ST-CAC: a low-cost crosstalk avoidance coding mechanism based on three-valued numerical system

Zahra Shirmohammadi Ata Khorami Martin Eugenio Omana

Abstract
Appearances of specific transition patterns during data transfer in bus lines of modern high-performance computing systems, such as communicating structures of accelerators for deep convolutional neural networks, commercial Network on Chips, and memories, can lead to crosstalk faults. With the shrinkage of technology size, crosstalk faults occurrence boosts and leads to degradation of reliability and performance, as well as the increasing power consumption of lines. One effective way to alleviate crosstalk faults is to avoid the appearance of these specific transition patterns by using numerical-based crosstalk avoidance codes (CACs). However, a serious problem with numerical-based CACs is their overheads in terms of required additional bus lines for representing code words. To solve this problem, in this paper we present a novel CAC that is based on the use of three symbols (three-value) to represent the code words in the bus lines, rather than classical binary CACs based on binary, i.e., 0 and 1 symbols. Our proposed CAC, named summation-based tri-value crosstalk avoidance code (ST-CAC), reduces the worst-case delay in bus lines with respect to binary CACs, and it can efficiently be applied to any arbitrary channel width of lines. The use of three symbols to represent code words in ST-CAC enables to increase the number of code words of a numerical system without increasing the number of required bus lines significantly. The experimental results show that CACs based on the use of three symbols can reduce the number of additional lines compared to binary CACs by 33%. Moreover, we show in the paper, that the delay of wires in the presence our ST-CAC can reduce by 33% with respect to state-of-the-art binary value CACs.

Keywords Crosstalk avoidance codes (CACs) · Crosstalk faults · Network-on-chip · Reliability · Tri-valued numerical system coding mechanism

1 Introduction

Communications between Processing Elements (PEs) in modern high-performance computing systems such as communicating the structure of accelerators for deep convolutional neural networks [1], commercial Network on Chips (NoCs) [1] and memories [2] is sophisticated in the future. This is due to the overgrowing amounts of PEs on a chip [3]. According to the International Technology Roadmap for Semiconductors (ITRS), the number of (PEs) on a chip will reach 5000 by 2021. These PEs use bus lines of communication channels to exchange data between them [4, 5].

Sending and receiving these data among PEs is done by using communication channels. Each of these communication channels consists of tens of several hundred parallel and adjacent lines. These communication channels among PEs play a key role in the reliability of the data. One of the reliability challenges during the data transfer is crosstalk faults, which can threat the reliability of data traversal in the lines [6].

Crosstalk faults occur due to coupling capacitances between parallel and adjacent lines of communication channels. With the shrinkage of the technology size, lines get closer to each other accelerating the crosstalk faults occurrence. Crosstalk faults cause a glitch and/or timing violations on the affected line called the victim line [6, 7]. The severity of crosstalk faults depends on the transition patterns appearing on the communication channel lines. These patterns may have negative effects on the reliability, performance, and power consumption of systems [7–9]. Three-valued logic mechanisms such as shielding and repeater insertion at a physical level [10, 11] and timing skewing [12] at the transistor level can reduce crosstalk faults. However, they impose high overheads and are not favored by designers [8, 12, 13]. Crosstalk faults tackling mechanisms have been the subject of certain research groups [13–20]. These mechanisms can be categorized into a physical level, transistor level and a logic level of design abstraction. At the physical level, shielding the lines is among the simplest mechanisms that omit crosstalk faults completely [10]. Inserting the repeater between the segments of lines is another physical-level mechanism [11]. With the use of repeater insertion mechanisms, crosstalk fault is reduced by separating the line into several segments and driving each segmented part by inverting or non-inverting buffers. At the transistor level, the delay uncertainty caused by crosstalk coupling can be reduced by skewed transitions [12]. In skewed transitions, simultaneous opposite transitions between adjacent lines are avoided by generating the relative delay.

At the next higher level, logic level, Error Detecting and Correcting Codes (EDCC) [14] and Crosstalk Avoidance Codes (CACs) [13 15–21] have emerged as a promising solution for crosstalk mitigation. EDCCs impose high overheads to chips and their ability to correct and detect error is limited [15, 16]. CACs are not only technology-independent but also reduce crosstalk faults while requiring lower cost than other alternate solutions. CACs can prevent specific transition patterns from different classes of transitions. Coding the data has a lower area overhead than the other mechanisms at physical and transistor levels [17].

Moreover, there are CACs with the ability to correct possible errors; however, their ability to correct and detect is limited [10].

One of the challenges of CACs is the overheads of the codec. Channel partitioning is among mechanisms that can reduce the overheads of CACs codecs [17]. In channel partitioning, links of communication channels are divided into groups and data are transferred in these groups by separate encoder and decoder. However, partition-based mechanisms face with the problem of appearing transitions in the borders of partitioned groups. The other mechanism is the use of numerical systems. Numerical systems can be used efficiently to reduce the overhead of the codec modules. A numerical system is a mathematical notation for representing numbers of a given set by using bases in a consistent manner [17]. In this numerical system, $S_k, \ldots, S_1$ are the bases that show the weights. In numerical system-based CACs, the code word is represented according to the weight of the numerical system's bases. A numerical system plays an important role in the overheads including power consumption, area occupation and critical path of the codec modules, so choosing the proper numerical system can reduce the overheads of encoder and decoder.

Based on the transition patterns that they omit, numerical-based CACs can be categorized into Forbidden Pattern Free (FPF), Forbidden Transition Free (FTF) [15] codes, Forbidden Pattern Free (FPF) [17, 19, 20, 22, 23] codes, Forbidden Overlap Condition (FOC) [13, 21] codes and One Lambda Code (OLC) [13, 24]. Fibonacci-based [13, 15] and non-Fibonacci-based [19, 20] numerical systems have been proposed to generate FPF CACs in the literature. One of the Fibonacci-based numerical systems is the Fibonacci numerical system (referred to as Fibo-CAC hereafter) [15]. Fibo-CAC uses the Fibonacci sequences to generate code words. The other Fibonacci-based numerical system that has been proposed recently is an improved-Fibonacci coding mechanism (referred to as Improved-Fibo-CAC hereafter) [13]. Improved-Fibo-CAC uses the same bases of Fibonacci sequence but the only difference is in the penultimate bit position of bases that is duplicated in comparison with Fibo-CAC numerical system. Penultimate-Subtracted Fibonacci (PS-Fibo) [22] is the other numerical-based CAC that uses non-Fibonacci-based numerical systems to generate FPF CACs [19, 20]. Recently, [24] proposes the OLC coding mechanisms Subtraction-based-Numeral (Sub-Num).

However, state-of-the-art numerical-based CACs use the classical binary values (i.e., 0 and 1) to represent the data in the bus lines. However, since CACs prevent specific transition patterns to occur, they require increasing the number of additional bus lines in order to enable to represent the same original data. It has been shown in [17] that for binary CACs the number of additional bus lines increases with the number of the original number of bus lines.

To address this problem, in this paper we present a novel CAC that is based on the use of three symbols (0, 1 and 2, each one associated to a different voltage value) to represent the data in the bus lines, that will be hereafter referred to as Summation-based Tri-value-based Crosstalk Avoidance Code (ST-CAC). Also, the number of data words that can be represented using the ST-CAC coding mechanism is presented in this paper. We show in the paper that our ST-CAC enables reductions in worst-case propagation delay in bus lines up to 34% with respect to binary value CACs and it can be applied to any arbitrary width of bus lines. This is mainly because the use of three symbols to represent data in ST-CAC enables to increase the number of code words of a numerical system without increasing significantly the number of required bus lines. The experimental results show that ST-CAC can reduce 33% the number of additional bus lines compared to the most recent binary-value CAC. Moreover, we show in the paper, that the area overhead and power consumption required by the encoder and the decoder of our ST-CAC is comparable to that required by the encoder and the decoder of binary CACs.

The key contributions of this paper include the following:

- The main problem of CACs is the overhead of lines. This is due to this fact that with preventing the patterns, more lines are required to represent code words. To reduce the line's overheads in traditional CACs, the idea of three-valued-based CAC is proposed that benefits three symbol logics to represent the data in the bus lines, so we propose to use a three-valued CAC to reduce the number of bus lines required by traditional binary CACs.
- To omit the worst pattern of three-valued-based CAC on lines, a generation algorithm of 3VT-TOD free code words is proposed. In addition, maximum cardinality of Tri-value CAC that is maximum numbers of presentable code words is formulated and presented.
- We propose a novel encoding methodology that avoids to have worst-case transitions in three-valued CACs, thus enabling to minimize the effects of crosstalk effects. This overhead efficient numerical-based three-valued Crosstalk Avoidance Code is called Summation-based and numerical-based Tri-value Crosstalk Avoidance Code (ST-CAC) that benefits three symbols to represent the data in the bus lines, this can reduce the wire overheads of chips with neglectable codec overheads.
- The deterministic mapping algorithm for the ST-CAC coding mechanism is proposed that converts data words to code words.
- Transistor-level implementation for the decoder/encoder of ST-CAC is proposed.

The rest of the paper is organized as follows; in Sect. 2 crosstalk classification, based on tri-value transitions is discussed as a background. In Sect. 3, motivation of the proposed mechanism is discussed. The proposed coding mechanism and the implementation of the codec is presented in Sects. 4 and 6, respectively. In 5, the circuit implementation of coding mechanism is proposed. Finally, Sect. 7 concludes the paper.

2 Background

Parallel and adjacent bus lines between PEs are responsible for sending and receiving data between them, and the reliability of such data is seriously threatened by crosstalk faults. Crosstalk fault is caused by coupling capacitances between parallel bus lines, and their main negative effects are unwanted voltage glitches on bus lines that should be a stable, or the additional delay in the propagation of rising/falling transitions [6, 7].

The intensity of the increase in the delay on bus lines due to crosstalk faults depends on the data being transmitted on the bus. According to [17], the propagation delay τ_j of a bus line j named victim line) placed in between other two adjacent lines (i.e., line $j-1$ and line $j+1$, named aggressors) is given by [17]:

$$\tau_j = s \cdot |C_L \cdot \Delta V_j + C_I \cdot \Delta V_{j,j-1} + C_I \cdot \Delta V_{j,j+1}| \tag{1}$$

where s is a constant that depends on the line resistance and line driver strength; ΔV_j denotes the voltage swing on the victim line j; $\Delta V_{j,j+1}$ and $\Delta V_{j,j-1}$ denote the relative voltage swing between the victim line j and aggressor $j+1$ and $j-1$, respectively; $C_L$is the physical capacitance between the line and the substrate; and C_I is the physical inter-line capacitance between two adjacent lines.

For bus lines carrying data encoded by binary values, ΔV_j can be equal to (a) 0 for no transition in the victim line j, or (b) V_{dd} (for a $0 \to 1$ transition) or $-V_{dd}$ (for a $1 \to 0$ transition). $-V_{dd}$ is drain supply. On the other hand, $\Delta V_{j,j+1}$ and $\Delta V_{j,j-1}$ can be equal to (a) 0 for no transition in the adjacent lines or for transitions in the adjacent lines with the same direction, (b) $\pm V_{dd}$ for a transition in line j and no transition in the adjacent aggressor $j+1$ (or $j-1$), or (c) $\pm 2V_{dd}$ for opposite transitions between the victim line j and the aggressor $j+1$ (or $j-1$). Considering V_{step} as the difference between the voltages associated with the two logic levels, Eq. (1) can be written as Eq. (2) as follows:

$$\begin{aligned} \tau_j &= s \cdot C_L \cdot V_{\text{step}} |\delta_j + 2\lambda \cdot \delta_j - \lambda \cdot \delta_{j,j-1} - \lambda \cdot \delta_{j,j+1}| \\ \lambda &= \frac{C_I}{C_L}, \delta_j = |\frac{\Delta V_j}{V_{\text{step}}}|, \delta_{j,j+1} = \frac{\text{sign}(V_j) \times \Delta V_{j+1}}{V_{\text{step}}} \\ \delta_{j,j-1} &= \frac{\text{sign}(V_j) \times \Delta V_{j-1}}{V_{\text{step}}} \end{aligned} \tag{2}$$

For transitions on the victim line j it is $\delta_j = 1$, otherwise $\delta_j = 0$. Also, for $\delta_{j,j+1} = 1$ ($\delta_{j,j-1} = 1$).

For bus lines carrying data encoded by binary values, $\delta(j, k)$ can be equal to: a) 1 if two transitions have the same direction on the tandem lines b) -1 if they do not have the same transition directions and c) 0 for no transition. The normalized total crosstalk fault is defined as [17]:

$$X_{\text{eff},j} = |2\delta_j - \delta_{j,j-1} - \delta_{j,j+1}| \tag{3}$$

For bus lines carrying data encoded by binary values, $X_{\text{eff},j}$ is minimized if $\delta_j = \delta_{j,j-1} = \delta_{j,j+1}$, while it is maximized if $\delta_{j,j-1} = \delta_{j,j+1} = -\delta_j$. Equation (1 can be rewritten as:

$$\tau_j = s \cdot C_L \cdot V_{\text{step}} |\delta_j + \lambda \cdot X_{\text{eff},j}| \tag{4}$$

For $\lambda \gg 1$ (which is true for scaled technologies), τ_j is linearly related to $X_{\text{eff},j}$. Since δ_j, $\delta_{j,j-1}$ and $\delta_{j,j+1}$ can be equal to -1, 0 or 1, then we have $X_{\text{eff},j} = 0, 1, 2, 3, 4$ or 5. In Table 1, the value of transition pattern in the bus lines based on the value of $X_{\text{eff},j}$ is shown. As can be seen, $X_{\text{eff},j} = 4$, which corresponds to the worst crosstalk effect,

Table 1 Transition patterns on binary-value channel

X_{eff}	Transition patterns			
1	000	111		
2	−00	− 11	11 −	00−
3	−0 −	−1−		
4	−10	10 −	01 −	−10
5	010	010		

occurs the victim line presents an opposite transition with respect to the aggressors [18] (i.e., for the transition pattern 010 → 101).

On the other hand, for the case the data in the bus lines is encoded by means of three symbols (0, 1 and 2), the voltage value associated to each symbol is 0V for the symbol 0, $V_{dd}/2$ for the symbol 1, and V_{dd} for the symbol 2. Therefore, for this case, δ_j,$\delta_{j,j-1}$ and $\delta_{j,j+1}$ can assume one of the following values: − 2, − 1, 1 and 2 based on the transition appearing on the wires.

Moreover, according to Eq. (3), for this case the normalized crosstalk $X_{\text{eff},j}$ can assume one of the following values 0, 1, 2, 3, 4, 5, 6, 7 or 8.

By substitution values of these relative delays in Eq. (3), maximum value for $X_{\text{eff},j}$ is equal to $X_{\text{eff},j} = 8$. In other words, transition patterns that cause the line to experience normalized crosstalk effect of 8 imposes the highest crosstalk effect on the victim line. Transition patterns that cause the worst crosstalk effects on the victim line are 202 → 020 and 020 → 202 [18], because obtain a maximum value of $X_{\text{eff},j} = 8$. These transitions are called 3Value Tri-opposite Direction (3VT-TOD) transitions and are the transitions that the methodology presented in this paper aims to avoid and are shown in Table 2.

3 Motivations

Numerical-based CACs prevent the specific transition patterns to occur in the generated code words [17]. For example, in FPF CACs, '010' and '101' transition patterns are prevented. This reduces the crosstalk faults' delay of $X_{\text{eff}} = 4$ to $X_{\text{eff}} = 2$. However, the prevention of these transition patterns reduces the number of data words that can be used to represent data words. The number of code words that can be generated using numerical-based CACs is called the coding mechanism's Cardinality. Considering k as the number of lines between PEs, it is proved that the Maximum Cardinality Size (MCS) [17] of FPF CACs is equal to $2 \times F_{k+1}$ where F_{k+1} is the

Table 2 Transition patterns on three value

X_{eff}	Worst transition patterns in three value	
8	202	020
8	020	202

(k+1)th number of the Fibonacci sequence [17]. Besides, the MCS for FTC CACs is equal to F_{k+2}. Considering g_k as MCS for OLC CACs [13], it can be calculated by:

$$g_k = g_{k-1} + g_{k-5} \quad \text{for} \quad k \geq 6$$

where k is the numbers of lines and $g_1 = 2$, $g_2 = 3$, $g_3 = 4$, $g_4 = 5$ and $g_5 = 7$. $S_k, \ldots, S_1$ are the bases that show the weights. Also, considering s_k as MCS for OLC CACs, the MCS of FOC can be calculated by:

$S_k = S_{k-1} + S_{k-2} + S_{k-3}$ where $S_1 = 2$, $S_2 = 4$, $S_3 = 7$ [13]. Therefore, the main problem of numerical-based CACs, such as FPF, FOC, FTC, and OLC, is their inability to represent all possible 2^k code words in a bus with k lines, because of the elimination of some specific transition patterns [13]. To solve this problem, designers need to increase the number of lines in the bus in order to be able to represent the required 2^k code words. As an example, in order to represent $2^k = 256$ data words (with $k = 8$) when adopting a FPF CAC, number of additional bus lines equal to $[\log_2^{(2^k - \text{MCS})}] = 8$ is needed, where $\text{MCS} = 2 \times F_{k+1} = 2 \times F_9 = 42$ is the maximum cardinality size for the FPF CAC with $k = 8$.

As an example, Fig. 1 reports the MCS achievable by some CACs (FOC, FPF, FTC, and OLC) as a function of the number of bus lines. The figure also reports the maximum achievable MCS in case of no CAC is adopted in the bus without using CAC. As can be seen, as the channel width increases, the MCS achievable with binary CACs reduces significantly with respect to the maximum possible MCS obtained when no CAC is adopted. As a consequence, the number of additional lines (thus also associated area overhead) required by binary CACs increases significantly with the number of bus lines. This problem and with respect to this fact that ECCs [25] have a higher cost, we motivated to propose a three-valued coding mechanism that increases the MCS of code space and reduces the additional lines in communication channels.

To address this problem, in this paper we present a novel CAC that is based on the use of three symbols (0, 1 and 2, each one associated to a different voltage value)

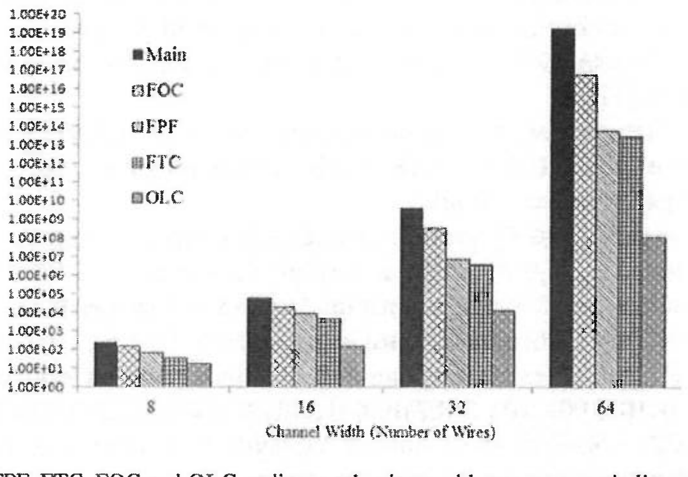

Fig. 1 MCS of FPF, FTC, FOC and OLC coding mechanisms with respect to main line code word capacity

to represent the data in the bus lines, we refer to it as Summation-based Tri-value Crosstalk Avoidance Code (ST-CAC). By using three symbols to represent data, our ST-CAC enables to increase the number of code words of a numerical system without increasing significantly the number of required bus lines. In Table 3, different examples for crosstalk avoidance coding mechanisms are shown.

4 Proposed three-valued coding mechanism

As discussed in Sect. 2, in bus lines where the data are encoded by using three symbols, the 3Value Tri-opposite Direction (3VT-TOD) transitions ($020 \rightarrow 202$ and $202 \rightarrow 020$) impose the worst crosstalk effects on the victim line. In this section, 3VT-TOD free code words and the maximum cardinality of theses codes are presented. Then an overhead efficient numerical-based three-valued CAC called Summation-based Tri-value Crosstalk Avoidance Code (ST-CAC), which is based on the use of three symbols (0, 1 and 2, each one associated to a different voltage value) to represent the data in the bus lines. Our ST-CAC enables a reduction of up to 33% the number of additional lines compared to the most recent binary CAC in literature. Moreover, our ST-CAC also enables reductions in terms of area overhead and power consumption required by the encoder and the decoder over those of binary CACs.

4.1 3VT-TOD free codes

As described in Sect. 2, 3VT-TOD transition patterns impose the worst crosstalk delay on the victim line. The main goal of our ST-CAC is to reduce crosstalk delay by avoiding 3VT-TOD transition patterns. In particular, 3VT-TOD free transition patterns can be obtained by using CACs based on data words encoded by three symbols (0, 1 and 2). We could simply generate 3VT-TOD free code word by removing all code words containing $020 \rightarrow 202$ and $202 \rightarrow 020$ transitions. This could be simply achieved by performing an extensive search over all possible code words; however, it would be unfeasible in terms of computation time for buses with a realistic number of lines [17].

3VT-TOD free code words can be generated using the inductive procedure. Let P_K be the set of 3VT-TOD free code words, a k-bit vector $D_k = d_k d_{k-1} \dots d_1$ is the vector of the generated code word.

Any code word $D_{k-1} \in P_{k-1}$ can be considered as concatenating $D_k = d_k d_{k-1} \dots d_1$ with bit d_k where $D_{k-1} \in P_{k-1}$. The algorithm for generating 3VT-TOD free code words is shown in Fig. 2. In this algorithm, $'.'$ denotes concatenation operation. d is the generated 3VT-TOD free code word and P_i for $k > 2$ are the collections of generated 3VT-TOD free codes in each round of the algorithm. For example {000,100,2 00,001,101,201,102,002,010, 110,210,011,111,211,012,112,212,120,220,021,121,2 21,022 ,122,222 } is all of the vectors of VT-TOD free code words in 3-link using inductive procedure.

Table 3 6-bit FPF code word generated by FIBO-CAC [16], improved-FIBO-CAC [15], PS-FIBO [22] and Su-Num [24]

	Fibo-CAC	Improved-Fibo -CAC	PS-Fibo	Sub-Num		Fibo-CAC	Improved-Fibo -CAC	PS-Fibo	Sub-Num
	$S_6S_5S_4S_3S_2S_1$	$S_6S_5S_4S_3S_2S_1$	$S_6S_5S_4S_3S_2S_1$	$S_6S_5S_4S_3S_2S_1$		$S_6S_5S_4S_3S_2S_1$	$S_6S_5S_4S_3S_2S_1$	$S_6S_5S_4S_3S_2S_1$	$S_6S_5S_4S_3S_2S_1$
	1 1 2 3 5 8	1 1 2 3 5 13	5 5 11 2 1 1	1 0 1 3 1 2		1 1 2 3 5 8	1 1 2 3 5 13	5 5 11 2 1 1	1 0 1 3 1 2
0	000000	000000	000000	000000	13	000011	000000	001100	------
1	100000	100000	000001	110000	14	100011	100001	001110	------
2	110000	110000	000011	000001	15	110011	110001	001111	------
3	011000	011000	000110	110001	16	000111	011001	011000	------
4	111000	111000	000111	011100	17	100111	111001	011001	------
5	001100	001100	000001	111100	18	001111	000011	011100	------
6	011100	011100	100001	000111	19	011111	100011	011110	------
7	111100	111100	100011	011111	20	111111	110011	111110	------
8	000110	000110	100110	111111	21	------	000111	111000	------
	000001								
9	100110	100110	100111	------	22	------	100111	111001	------
	100001								
10	001110	001110	110000	------	23	------	001111	111100	------
	110001								
11	011110	011110	110001	------	24	------	011111	111110	------
	011001								
12	111110	111110	110011	------	25	------	111111	111111	------
	111001								

Fig. 2 The generation algorithm of the 3VT-TOD free code word

```
$P_2 = \{00,01,02,10,11,12,20,21,22\}$
for $k \geq 3$ do
        $P_k = \{\}$;
                for $\forall D_{k-1} \in P_{k-1}$ do
    if $d_{k-1}.d_{k-2}$
 $= (00$ or $01$ or $10$ or $11$ or $12$ or $21$ or $22)$ then
            add $0.D_{k-1}$ and $1.D_{k-1}$ and
$2.D_{k-1}$ to $P_k$;
           else if $d_{k-1}.d_{k-2} = 02$ or $20$ then
             add $1.D_{k-1}$ and $d_{k-1}.D_{k-1}$ to $P_k$;
           end if
  end for
end for
```

4.2 Maximum cardinality of 3VT-TOD free codes

According to the algorithm shown in Fig. 2, 3VT-TOD-free coding mechanisms prevent specific transition patterns from code words. This reduces the ability of a 3VT-TOD-free coding mechanism in representing data words. For communication channels with k lines, the cardinality of 3VT-TOD free codes can be calculated based on a proof provided. In this regards, the maximum cardinality of 3VT-TOD free codes is equal to:

$$G_g(k) = 2G_g(k-1) + 2G_g(k-2) + G_g(k-3) \tag{5}$$

where $G_g(k)$ is the total number of distinct vectors in k lines. Equation (13) is a recursive relation that shows the maximum cardinality of 3VT-TOD free code with the primary value of $G_g(3) = 9, G_g(4) = 25$ and $G_g(5) = 71$. to propose and proof the maximum cardinality of 3VT-ToD Free coding mechanisms, first this definitions should be provided:

$G_g(k)$ is the total number of distinct vectors in k line. $G_{gg}(k)$ is the total number of distinct 3VT-TOD free vectors which satisfy the conditions of $d_k.\ d_{k-1} \neq 02$ and 20. $G_{gb}(k)$ is the total number of 3VT-TOD-free vectors which satisfy $d_k.\ d_{k-1} = 02$ and 20 conditions.

$G_{gg0}(k)$ is subsection of $G_{gg}(k)$ where $d_k = 0$
$G_{gg1}(k)$ is subsection of $G_{gg}(k)$ where $d_k = 1$
$G_{gg2}(k)$ is subsection of $G_{gg}(k)$ where $d_k = 2$
$G_{gb0}(k)$ is subsection of $G_{gb}(k)$ where $d_k.\ d_{k-1} = 02$
$G_{gb2}(k)$ is subsection of $G_{gb}(k)$ where $d_k.\ d_{k-1} = 20$

According to the algorithm in Fig. 2 and the definition of G_g, G_{gg} and G_{gb} we get:

$$G_g(k) = G_{gg}(k) + G_{gb}(k) \tag{6}$$

$$G_{gg}(k) = G_{gg0}(k) + G_{gg1}(k) + G_{gg2}(k) \tag{7}$$

$$G_{gb}(k) = G_{gb0}(k) + G_{gb2}(k) \tag{8}$$

Also:

$$G_{gg}(k) = 2G_{gg0}(k-1) + 3G_{gg1}(k-1) + 2G_{gg2}(k-1) + 2G_{gb0}(k-1) + 2G_{gb2}(-1) \tag{9}$$

$$G_{gb}(k) = G_{gg0}(k-1) + G_{gg2}(k-1) \tag{10}$$

$$G_g(k) = 2G_g(k-1) + G_{gg}(k-1) \tag{11}$$

$$G_g(k) = 2G_g(k-1) + 2G_g(k-2) + G_{gg1}(k-2) \tag{12}$$

And with substituting $G_{gg1}(k-2)$ with $G_g(k-3)$:

$$G_g(k) = 2G_g(k-1) + 2G_g(k-2) + G_g(k-3) \tag{13}$$

Equation (13) is the results of Eqs. 6–13 and is a recursive relation that shows the MCS of 3VT-TOD-free vectors in k-line of the channel with the primary value of $G_g(3) = 9, G_g(4) = 25$ and $G_g(5) = 71$.

4.3 Summation-based and numerical-based tri-value coding mechanism

To provide an overhead-efficient 3VT-TOD free coding mechanism, numerical systems are among the efficient mechanisms [17]. Each numerical system consists of a sequence of integers as a base used as a weight of code words in representing code words. Fibo-CAC, Improved Fibo-CAC, PS-Fibo, and Sub-Num are among the recently proposed CACs that use numerical systems to generate code words. For example in the binary-value communication channel, the Fibonacci numerical system uses the Fibonacci sequence as a base to represent code words. As an example, in a channel with 6-line, the numerical system sequence of bases consists of 8 5 3 2 1 1 and a data word with the value of 18 shown is mapped to '111100' that means $111100 = 1 \times 8 + 1 \times 5 + 1 \times 3 + 1 \times 2 + 1 \times 0 + 1 \times 0$. However, as the channel width increases, the MCS achievable with binary CACs reduces significantly with respect to the maximum possible MCS obtained when no CAC is adopted. As a consequence, the number of additional lines (thus also associated area overhead) required by binary CACs increases significantly with the number of bus lines. In this regards, a tri-value communication channel can solve the overhead problem of binary CAC efficiently. Tri-value communication channel should satisfy the following conditions to be used as a 3VT-TOD free coding mechanism:

- **The numerical system should be complete** Completeness means that for each data word υ, there should be at least one code word representation in the numerical system. S_is are the values of bases in the sequence of the numerical system.
- **The numerical system should be able to generate 3VT-TOD free code words** Numerical system should be able to represent each code word d without 3VT-TOD transition patterns.

Using these two conditions, Summation-based Tri-value Crosstalk Avoidance (ST-CAC) is proposed in this paper. Considering the $NS_{\text{ST-CAC}}$ as the sequence of bases of the proposed numerical system we have:

$$NS_{\text{ST-CAC}} = (S_k, S_{k-1}, \dots, S_{i+2}, S_{i+1}, S_i, \dots, S_1)$$

$NS_{\text{ST-CAC}}$ bases can be calculated by Eq. 14, as follows:

$$S_i = \begin{cases} 1 & i = k \\ 3 & i = k-1 \\ 6 & i = k-2 \\ 2S_{i+1} + S_{i+2} & 1 \le i \le k-3. \end{cases} \tag{14}$$

In Eq. 14, S_i is the base of the ith base of the numerical system. S_i defines rules for generating the bases sequence based on each bit position of the numerical system. According to Eq. 14, to generate the base sequence of the numerical system in k line channel, except the last three bit positions including $k, k-1$ and $k-2$ that is equal to $1, 3$ and 6, respectively, the rule for $1 \le i \le k-3$ is the summation doubled of S_{i+1} and S_{i+2}. As the suggested numerical system is based on the summation of the digit of sequences in tri-value system, the proposed CAC is called Summation-based Tri-value Crosstalk Avoidance Code (ST-CAC). For example in a 5-line channel using ST-CAC, the base sequence is equal to 1 3 6 15 36. As shown in continue, all of the code words using the ST-CAC numerical system have the condition to be used as 3VT-TOD free codes. The ST-CAC numerical system has the condition to be used as the 3VT-TOD free coding mechanism. It should be shown that:

- **ST-CAC numerical system is complete** There is an equivalent code word, to generate any 3VT-TOD free code word using the ST-CAC coding mechanism. According to 3VT-TOD, free coding mechanisms condition's ST-CAC is complete because whenever $S_1 = 1$, the $S_i \le 1 + 2\sum_{j=1}^{i-1} S_j$ relation is satisfied.
- **The numerical system should be able to generate 3VT-TOD free code words** Beside the completeness of numerical system, the condition for generating 3VT-TOD free code is satisfied by ST-CAC numerical system because according to the definition of this numerical system, these features can be proved:

  Feature 1 Based on the definition of ST-CAC in Eq. 14, each '020'can be replaced with '001'.

  Feature 2 Based on the definition of ST-CAC in Eq. 14, each '202'can be replaced with '012'.

As ST-CAC numerical system is complete and has a condition to be used as a 3VT-TOD free code, it can be proved that ST-CAC can be used as the 3VT-TOD free coding mechanism.

4.4 Mapping algorithm and hardware architecture of ST-CAC

To generate the 3VT-TOD free code word using the ST-CAC coding mechanism, an iterative mapping algorithm shown in Fig. 3 is proposed. This mapping

Fig. 3 Mapping algorithm of ST-CAC coding mechanism

```
$for\ (i = k;\ to\ k - 1; i + +)$
  $if\ (r_{i-1} \geq 2S_i + S_{i+1})then$
      $d_i = 2;$
      $r_i = r_{i-1} - 2S_i;$
$elseif\ (r_{i-1} \geq S_i)then$
      $d_i = 1;$
      $r_i = r_{i-1} - S_i;$
$else$
      $d_i = 0;$
      $r_i = r_{i-1};$
$for\ (i = k - 2;\ to\ 1; i + +)$
      $if\ (r_{i-1} \geq 2S_i)then$
      $d_i = 2;$
      $r_i = r_{i-1} - 2S_i;$
   $elseif(r_{i-1} \geq S_i)then$
      $d_i = 1;$
      $r_i = r_{i-1} - S_i;$
$else$
      $d_i = 0;$
      $r_i = r_{i-1};$
         $return\ d_k \ldots d_2 d_1$
```

algorithm maps data word $\upsilon = \upsilon_k \upsilon_{k-1} \ldots \upsilon_1$ to an equivalent 3VT-TOD code word $d = d_k d_{k-1} \ldots d_1$. In this regard, each state of this algorithm generates one bit of code word d_i and the reminder of r_i.

This reminder is required in the next stage. The value of each bit of d_i is calculated based on comparison with the values of $2S_i$, S_i and $2S_i + S_{i+1}$. As shown in Fig. 3, calculation of the most and penultimate bit position of the 3VT-TOD-Free code words, i.e., first and penultimate rounds of the algorithm are different from other bit positions of the code word. ST-CAC coding algorithm can simply be applied to any arbitrary channel width. In the receiver, the ST-CAC decoder calculates $\upsilon = \sum_{i=1}^{k} u_i \times S_i$ for a k-bit channel.

Hardware architecture of ST-CAC coding mechanism including (a) encoder and (b) decoder for a k-bit channel is shown in Fig. 4. For encoding and decoding procedure, processing (P_i s) are used that are processors responsible for computation.

$$\upsilon = \sum_{i=1}^{k} d_i \times S_i \tag{15}$$

where $S_K, S_{K-1}, \ldots, S_{i+2}, S_{i+1}, S_i, \ldots, S_1$ are the bases sequence defined according to the proposed numeral system for a k-bit channel. In this way, the ST-CAC decoder can simply be determined for any width of the channel. For example, the 4-bit 3VT-TOD code word $d =$'1021'coded by the ST-CAC coding mechanism can map to data word $v = 1 \times 1 + 0 \times 3 + 2 \times 6 + 1 \times 15 = 28$.

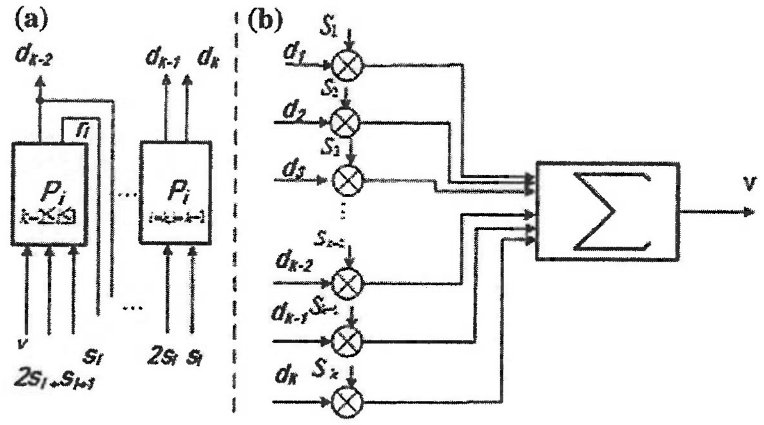

Fig. 4 Hardware architecture of ST-CAC coding mechanism a encoder, b decoder

5 Circuit implementation

To transfer data words from the receiver to transmitter using ST-CAC, the voltage-mode implementation is used. Figure 5a shows the schematic implementation of switches to generate tri-value. Using this circuit, the input bits, B_1B_0, are converted into three voltage levels on the line. Using this circuit, three levels of voltages are connected to the line using three switches. These switches are controlled by C_1, C_2, and C_3 signals. Figure 5b represents the circuit-level implementation of tri-value communication. To apply $0.75V$, $0.45V$, and $0.15V$ that represent tri-values, only PMOS and NMOS transistors are used as switches. Each ternary data word is determined by two binary bits, i.e., B_1B_0. Then, the ternary data word, B_1B_0, is represented by a binary signal to be transmitted through the lines. The output bits of the digital decoder are converted into three voltage levels to be transmitted. Then, in the receiver, the voltage level of the line (three possible levels) is converted into binary bits, B_1B_0 In this regards, the switch is

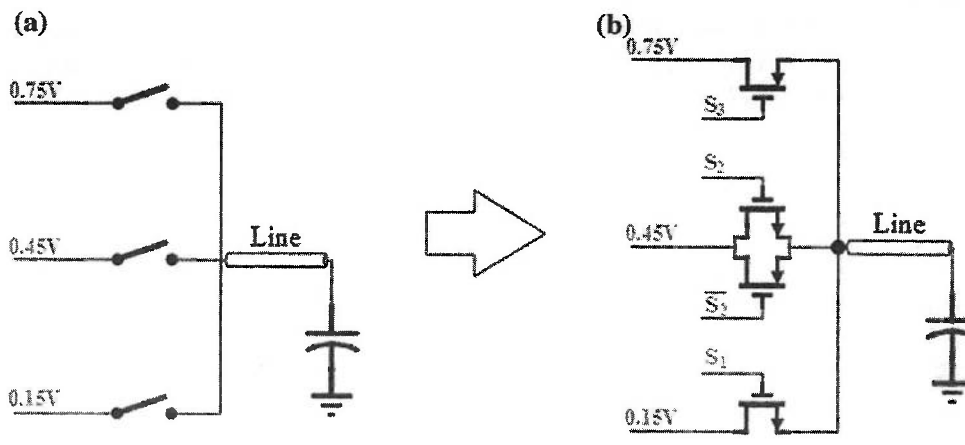

Fig. 5 The implementation of switches to generate tri-value

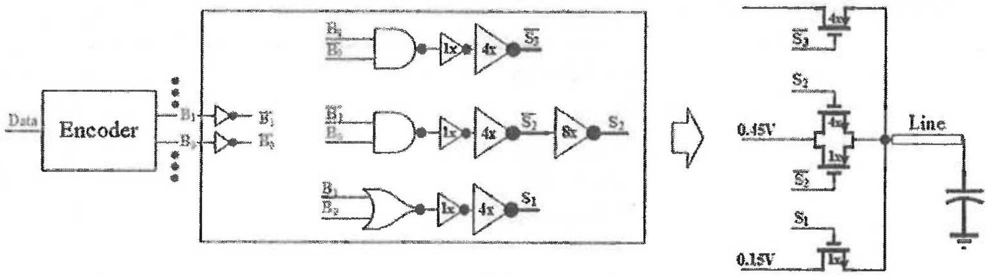

Fig. 6 The Architecture of controller that generates input signals of switches in the transmitter

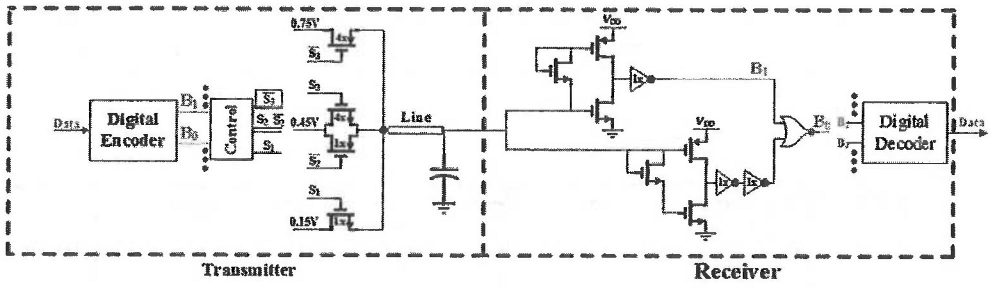

Fig. 7 The architecture of transmitter and receiver

required to generate tri-value signals. In this regards, the switch is required to generate tri-value signals. In fact, the other type of transistor is almost off when the switch is on so it can be omitted from the switch without any limiting effect. Two input binary signals, B_1B_0, which represent one ternary signal is used to generate C_1, C_2, and C_3 signals. In order to generate signals using B_1B_0, a controller is required.

The architecture of the transmitter and receiver with controllers is shown in Fig. 7.

Figure 6 shows the architecture of a controller circuit that generates input control signals of switches. The output buffers should be sized considering the sizing of the transistors used in the switches which are determined by the length of the line. In order to decode the received signal, two inverters with large and low-voltage thresholds are used. Figure 6 represents the decoder circuit. The received signal is converted to two binary signals representing the input signals.

The circuit is implemented in 45*nm* CMOS technology. The circuit is tested under typical conditions presented in Table 4. This table shows that the proposed architecture operates up to 15.5 Gbit/s. Figure 8a presents a sample set of input waveforms (of B_1B_0). B_1B_0 can be "00", "10", and "01"as shown in Fig. 8. Figure 8b presents the output waveform which appears on the line. As can be seen, there are three voltage levels of 0.75 V, 0.45 V and 0.15 V. Also, the worst-case delay is about 52ps that it shows a reduction of 43% with respect to the line without using any coding mechanism. Also, Fig. 8c presents the received bits. The worst-case delay is about 60 ps.

Table 4 Parameters and results

Parameter	Value
Technology	TSMC 45 nm
C_{Ground}	0.4132 f
$C_{\text{Between Wires}}$	5.0568 f
Resistance	0.6875 Ohms
Voltage supply	0.9 V
Bit rate	16.5 Gb/s
Transmit delay	52 ps
Receive delay	60 ps
Transmitter power consumption (@1 $textGb/s$)	24 uW
Receiver power consumption (@1 $textGb/s$)	8.2 uW
Line load	100 fF

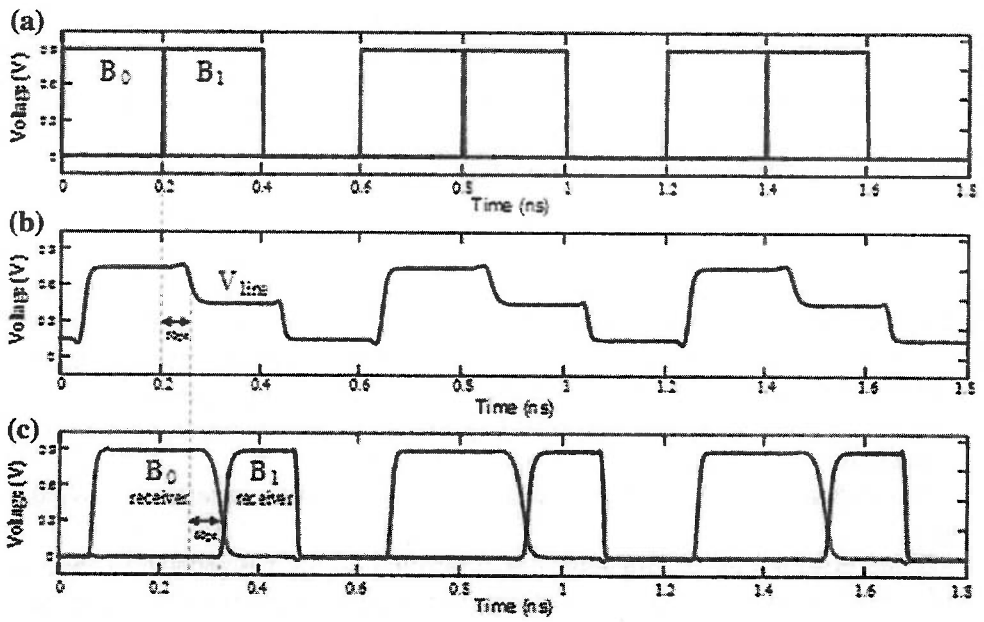

Fig. 8 The sample waveforms on the wire

6 Experimental evaluations

In this section, the effects of inserting the ST-CAC coding mechanism, including additional lines, the power consumption of lines and the overhead of codec including; power consumption, area occupation and critical path are investigated in more details. In this regard, VHDL-based simulations and Design Compiler tool is used to calculate the imposed overheads of codecs. Also, the SPICE simulations are used to calculate the power consumption of lines in 45 nm technology size. For fair comparisons, our proposed coding mechanism is compared with state-of-the-art CAC including PS-Fibo, Improved-Fibo-CAC and Sub-Num coding mechanisms.

6.1 Additional line

As described in Sect. 3, the main advantage of our ST-CAC is the use of three symbols to encode the data on the bus, which enables to reduce the number of lines in communication channels. As an example, Fig. 9 reports the total numbers of crosstalk-free code words achievable with our ST-CAC and with a traditional two value CAC as a function of the number of bus lines (channel width). We can observe that our ST-CAC enables to significantly reduce the number of additional lines with respect to a traditional binary CAC by 33%. This is because the use of 3 symbols $\{0, 1, 2\}$ to encode the data in the bus line increases the ability of the numerical system in representing crosstalk-free code words.

Moreover, the number of required lines and the percentage of line saving allowed by our ST-CAC with respect to a traditional CAC are shown in Fig. 10. These results in different widths show that ST-CAC unlike two value CACs only imposes additional lines to the system, but also it saves them.

6.2 Power consumption of lines

To evaluate the effects of the ST-CAC coding mechanism on the power consumption of lines, SPICE simulations are carried out. SPICE simulations are used to accurately simulate lines of channels. To model the lines of the channel, the Predictive Interconnect Model (PIM) [26] is used. In this model, the line width (w), space between lines (s), line thickness (t) and line distance from ground layer height (h) of theline in 45 nm technology are used. In these simulations, each line d_i of the channel induces capacitance of C_{iG}, the resistance of R_i and inductance of L_i. The results

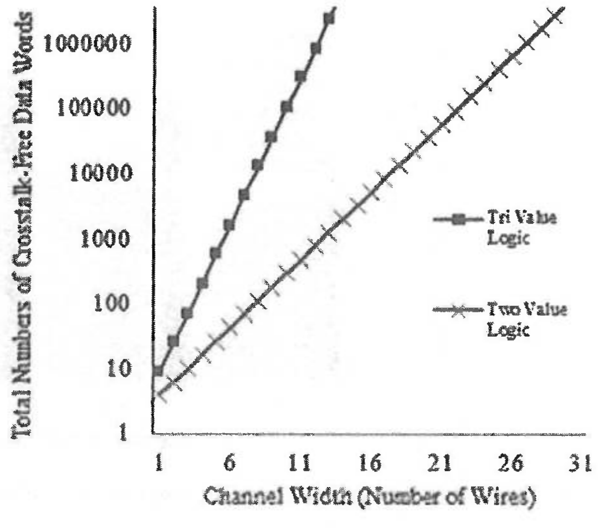

Fig. 9 The number of crosstalk-free code words in 3-value and 2-value logic numerical systems

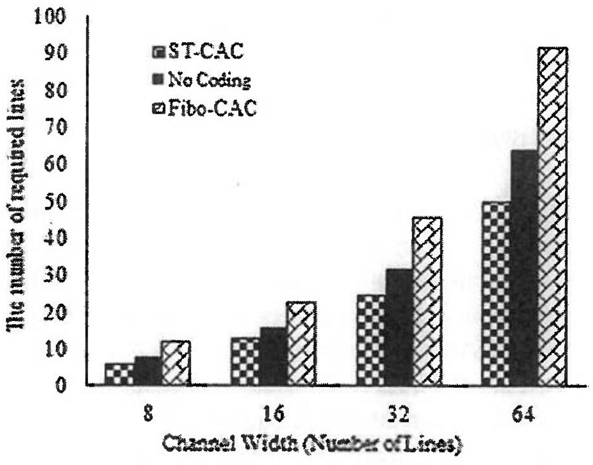

Fig. 10 The number of required lines for ST-CAC with respect to 2D-CAC, shielding and no coding

of the power consumption of lines by applying different bit streams are reported in Fig. 11. These results confirm that power consumption of lines in the channel is improved with respect to no coding channel and channels using Fibo-CAC, Improved-Fibo-CAC system, PS-Fibo and Sub-Num CACs coding mechanisms.

To have fair comparisons in terms of the dynamic power consumptions, the dynamic power consumption of redundant lines should be taken into consideration. The power consumption of lines is affected by switching activity in lines, ST-CAC can reduce the switching activity by reducing the 3VT-TOD-Free. In this regard, to have fair comparisons, we have also estimated the power consumption of these redundant lines.

6.3 Overhead of codec

To evaluate the imposed overhead of codec, ST-CAC coding mechanisms with respect to Fibo-CAC, Improved-Fibo-CAC system, PS-Fibo and Sub-Num CACs

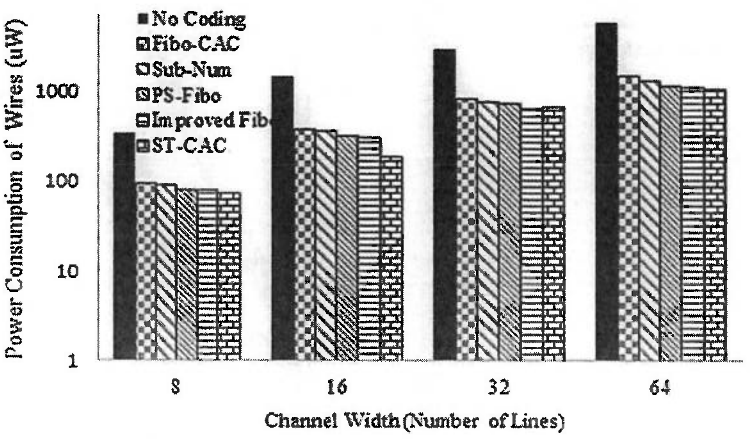

Fig. 11 Power consumption of lines in the presence of ST-CAC with respect to other codecs

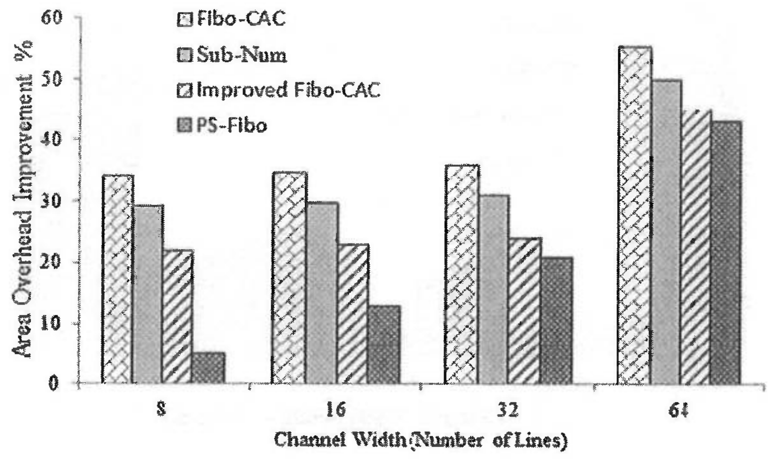

Fig. 12 Power consumption improvement in codec of ST-CAC with respect to other codecs

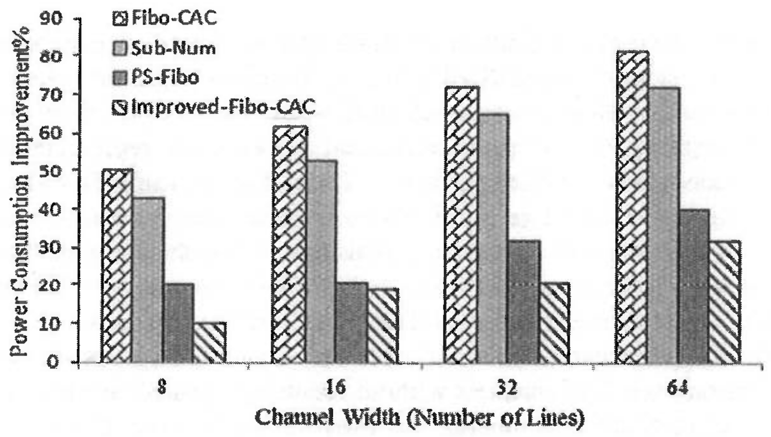

Fig. 13 Area occupation improvement in ST-CAC codec with respect to other codecs

coding mechanisms, they are implemented and evaluated to estimate power consumption, area occupation and critical path timing of codec. In this line, these coding mechanisms are evaluated by VHDL-based simulations and synthesized using Nangate 45 nm technology library. Experiments are done in a wide range of channel widths including 16, 32 and 64 channel widths. Percentage of the percentage of, power consumption improvement in codec, area occupation improvement in codec and the percentage of critical path timing improvement in codec for each coding mechanism are reported in Figs. 12, 13 and 14, respectively. As shown in these figures, ST-CAC codec outperforms Fibo-CAC, Improved-Fibo-CAC system, PS-Fibo and Sub-Num CACs codecs and imposes lower overhead in terms of all mentioned characteristics. In addition, experiments show that the ST-CAC coding mechanism provides more improvements when the width of the channel grows.

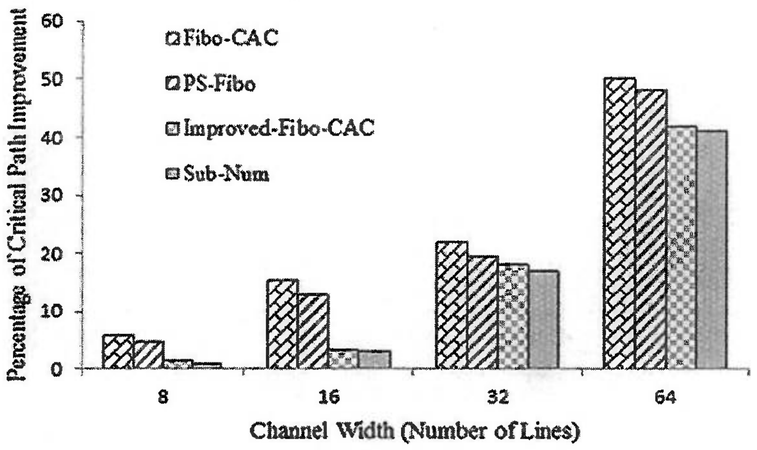

Fig. 14 Critical path improvement in ST-CAC codec with respect to other codecs

7 Conclusions

CACs can reduce crosstalk fault by omitting specific transition patterns from channels. However, state-of- the-art CACs impose additional lines on systems. To solve this problem and to slow down the channel width growth, this paper presents Tri-value numerical systems. In these numerical systems, the representation digits of generated code words can be changed to 0, 1 and 2 set by using Tri-value numerical systems instead of 0 and 1 set in binary-value numerical systems. Calculating the MCS of Tri-value numerical systems reveals that it can efficiently increase the total number of code words in specific channel width. We also propose a Tri-value coding mechanism called Summation-based Tri-Value Coding Mechanism (ST-CAC). The proposed coding mechanism is based on a novel numeral system that can be used for any arbitrary width of channels without requiring channel partitioning. The proposed coding mechanism eliminates the worst crosstalk-induced transition patterns from channels and offers invariant delay for channels. The wide range of carried out simulations confirms that the proposed coding mechanism improves the reliability of channels with lower power consumption, timing and area overheads compared with other coding mechanisms.

References

1. Choi W, Duraisamy K, Kim RG, Doppa JR, Pande PP, Marculescu D, Marculescu R (2018) On-chip communication network for efficient training of deep convolutional networks on heterogeneous manycore systems. IEEE Trans Comput 67(5):672–686
2. Pd SM, Lin J, Zhu S, Yin Y, Liu X, Huang X, Song C, Zhang W, Yan M, Yu Z et al (2017) A scalable network-on-chip microprocessor with 2.5 d integrated memory and accelerator. IEEE Trans Circuits Syst I Regul Pap 64(6):1432–1443
3. Itrs, the international technology roadmap for semiconductors-2015 edition. [Online]. (2015) Available: http://www.public.ITRS.net. Accessed 2015
4. Benini L, De Micheli G (2002) Networks on chips: a new soc paradigm. IEEE Trans Comput 35(1):70–78
5. Sridhara SR, Shanbhag NR (2007) Coding for reliable on-chip buses: a class of fundamental bounds and practical codes. IEEE Trans CAD Integr Circuits Syst 26(5):977–982
6. Tehranipour MH, Ahmed N, Nourani M (2003) Testing soc interconnects for signal integrity using boundary scan. In: Proceedings of 21st VLSI Test Symposium. IEEE, pp 158–163

7. Zimmer H, Jantsch A (2003) A fault model notation and error-control scheme for switch-to-switch buses in a network-on-chip. In: Proceedings of the 1st IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis. ACM, 2003, pp 188–193
8. Frantz AP, Kastensmidt FL, Carro L, Cota E (2006) Dependable network-on-chip router able to simultaneously tolerate soft errors and crosstalk. In: IEEE International Test Conference, 2006. ITC'06. IEEE, pp 1–9
9. Agarwal K, Sylvester D, Blaauw D (2006) Modeling and analysis of crosstalk noise in coupled rlc interconnects. IEEE Trans Comput Aided Des Integr Circuits Syst 25(5):892–901
10. Zhang J, Friedman EG (2004) Effect of shield insertion on reducing crosstalk noise between coupled interconnects. In: Proceedings of the 2004 International Symposium on Circuits and Systems, 2004. ISCAS'04. IEEE, vol 2, pp II–529
11. Kaul H, Seo J-s, Anders M, Sylvester D, Krishnamurthy R (2008) A robust alternate repeater technique for high performance busses in the multi-core era. In: IEEE International Symposium on Circuits and Systems, 2008. ISCAS. IEEE, pp 372–375
12. Ghoneima M, Ismail YI, Khellah MM, Tschanz JW, De V (2006) Reducing the effective coupling capacitance in buses using threshold voltage adjustment techniques. IEEE Trans Circuits Syst I Regul Pap 53(9):1928–1933
13. Wu X, Yan Z (2011) Efficient codec designs for crosstalk avoidance codes based on numeral systems. IEEE Trans Very Large Scale Integr (VLSI) Syst 19(4):548–558
14. Murali S, Theocharides T, Vijaykrishnan N, Irwin MJ, Benini L, De Micheli G (2005) Analysis of error recovery schemes for networks on chips. IEEE Des Test Comput 22(5):434–442
15. Duan C, Calle VHC, Khatri SP (2009) Efficient on-chip crosstalk avoidance codec design. IEEE Trans Very Large Scale Integr VLSI Syst 17(4):551–560
16. Duan C, Tirumala A, Khatri SP (2001) Analysis and avoidance of cross-talk in on-chip buses. In: Hot Interconnects 9, 2001. IEEE, pp 133–138
17. Duan C, LaMeres BJ, Khatri SP (2010) On and off-chip crosstalk avoidance in VLSI design. Springer, Berlin
18. Sotiriadis PP, Chandrakasan A (2001) Reducing bus delay in submicron technology using coding. In: Proceedings of the 2001 Asia and South Pacific Design Automation Conference. ACM, 2001, pp 109–114
19. Shirmohammadi Z, Miremadi SG (2015) S2ap: An efficient numerical-based crosstalk avoidance code for reliable data transfer of NOCS. In: 10th International Symposium on Reconfigurable Communication-Centric Systems-on-Chip (ReCoSoC). IEEE, pp 1–6
20. Shirmohammadi Z, Miremadi SG (2016) On designing an efficient numerical-based forbidden pattern free crosstalk avoidance codec for reliable data transfer of nocs. Microelectron Reliab 63:304–313
21. Chang C-S, Cheng J, Huang T-K, Huang X-C, Lee D-S, Chen C-Y (2015) Bit-stuffing algorithms for crosstalk avoidance in high-speed switching. IEEE Trans Comput 64(12):3404–3416
22. Shirmohammadi Z, Mozafari F, Miremadi S-G (2017) An efficient numerical-based crosstalk avoidance codec design for nocs. Microprocess Microsyst 50:127–137
23. Shirmohammadi Z (2019) Op-fibo: an efficient forbidden pattern free CAC design. Integration 65:104–109
24. Shirmohammadi Z, Mahdavi Z (2018) An efficient and low power one-lambda crosstalk avoidance code design for network on chips. Microprocess Microsyst 63:36–45
25. Lakshmi KASS, Keerthi A, Sri KM, Vinodhini M (2020) Code with crosstalk avoidance and error correction for network on chip interconnects. In: 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI) (48184). IEEE, pp 75–79
26. Predictive technology model, arizona state univ., temper, az. [Online]. (2020) http://ptm.asu.edu/. Accessed 2020