination, i.e. $\binom{n}{2}$. Each row represents the set of conditions that describe a particular combination of transitions. Figure (5) shows how the mutex function is used to form the general condition matrix for a four wire phase encoding scheme.

$$GC(4) = \begin{bmatrix} M1(A,B) & M1(A,C) & M1(A,D) & M1(B,C) & M1(B,D) & M1(C,D) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ M24(A,B) & \cdots & \cdots & \cdots & \cdots & M24(C,D) \end{bmatrix}$$

Fig.5: General Condition Matrix for Four Wire Phase Encoded Channel

Thus the general condition (GC) matrix for a four wire channel can be written as follows:

$$GC(4) = \begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 0 \\
1 & 1 & 1 & 0 & 1 & 1 \\
1 & 1 & 1 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 1 \\
0 & 1 & 1 & 1 & 1 & 1 \\
0 & 1 & 1 & 1 & 1 & 0 \\
1 & 0 & 1 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 1 \\
0 & 1 & 0 & 1 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 1
\end{bmatrix}$$

The first row [1 1 1 1 1 1], for example, represents the combination (abcd).

The third concept is *a Condition Matrix (CM)* which can be formed using any subset of the complete condition set. It has a number of columns equal to the number of conditions and a number of rows which is equal to the number of combinations of transitions (information symbol) that can be represented by these conditions.

A condition matrix which has R rows and C columns is said to be a *complete state matrix* if and only if $R >= 2^C$. The set of conditions whose condition matrix is a complete state matrix is said to be a *complete state set*. For each possible conditions code in a complete state set of conditions, there is a combination of transitions that represents that code.

As can be noticed CG (4) is not a complete state condition matrix. However there are some subsets of these conditions whose condition matrix is a complete state matrix such as set (a), where: set (a) = {(M (a, b), M (a, c), M (a, d)}.

The condition matrix of set (a) is shown below:

$$CM(\text{set a}) = \begin{bmatrix}
M1(a,c) & M1(a,d) & M1(a,b) \\
\vdots & \vdots & \vdots \\
M8(a,c) & M8(a,d) & M8(a,b)
\end{bmatrix} = \begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 1 \\
0 & 1 & 0 \\
0 & 1 & 1 \\
1 & 0 & 0 \\
1 & 0 & 0 \\
1 & 0 & 1 \\
1 & 1 & 0 \\
1 & 1 & 1
\end{bmatrix}$$

Another concept to be defined is the *Condition Graph*. A condition graph for a set of conditions can be obtained by representing each wire by a letter, and connecting each two wires whose signal transitions are to be compared.
Each condition set has its own condition graph. The condition graph of the four wire complete set of conditions is shown in figure 6.

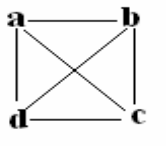


Fig.6: Complete Condition Graph for Four Wires

It should be pointed out here that the condition graph for a complete state set of conditions does not have any loops. Figure 7 shows the condition graph of two set of conditions,
Set (a) and set (b), where: Set (a) = {M (a, b), M (a, c), M (a, d)}.
Set (b) = {M (a, b), M (a, c), M (b, c)}.

Fig.7: Condition Graphs: (1) set (a) ;(2) set (b)

A complete state condition matrix can be obtained using the set (a) .This is not possible for the set (b) as its condition graph has a loop. This is illustrated in table 1. The first three columns of table 1a show the condition codes; the forth one shows the possible combinations which can be formed using the set (a) of conditions. Table 1b shows the condition codes and the corresponding combination which are based on the set (b) of conditions.

Conditions Codes			Set (a)
M(a, b)	M(a, c)	M(a, d)	
0	0	0	bcda
0	0	1	bcad
0	1	0	bdac
0	1	1	bacd
1	0	0	cdab
1	0	1	cabd
1	1	0	dacb
1	1	1	abcd

(a)

Conditions Codes			Set(b)
M(a, b)	M(a, c)	M(b, c)	
0	0	0	cbad
0	0	1	bcad
0	1	0	Impossible
0	1	1	bacd
1	0	0	cabd
1	0	1	Impossible
1	1	0	acbd
1	1	1	abcd

(b)

Table 1: Conditions Combinations Mapping

As can be seen using set (a) eight possible combinations can be obtained, each of which corresponds to a unique set of conditions. In the case of set (b), it is not possible to represent each condition code by a combination. For example, consider the code (010) which means
(a < b), (a > c) and (b < c). There is no combination of transitions that can be represented by this condition code. Therefore only set (a) can be considered a complete state set.
The condition graph can be employed as a tool to check whether or not a certain set of conditions is a complete state set.
An essential concept in this article is the concept of *Clusters*. A cluster is a group of combinations which can be generated from one another by swapping any two neighbouring transitions on any two spatially adjacent wires. A cluster can contain any number of combinations. To clarify, let us consider a special set of conditions which describe the transition order between non-adjacent wires. Let us denote this set Non-Adjacent wire condition set (NA). For example for 4 wire channel:
NA (4) = {M (a, c), M (a, d), M (b, d)}.
All the combinations in one cluster have the same NA conditions values. E.g. for cluster one in figure 8, all combination have their NA set equal to {0, 0, 0}.
Figure 8 shows how the different combinations are grouped into clusters. For example, the fifth bubble contains {acdb, adcb, adcb}. The second member of this cluster is generated from the first one by swapping d and c. The third combination is generated from the second one by swapping b and c. It is not possible to generate any new combination from the third member without violating the relationship between transitions on non-adjacent wires. The combinations which are generated from each other are connected with lines. It should also be noted that the number of combinations in each cluster is not constant.
The number of different clusters which can be formed depends on the number of wires, e.g. for a four wire phase encoded channel, eight different clusters can be obtained as shown in figure 8.

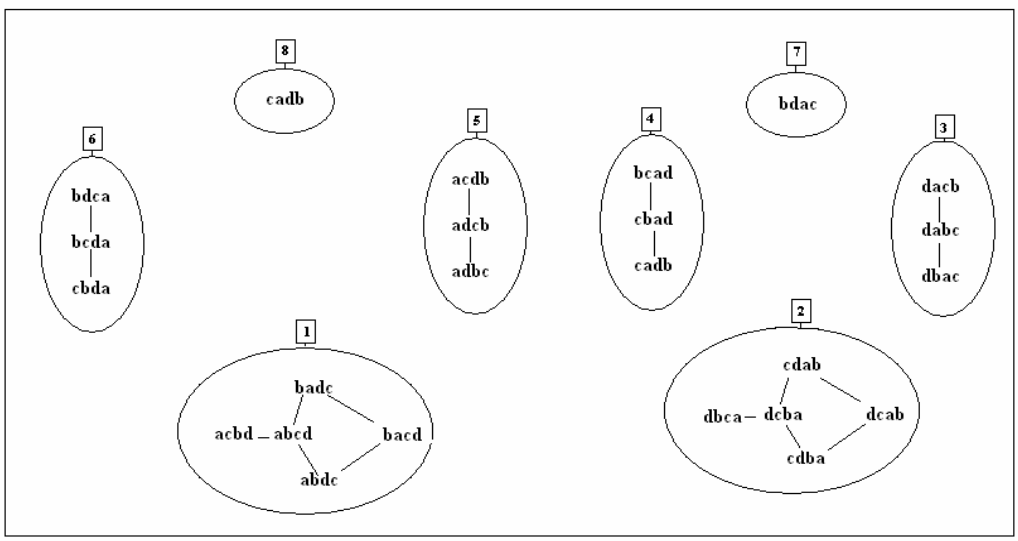


Fig. 8: Cluster Concept

The last term to be defined is the *Cluster Condition Matrix (CCM)* which can be defined as a matrix which has a number of rows equal to the number of clusters, and a number of columns equal to the number of conditions in the Non- Adjacent wire condition set.

The cluster condition matrix for a four wire phase encoded channel can be written as follows:

$$CCM(4) = \begin{bmatrix} M1(a,c) & M1(a,d) & M1(b,d) \\ \vdots & \vdots & \vdots \\ M8(a,c) & M8(a,d) & M8(b,d) \end{bmatrix}$$

The number of rows is eight which is equal to the number of clusters. The number of columns is three which is equal the number of conditions that describe non-adjacent wires.

4. Fault Tolerance Techniques

In this section three fault tolerance techniques for phase encoding transmission protocol will be introduced. The first one is based on the traditional theory of concurrent error detection and correction [26]. The second one is based on the cluster concept. The essence of the third technique is to encode each data bit as phase between signals on two particular non-adjacent wires. This allows the detection or detection and correction of type 2 faults.

4.1. Cluster-based Concurrent Error Detection (CCED) or Detection and Correction (CCEC) Techniques

The essence of this method is to map the normal output vector space of a system onto an extended code space such that only a subset of the code space represents valid information. This mapping can be obtained by adding extra bits, which are called check bits, to the data word to form a codeword which has useful error detection or detection and correction properties. Figure 9 shows how this mapping is performed on two bit data word in order to obtain codewords which have one bit error detection property. For example, if the data word (10) is to be sent, the encoder at the sender side adds the parity bit (1) to the data word to form the code word (101). If error occurred during transmission and it has been a one bit error, i.e. the codeword would turn into (001, 100 or 111), and then the decoder at the receiver side would detect a non-code word and will generate an error signal.

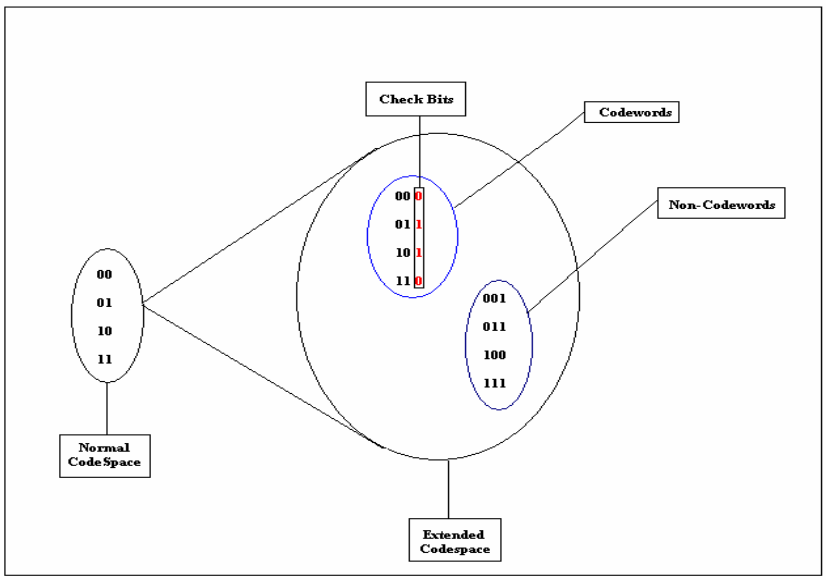


Fig.9: Error Correction Concept

If this technique is to be employed on phase encoded channel under consideration, in order to detect or detect and correct faults introduced in section 3, the phase conversion problem must be mapped into codeword alteration problem. Our simulations showed that the occurrence of type 1 faults is more frequent than type 2 and/or type 3 ones, therefore it was decided that the phase encoder should be designed in such a way that phase conversion between any two signals on adjacent wires should only cause one bit error. This reduces the effect of the crosstalk noise on the data integrity. For example, let us study the case of a 4 wire phase encoded channel. Assume that the data word (0000) was phase encoded as (abcd) and during its transmission was altered to (abdc). Here we have two possibilities. The first one is when the hamming distance between the two original codes is equal to one, e.g. the code of (abdc) represents (0001), (0010), (0100) or (1000). The second possibility is when the hamming distance between the two codes is more than one, e.g. (abdc) code is (0111). As can be clearly noticed type 1 faults causes one bit error in the first case and 3 bit error in the second case. A clever mapping between the data word and the combinations can improve the reliability of the channel. This mapping can be achieved using the cluster concept as shown in figure 10.

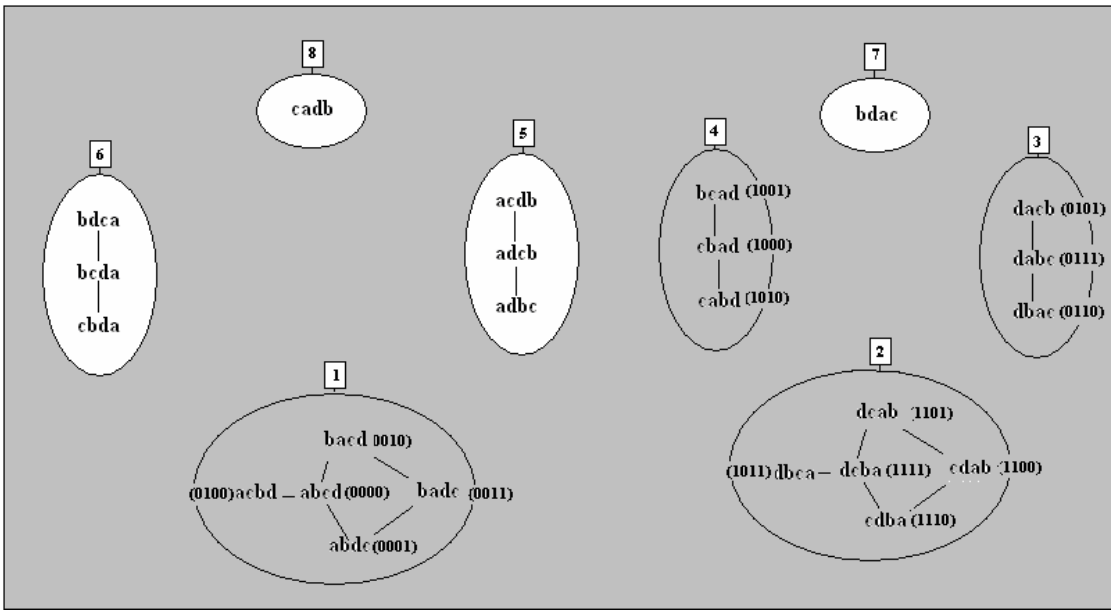


Fig. 10: Cluster Mapping for CCED

In this case every two combinations which belong to the same cluster and generated from one another are assigned two codes whose hamming distance is 1. Concurrent testing techniques can now be applied. This can be achieved by adding extra circuitry to the original design. Figure 11 shows general block structure for error correction (a) and error detection (b) for M wire phase encoded channel.

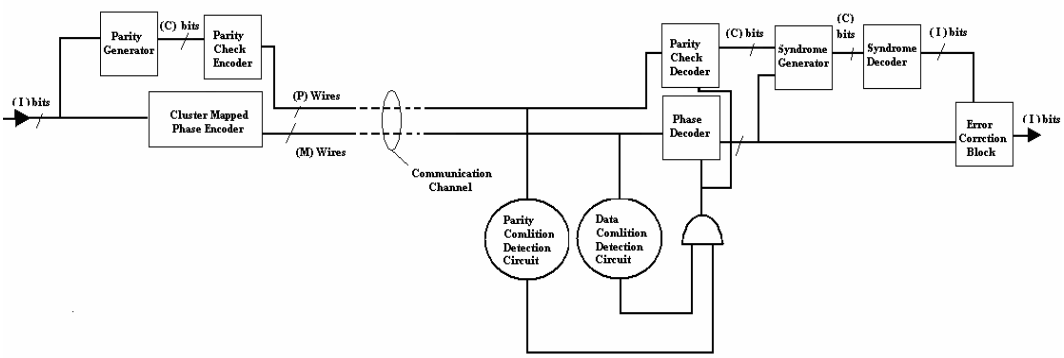


Fig.11.a: General Concurrent Error Correction Scheme for Phase Encoded Channel

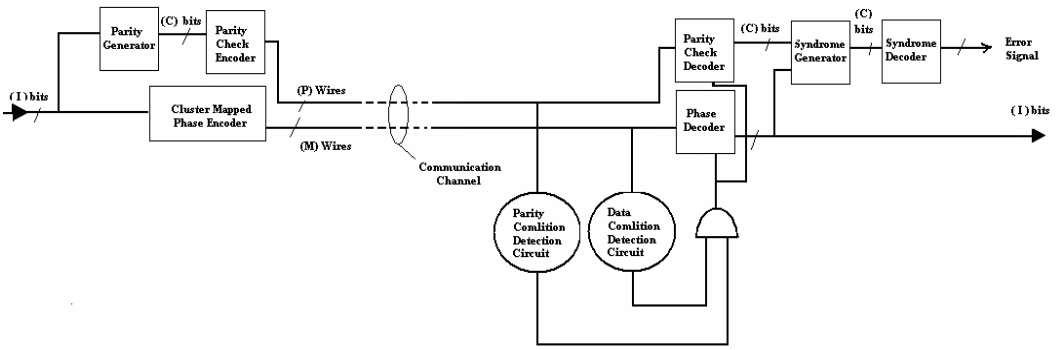


Fig.11.b: General Concurrent Error Detection Scheme for Phase Encoded Channel

At the sender side the data bits to be encoded are given to the input of a clusters mapped phase encoder, and to the inputs of a parity generator that is organized as a binary XOR tree. The latter computes the check bits which are then applied to the input of a parity check encoder. Data and check bits are sent on separate wires but at the same time, therefore both data and parity encoder should be synchronised. At the receiver side, the data bits are given to the inputs of the phase decoder and the check bits are given to the input of the parity decoder. Both data and parity encoder are activated at the same time when the completion detection signals are generated. The syndrome generator outputs the syndrome patterns which are applied to the syndrome decoder. The latter generates the error signal. In the case of error correction another block is added to the scheme called the error correction block. The completion detection circuits can be built using C-elements [15]. The complexity and the structure of parity completion detection circuit depend on the type of parity encoder and the number of parity bits. For example, consider the case of single bit error detection technique applied to a 4 wire phase encoded channel. One parity bit is needed in this case to detect single type 1 error. The parity completion detection circuit consists of a mutex element. The sender consists of three blocks, namely: four wire cluster mapped phase encoder, parity generator and parity encoder. The phase encoder has four inputs and four outputs. It encodes an input data signal into sequence of transitions at its outputs. This encoding obeys the cluster mapping concept shown in table 2. The encoder consists of two blocks. The first one is one hot encoder which has 16 outputs, each of which corresponds to a particular combination at its four inputs. Those outputs are given to a delay generator which translates them into sequences of transitions on its four outputs. The parity generator is a 4 input XOR gate. Phase coding is used to encode the parity bits because of its robustness to transient errors; therefore two wires are needed to send the parity bits. The maximum possible value is assigned to the phase between the transitions on the two parity check wires to avoid phase conversion. In order to minimise crosstalk between the parity wires and the data wires, two methods can be used. The first one consists of placing the two parity wires on both sides of the channel than swapping their places an even number of times.

Another solution is to introduce grounded shielding wires between the parity and data wires. Both techniques were simulated and compared as will be shown in the next section. The decoder consists of the following blocks, namely: completion detection circuitry, phase decoder, parity check decoder and error detection circuitry. Completion detection circuitry is built using C-elements. The parity check decoder consists of three blocks. The first one is an array of 6 mutex elements whose outputs are unique to each sequence of transitions. These outputs are given to a condition decoder block which generates the original data. The parity check decoder consists of a mutex element. The four outputs of the condition decoder are XORed together then compared with the parity check matrix using XNOR gate that generates the error signal.

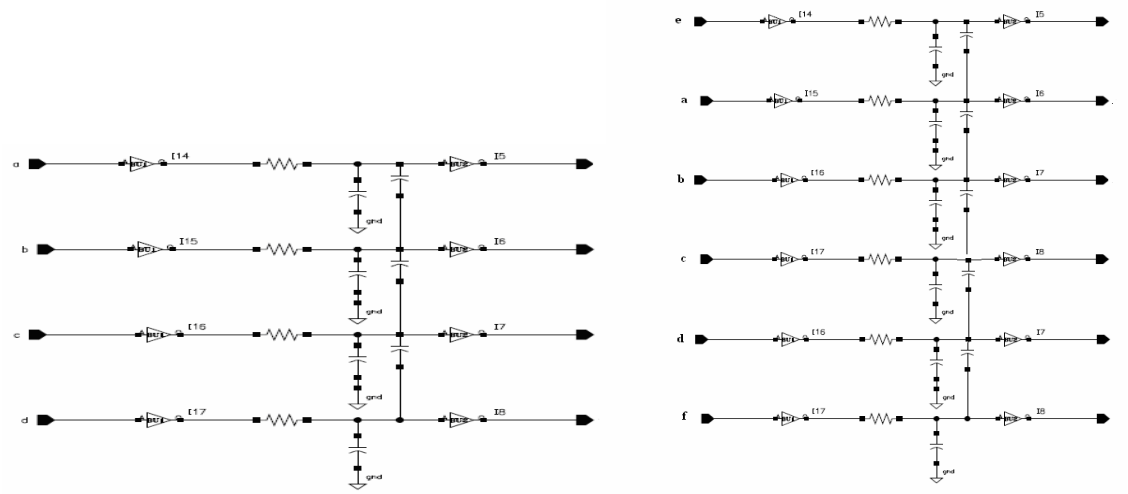
Combinations		Codes	
Data Wires	**Parity Wires**	**Data**	**Parity**
abcd	fe	0000	0
abdc	ef	0001	1
bacd	ef	0010	1
badc	fe	0011	0
acbd	ef	0100	1
dacb	fe	0101	0
dbac	fe	0110	0
dabc	ef	0111	1
cbad	ef	1000	1
bcad	fe	1001	0
cabd	fe	1010	0
dbca	ef	1011	1
cdab	fe	1100	0
dcab	ef	1101	1
cdba	ef	1110	1
dcba	fe	1111	0

Table 2: Cluster Mapping for Four Wires Phase Encoded Channel

Simulations

To determine the efficiency of this technique in detecting type 1 faults, the sender and the receiver have been designed in CADENCE and simulated using AMS CMOS 0.35μm process. It was assumed that wires do not change their special order. The simulation was performed in three stages. The aim of the first stage was to check the functionality of the design. The outputs of the sender were connected directly to the inputs of the receiver, than all possible 16 combinations (0000 to 1111) were applied to the encoder. The decoder was able to regenerate the original data in all cases. The aim of the second stage is to establish if the circuit would generate an error signal when a phase conversion occurs between two signals on adjacent wires (type 1 faults). A number of faulty sequences (e.g. cabd, ef) were applied to the decoder, each of which has single phase conversion between signals on two neighbouring wires. The circuit

was able to detect the error in all cases.



(a) **(b)**

Fig. 12: Phase Encoded Communication Channel First Order Model

The aim of the third stage was to study the impact of the addition of the two parity check wires on crosstalk noise, therefore three experiments were undertaken. In the first one we used a 4 wire communication channel model where the RC non-idealities of the wires are taken into account (figure 12.a). In the second one we used a 6-wires communication channel model (see figure 12.b).The parity check wires were placed on both sides of the channel, each on one side, then their places were swapped four times along the channel. In the third one the parity wires were also placed on the sides of the channel, and a grounded shielding wire was placed between each parity wire and the neighbouring wire. The channel length was chosen to be (2mm). All 24 combinations were then applied to the inputs of the channels. The results were as follows:

No. of Phase Conversion Errors		
No Parity Wires	Two rerouted Parity Wires	Two Parity Wires and Two Shielding wires
6	12	7

Table 3: The Impact of the Addition of Parity Wires on Crosstalk Errors

As can be noticed, the shielding wires technique is more efficient in preventing crosstalk between parity and data wires. Another approach to this problem is to route the parity wires in a different path, hence the addition of parity wires does not worsen the channel reliability characteristic.

4.2 Cluster-based Partial Coding Technique (CPC)

Although CCED method improves the reliability of the channel, it has large area overheads. CPC is another approach based on the cluster concept; it has the same fault tolerance ability but requires less area overheads. The idea of this method is to encode the data by means of phase

difference between transitions of signals on non-adjacent wires so that the phase between any two signal on neighbouring wires does not carry any information. The theory behind this method is that the crosstalk between signals on non-adjacent wires is less than it is between signals on adjacent wires because of the fact that capacitive coupling effects attenuate with distance. This partial order coding masks type 1 fault and does not require any additional hardware. However less number of bits can be sent as not all combinations can be used; In fact the number of states in this case is equal to the number of clusters.

For example, consider the case of a four wire channel. Eight clusters can be obtained; therefore three bits of information can be sent. It should be pointed out here that the number of cluster that can be formed depends only on the number of wires in the channel (e.g. for 5 wire channel 42 clusters can be formed). Each cluster is assigned a three bit digital code as shown in figure 13.The digital codes are assigned to each cluster in such a way that type 2 faults cause only 1 bit error .This can be done using the cluster condition matrix. As explained previously each row of this matrix represents the unique set of conditions that distinguish the combinations in a cluster from those in another. Two clusters which have a hamming distance between their cluster conditions set value equal one are assigned digital codes whose hamming distance is one. Figure 13 shows how this mapping is done for a four wire channel. One combination from each cluster is used to design the coding circuitry as shown in table 4. This table has four columns. The first one represents the number of clusters as shown in figure 13. The second column is the set of cluster conditions that characterise each cluster (see section 2); the third column contains the combinations which are chosen to design the sender and the receiver. The last column shows the digital code given to each combination. All the chosen combinations have the maximum possible phase difference between transitions on non-adjacent wires. The latter measure was proved to reduce the number of phase conversion faults caused by crosstalk as will be shown in the second stage of the simulation (see figure 15).

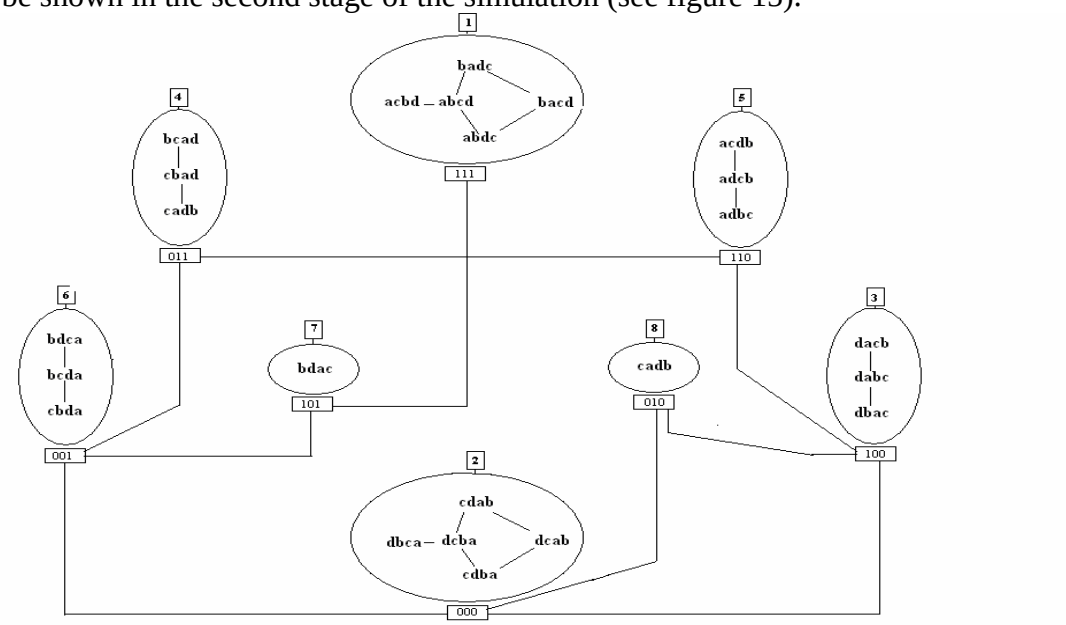


Fig.13: Cluster Mapping for CM

Cluster Number	Cluster Condition Set	Combination	Code
2	000	dcba	000
6	001	bcda	001
8	010	cadb	010
4	011	cbad	011
3	100	dabc	100
7	101	bdac	101
5	110	adcb	110
1	111	abcd	111

Table 4: Cluster Partial Order Coding for Four Wires Phase Encoded Channel

Simulations

The sender and receiver have been designed in CADENCE and simulated using AMS CMOS 0.35µm process.

It was assumed that wires do not change their special order. The simulations were performed in four stages. The aim of the first stage was to check the functionality of the design. The outputs of the sender were connected directly to the inputs of the receiver than all possible eight combinations (000 to 111) were applied to the encoder. The decoder was able to regenerate the original data in all cases. The aim of the second stage was to study the effect of clustering on crosstalk noise. Five experiments were performed, each of which on a different channel length. The remaining parameters (e.g. coupling capacitance) were identical in the five experiments. The four wire communication channel model described in section 4.1 was used in all of them. All 24 combinations were applied on the channel inputs in each case, and then the number of phase conversion errors was calculated. Figure 14 shows the results of this simulation. As can be clearly seen by using CPC technique the number of errors caused by crosstalk was significantly reduced, especially for long wires.

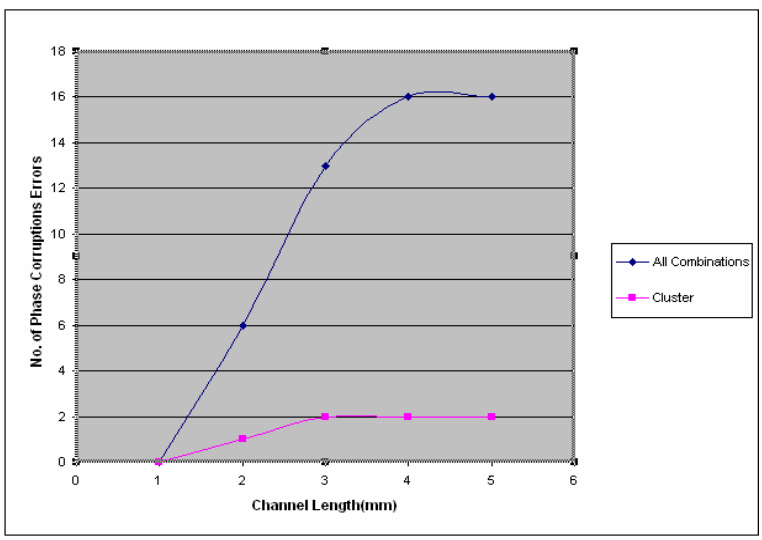


Fig. 14: The Effect of Cluster-based Partial Coding on Crosstalk Faults

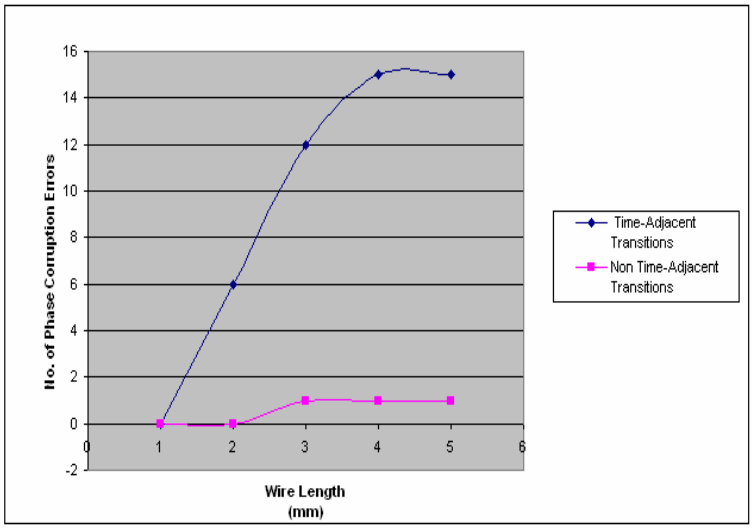


Fig. 15: The Effect of Phase Difference on Crosstalk Noise

In order to examine the effect of signals phase difference on crosstalk noise, the results of the previous simulation were analysed from a different prospective. In each experiment we calculated the number of phase conversion errors between transitions which have minimum phase difference (500ps in this case) and between transitions whose phase difference is equal to 1ns or 1.5ns. For example consider the combination *acbd.* The pairs of transitions on (a, c), (c, b) and (b, d) have the minimum possible phase, whereas those on (a, b), (c, d) and (a, d) have larger phases differences. The first group was called time-adjacent transitions and the second group was called non time-adjacent transitions. As can be noticed from the results shown in figure 15, a considerable reduction in phase conversion errors can be obtained by increasing the phase difference between transitions on wires. This fact was exploited in the design of the scheme explained previously. The aim of the third stage was to study the effect of this technique on the channel performance. In other words to establish if the channel speed would hence increased when cluster-based partial coding was used. The 24 combinations were applied at the channel inputs at several frequencies. The order and the phase of the transitions were checked at the channel outputs. The results are shown in figure16.

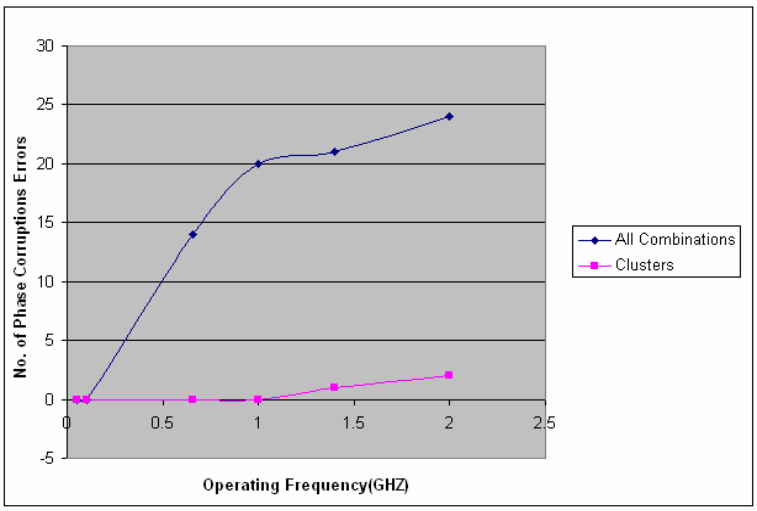


Fig. 16: The Effect of Cluster-based Partial Coding on phase encoded channel performance

As can be seen the number of phase conversion errors become prohibitively large at frequencies higher than 0.5 GHz if all combinations are used to encode the data.
However the cluster-based partial coding technique allows the use of higher frequencies without degrading the channel reliability.
The maximum frequency of phase encoded channels as calculated in [15] depends on several factors, namely: the speed of the receiver, i.e. the time needed to latch a data symbol and be ready for the next; the number of wires and layout process issues.
It was found that for a four wire channel the maximum frequency is 0.66 GHz. The parameters that were used in the calculations correspond to the 0.6µm technology. For more advanced technology, the effect of cluster-based partial coding in achieving higher speed communication would be more evident.

4.3. Direct Mapping Technique (DM)

This method consists of encoding each data bit as phase difference between signals on two particular non-adjacent wires as illustrated in table (5).

Data	Wires
I1	a & c
I2	a & d
I3	b & d

Table 5: Direct Mapping for 4 wire phase Encoded Channel

Direct mapping masks type 1 faults as phase between adjacent wires does not carry any information. It also allows the detection and/or the correction of type 2 faults. This technique will be implemented on a four wire phase encoded channel in order to detect single type 2 error.

The first step is to choose a complete state subset. Let us denote it set (c).
Set (c) = {M (a, c), M (a, d), M (b, d)}.

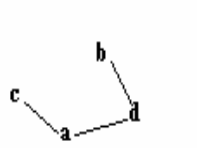


Fig.17: Condition Graph for Set (c)

Those three conditions form a complete state subset as their conditions graph has no loops (see figure 17).Then the conditions matrix must be formed which in this case has 3 columns and 8 rows. The condition matrix for set (c) is shown below:

$$CM(\text{set c}) = \begin{bmatrix} 0&0&0 \\ 0&0&1 \\ 0&1&0 \\ 0&1&1 \\ 1&0&0 \\ 1&0&1 \\ 1&1&0 \\ 1&1&1 \end{bmatrix}$$

The third step is to find a combination of transitions which satisfies each element of the condition matrix and write it down in a table called the mapping table as illustrated below.

Conditions Codes			Set (c)
M(a, c)	M(a, d)	M(b, d)	
0	0	0	dcba
0	0	1	bdca
0	1	0	cadb
0	1	1	cabd
1	0	0	dbac
1	0	1	bdac
1	1	0	acdb
1	1	1	abcd

Table 6: Mapping Table

This mapping table will be used to design the first block of the phase encoder.

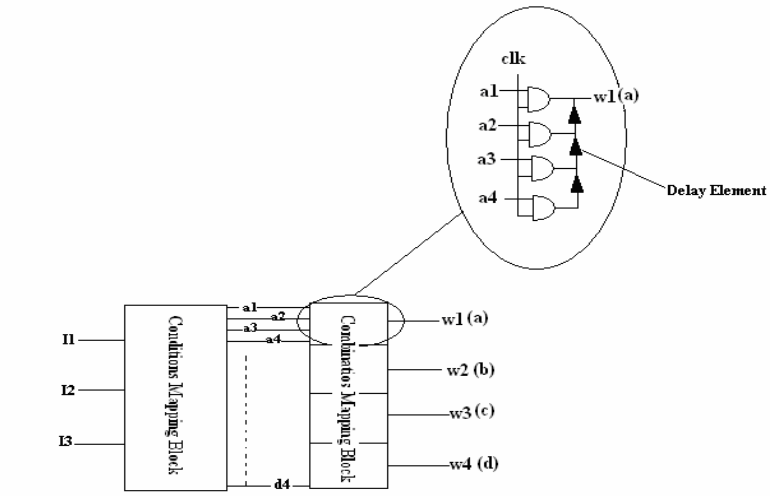


Fig. 18: DM Phase Encoder for Four Wire Phase Encoded Channel

The phase encoder consists of two blocks, namely: conditions mapping block and combinations mapping block. The conditions mapping block has three data inputs (I1, I2, I3), and 16 outputs (a1, a2, a3, a4 ….d4), which represent the phase of the transition on each wire, e.g. if (a1=1) that means the transition on wire *a* is the first one. The mapping table from step 3 is used to build the truth table of this function.

Table 7 shows the truth table of the first 4 outputs, the first three columns represent the data, and the conditions that they are mapped to. The forth column represents the combinations which can be obtained from the mapping table.

I1	I2	I3	Combinations	A1	A2	A3	A4
M(a, c)	M(a, d)	M(b, d)					
0	0	0	dcba	0	0	0	1
0	0	1	bdca	0	0	0	1
0	1	0	cadb	0	1	0	0
0	1	1	cabd	0	1	0	0
1	0	0	dbac	0	0	1	0
1	0	1	bdac	0	0	1	0
1	1	0	acdb	1	0	0	0
1	1	1	acbd	1	0	0	0

Table 7: Truth Table for (a1, a2, a3, a4)

Using logic minimisation the following logic equations are obtained.

$A4 = \overline{I1}\ \overline{I2}$

$A3 = I1\ \overline{I2}$

$A2 = \overline{I1}\ I2$

$A1 = I1\ I2$

The logic equations of the rest of the outputs can be derived in the same fashion.
The function of the combination mapping block is to map the outputs of the first block into phase difference between transitions on wires, it has four identical sub-blocks, each of which corresponds to a particular wire. The structure of the first sub-block is shown in figure 19.
The phase decoder consists of three mutex elements, each of which translates the order of transition between its two wires into a data bit, and a completion detection circuit (see figure 19).

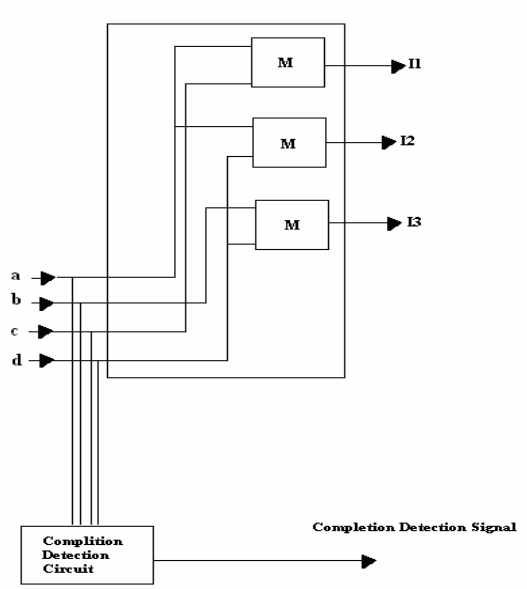


Fig.19. Four Wire DM Phase Decoder

The DM method simplifies the design of the phase decoder significantly as there is no need for the decoding array used in the initial design in [15]. This decreases the area overheads. In order to quantify this reduction the number of 2-inputs gates needed to implement the receiver for each method has been calculated and compared with its counterpart in the original design in [15]. The results are shown in table 8.
The checking circuitry is designed in the same manner described in section 4.1. Figure 20 shows a block diagram of the scheme.

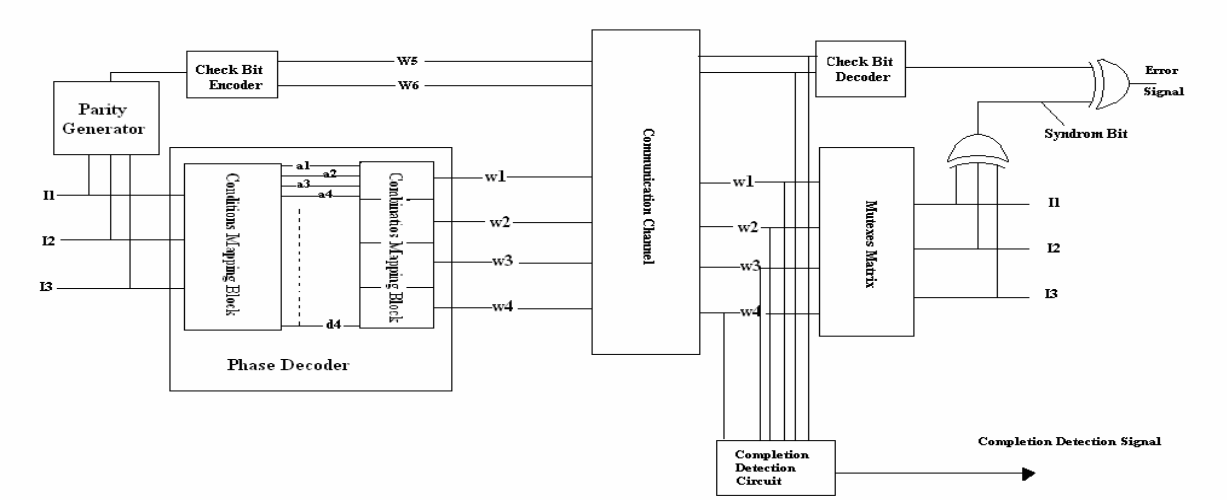


Fig.20: Single type 2 Faults Error Detection DM Block Diagram for 4-Wire Phase Encoded Channel

Simulations

DM circuitry has been designed in CADENCE and simulated using AMS CMOS 0.35µm process. Its functionality has been verified by applying all possible combinations on the encoder and receiving them correctly at the decoder. The efficiency of this scheme in detecting type 2 faults has also been verified. In order to establish the effect of the addition of parity check wires on channel reliability, the same experiment as described in section 4.1 (the third stage simulation) was conducted, however this time only type 2 faults were taken into considerations. The results are shown in table 8.

No. of Type 2 Faults		
No Parity Wires	Two rerouted Parity Wires	Two Parity Wires and Two Shielding wires
1	**4**	**2**

Table 8: The Impact of the Addition of Parity Wires on Crosstalk Type 2 Errors

As can be seen, no significant impact on channel reliability is caused by the addition of the two parity check wires. This technique has the same effect on the phase encoded channel performance as the CPC method.

Table 9 shows a comparison between the three mentioned methods in terms of information capacity (column 3 and 4), extra hardware (column 5), fault tolerance capability (column 5), and its effect on performance (column 7) and on the design (column 8). The first column represents the number of wires in the phase encoded channel. The second column represents the method of fault tolerant design.

NO. of Wires	Method	Symbols	No of bits	Additional hardware for Testing circuitry	Fault Tolerance Capability	Effect On Performance	Effect on The Design
4	CCED	24	4	yes	Type 1 Detection	None	None
	CPC	8	3	no	Type 1 Masking	Three Times Improvement is possible	None
	DM	8	3	yes	Type 1 Masking Type 2 Detection	Three Times Improvement is possible	The receiver requires 90% less area.
5	CCED	120	6	yes	Type 1 Detection	None	None
	CPC	42	5	no	Type 1 Masking	Improves Performance	None
	DM	16	4	yes	Type 1 Masking Type 2 Detection	Improves Performance	The receiver requires 97 % less area
6	CCED	720	9	yes	Type 1 Detection	None	None
	CPC	258	8	no	Type 1 Masking	Improves Performance	None
	DM	32	5	yes	Type 1 Masking Type 2 Detection	Improves Performance	The receiver requires 99% less area

Table 9: The Characteristics of CCED, CPC and DM Techniques

5. Conclusions

The paper has analysed the effect of crosstalk on the behaviour of phase encoded communication method proposed by [15]. Such a method is self-timed and can be combined with the already known delay-insensitive encoding [13]. The latter is known to be resistant to delay variability, whereas the former enjoys extra information capacity at the cost of additional (relative) timing requirements. In fact, the phase encoded data transmission can be used as a way to communicate on small to medium length interconnects. When combined with delay-insensitive schemes for longer interconnects, our method would offer greater flexibility to the design of complex systems on a chip.

The paper has shown that phase encoded data integrity is put at risk by crosstalk-related delays. The phase between transitions deteriorates in long parallel wires due to coupling capacitance which depends on wire length and the operating frequency. This deterioration may lead to errors i.e. phase conversion, hence to system failure. The simulation showed that crosstalk related errors become prohibitively large for wire longer than 2 mm and at frequencies higher than 0.5 GHz. Therefore faults tolerant techniques are a necessity in order to allow the use of this communication protocol reliably. A fault model has been introduced to formalise the crosstalk problem. Three methods have also been presented. All of them are based on partial order coding concept. The CCED technique allows the detection of type 1 faults. The CPC method masks type 1 faults. It also leads to a significant improvement in the channel performance. This compensates for the reduction in the number of symbols that can be sent. The DM technique allows the detection of type 2 faults. It also masks type 1 faults. Improvement in channel speed is also possible. The DM method significantly simplifies the design of the phase decoder and reduces its area, which facilitates the implementation of phase encoded transmission protocol on a higher number of wires.

References

[1] F. Caignet, S. Delmas-Bendhia, and E. Sicard, "The challenge of signal integrity in deep-sub micrometer CMOS technology," *Proc. IEEE*, vol. 89, pp. 490–504, 2001.

[2] D. Sylvester and C. Hu, "Analytical modelling and characterization of deep-sub micrometer interconnect," *Proc. IEEE*, vol. 89, pp. 634–664, May. 2001.

[3] D. Pamunuwa, L.R. Zheng, and H. Tenhunen, "Maximizing throughput over parallel wire structures in the deep sub micrometer regime," *IEEE Trans. VLSI Systems*, vol. 11, pp. 224–243, Apr. 2003.

[4] N. A Kurd, '' Multi-GHz Clocking Schemes for Intel Pentium 4 Microprocessors '' Proc. ISSCC 2001 Feb 2001 pp 404-405.

[5] Semiconductor Industry Association, International Technology Roadmap for Semiconductors (ITRS) 2003, http://public.itrs.net.

[6] Semiconductor Industry Association, International Technology Roadmap for Semiconductors (ITRS) 2005, http://public.itrs.net

[7] D. Chapiro, Globally-Asynchronous Locally-Synchronous Systems, Ph.D. Thesis, Stanford University, 1984.

[8] J. Muttersbach, T. Villiger, H. Kaeslin, N. Felber, and W. Fichtner, "Globally-Asynchronous Locally-Synchronous Architectures to Simplify the Design of On-Chip Systems," *IEEE Inter. ASIC/SOC Conf.*, pp. 317-321, 1999.

[9] V. Akella, N. H. Vaidya and G. R. Redinbo, "Limitations of VLSI implementations of delay-insensitive codes", Proc. 26th Int. Symp. On Fault-Tolerant Computing, Sendai, June 1996, pp. 208-217.

[10] J. Bainbridge and S. Furber, "Delay Insensitive System on-Chip Interconnect using 1-of-4 Data Encoding", In Proceedings. Seventh International Symposium on Asynchronous Circuits and Systems. ASYNC 2001, pages 118–126, 2001.

[11] J. Bainbridge, W.B. Toms, D.A. Edwards, S.B. Furber, "Delay-Insensitive, Point-to-Point Interconnect using m-of-n Codes", Proc. IEEE Ninth International Symposium on Asynchronous Circuits and Systems (ASYNC'03).

[12] J. Bainbridge and S. Furber., "CHAIN: A Delay-Insensitive Chip Area Interconnect", IEEE Micro, Sep/Oct 2002, pp16-23.

[13] T. Verhoeff, "Delay-insensitive codes- an overview". Distributed Computing, 3(1):1-8, 1988.

[14] Ajanta Chakraborty and Mark R. Greenstreet,"Efficient self-timed interfaces for crossing clock domains", In *Proceedings. 9th International Symposium on Asynchronous Circuits and Systems* May 2003.

[15] C. D'Alessandro, D. Shang, A. Bystrov, A. Yakovlev and Oleg Maevsky, "Multiple-Rail Phase-Encoding for NoC", In Proc. ASYNC'06, IEEE, pages 107–116, March 2006.

[16] M. Nicolaidis, Eric Dupont and Peter Rohr," Embedded robustness IPs", In Proceedings of the 2002 Design, Automation and Test in Europe Conference and Exhibition (DATE'02), 2002.

[17] M. Nicolaidis,"Time redundancy based soft-error tolerance to rescue nanometre technologies", In 17th IEEE VLSI Test Symposium, April 1999.

[18] C. Molnar and I. Jones," Simple circuits that work for complicated reasons", In Proceedings Sixth International Symposium on Asynchronous Circuits and Systems, volume 1, pages 138–149, IEEE CS, April 2000.

[19] S. Muddu, "Repeater and Interconnect Strategies for High Performance Physical Designs," Proc. XI Brazilian Symp. Integrated Circuit Design, IEEE CS Press, 1998, pp. 226-231.

[20] D. Pamunuwa and H. Tenhunen, "Repeater Insertion to Minimise Delay in Coupled Interconnects," Proc. 14th Int'l Conf. VLSI Design, IEEE CS Press, 2001, pp. 513-517.

[21] L.D. Huang, H.M. Chen, and D.F. Wong, "Global Wire Bus Configuration with Minimum Delay Uncertainty," Proc. Conf. Design, Automation and Test in Europe (DATE 03), IEEE CS Press, 2003, pp. 50-55.

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

[23] B. Victor and K. Kreutzer, "Bus Encoding to Prevent Crosstalk Delay," IEEE/ACM Int'l Conf. Computer-Aide Design (ICCAD 01), IEEE Press, 2001, pp. 57-63.

[24] C. Duan and A. Tirummala, "Analysis and Avoidance of Cross-Talk in On-Chip Buses," Proc. Hot Interconnects 9 (HOTI 01), IEEE CS Press, 2001, pp. 133-138.

[25] D. Pamunuwa, L.R. Zheng and H. Tenhunen, " Maximising Troughput Over Parallel Wire structures in Deep Submicron Regime", IEEE Trans. VLSI Systems, vol 11, no 2, Apr. 2003, pp. 224-243.

[26] G. Russell and Ian L. Sayers, Advanced Simulation and Test Methodologies for VLSI Design, London, 1989.