International Journal of Computer and Communication Technology

Volume 7 | Issue 2 Article 6

April 2016

Implementation of Memory Less Based Low-Complexity CODECS

K. Vijayalakshmi
Department of Electronics and Communication Engineering, Nalanda Institute Of Engineering And Technology, Sattenapalli, vijayalakshmi_ec@yahoo.co.in

I.V.G Manohar
Department of Electronics and Communication Engineering, Nalanda Institute Of Engineering And Technology, Sattenapalli, ivgmanohar@gmail.com

L. Srinivas
Department of Electronics and Communication Engineering, Nalanda Institute Of Engineering And Technology, Sattenapalli, lsrinivas@gmail.com

Recommended Citation
Vijayalakshmi, K.; Manohar, I.V.G; and Srinivas, L. (2016) "Implementation of Memory Less Based Low-Complexity CODECS," *International Journal of Computer and Communication Technology*: Vol. 7 : Iss. 2 , Article 6.
DOI: 10.47893/IJCCT.2016.1347
Available at: https://www.interscience.in/ijcct/vol7/iss2/6

Implementation of Memory Less Based Low-Complexity CODECS

K.Vijayalakshmi, I.V.G Manohar & L. Srinivas

Department of Electronics and Communication Engineering,
Nalanda Institute Of Engineering And Technology, Sattenapalli
E-mail : vijayalakshmi_ec@yahoo.co.in

Abstract – In this work, we present a CODEC design for two classes of crosstalk avoidance codes (CACs), forbidden pattern codes (FPCs) and forbidden transition codes (FTCs). Our mapping and coding scheme is based on the Fibonacci numeral system and the mathematical analysis shows that all numbers can be represented by FTF vectors in the Fibonacci numeral system (FNS). The proposed CODEC design is highly efficient, modular and can be easily combined with a bus partitioning technique. We also investigate the implementation issues and our experimental results show that the proposed CODEC complexity is orders of magnitude better compared to the brute force implementation. Compared to the best existing approaches, we achieve a 17% improvement in logic complexity. A high speed design can be achieved through pipelining.

In this paper, we generalize the idea in and establish a generic framework for the CODEC design of *all* classes of CACs based on *binary mixed-radix* numeral systems. Using this framework, we propose CODECs for OLCs and FPCs with optimal code rates as well as CODECs for FOCs with near-optimal code rates.

Keywords **-** Crosstalk, on-chip bus, Fibonacci number, CODEC, crosstalk avoidance codes (CACs), interconnect

I. INTRODUCTION

Most of the crosstalk reduction techniques involve removing or lowering the probability of undesired patterns, and inevitably incur area overhead from the additional wires in the bus, additional circuitry or both. The efficiency of a crosstalk reduction scheme should be judged not only by the performance boost it brings about, but also by its area overhead as well. As an example, passive shielding requires a doubling of the number of wires, and hence incurs a 100% area overhead and therefore is not deemed efficient.

Bus encoding schemes can achieve the same amount of bus delay improvement as passive shielding, with a much lower area overhead [6, 7, 10, 5]. These codes are commonly referred to as *Crosstalk Avoidance Codes* (CACs). CACs can be memory-less [7, 6, 5] or memory-based [10]. Memorybased coding approaches generate a codeword based on the previously transmitted code and the current dataword to be transmitted [7, 10]. Although these type of codes need fewer additional bus wires, the CODEC complexity is generally considered too high for these coding schemes to be used in practice. The memory-less coding approaches, on the other hand, use a fixed code book to generate a codeword, solely based on the current input data. The CODECs for memory less codes are projected to be simpler.

Several different types of memory-less CACs have been proposed [6, 7, 5]. All these codes offer the same degree of delay performance improvement. The area overhead caused by the additional wires ranges from 44% to 68%, which is much better than passive shielding. Two of the most efficient memory-less codes are *forbidden-pattern-free* (FPF) CACs [1] and *forbidden-transition-free* (FTF) CACs. Their overhead performance is near identical, and both methods approach the theoretical lower bound. The FPF is slightly better, but by no more than one wire.

Unfortunately, efficient CODEC designs are not available for either of these codes. Due to the non-linear nature of these codes, researchers have struggled to find a mapping scheme that is mathematically systematic and efficient to implement. Attempts through brute force logic optimization have shown that the CODEC gate count becomes prohibitively large for a bus of reasonable size, since its complexity grows exponentially with the bus size [5, 10].

In some high-speed designs where crosstalk delay would have limited the clock speed, the technique of

shielding was used. This involves putting a grounded wire between every signal wire on the bus. Although this certainly is effective in preventing crosstalk within the bus, it has the effect of doubling the wiring area. Cross-chip buses often must be routed in higher metal layers, which are scaled more slowly than the rest of the geometry in order to prevent an unacceptable increase in resistance. Thus, routing resources are scarce at these levels, and it can be difficult to justify doubling the bus width. However, if we abstract the concept of shielding and just look at the signals on the wires of a shielded bus, we can *think* of it as a very simple bus encoding. Two wires are used for every data bit. A data bit of *"0"* is encoded as a *"00"* signal on the wires, and a "1" is encoded as "lo". The purpose of this "encoding" is to prevent adjacent wires from transitioning in opposite directions, and this particular encoding achieves that goal by forcing every other wire to a steady value.

II. BACKGROUND

We can model the chain of communication as shown in Figure 1. Adopting some terminology from coding theory, we say that the data words to be encoded are represented by symbols.

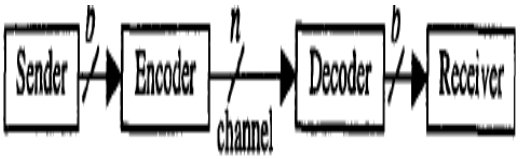

Fig 1. Model of Communication Chain

The mapping between symbols and actual data words is an implementation step and will not be discussed here.

codeword at time 1:	0010	0000	0100	0100
	↓	↓	↓	↓
codeword at time 2:	0110	1111	0001	0010
	valid	*valid*	*valid*	**invalid**

Fig 2.Examples of transitions

The values placed on the channel by the encoder are called *codewords,* and the mapping between symbols and codewords is called a *codebook.*

If the codebook changes with time, then the encoding is said to have *memory.* Specific to crosstalk-immune coding is the notion of which codewords can follow which. The fundamental rule is that, given a particular value currently on the channel, the next value cannot cause any adjacent wires to transition in opposite directions. We say that a codeword is *connected* to another codeword if it is valid to transition from one to the other. Figure 2 presents some examples of valid and invalid transitions. In order to import some terminology from graph theory, we can form a graph with the codewords as vertices and the connections as edges. This graph is undirected because the connection relation is symmetric. We can then say that the *neighbor set* of a codeword is the set of codewords that it is connected to, and its *degree* is the size of this set. Note that it is valid for a codeword to transition to itself, and thus every codeword has itself as a neighbor.

III. FPF ALGORITHM

Forbidden patterns are defined as the two 3-bit patterns "010" and "101". A *forbidden pattern free* code is a set of codewords which do not contain forbidden patterns on any 3 adjacent bus bits. For example, "100110" is FPF while "10111" is not an FPF code. By eliminating the forbidden patterns in the codewords, it is guaranteed that *Ceff* for any bit in the bus does not exceed $(1+2\lambda)CL$ [6] and hence the maximum delay is reduced by 50% compared to an uncoded bus.

The maximum cardinality of FPF codewords is $2f_{m+1}$ [6], where f_m is the m^{th} element in the Fibonacci sequence defined as:

$$f_m = \begin{cases} 0 & \text{if m} = 0, \\ 1 & \text{if m} = 1, \\ f_{m-1} + f_{m-2} & \text{if m} \geq 2. \end{cases}$$

Forbidden transition free CACs

The *forbidden transition* is defined as the simultaneous transition, in opposite directions, on any two adjacent wires in a bus. A code is forbidden transition free (FTF) if transitions between codewords do not generate forbidden transitions on any adjacent bits of the bus. This type of code was first investigated in [7].

Similar to FPF codes, FTF codes can be generated by eliminating certain patterns. Not to be confused with the 3-bit forbidden patterns, we refer to these patterns as *prohibited pattern*.

The prohibited pattern is either a "01" or "10" on two adjacent bus bits. The possible data patterns on two wires in a bus are "00", "01", "10" and "11". It is easy to see that the elimination of either "01" or "10" on two adjacent bits will cause the pair to be forbidden transition free. It has been proven in [7] that having alternating prohibited patterns on bits d_{2k}, d_{2k-1} and

d_{2k+1}, d_{2k} yields a set with the maximum number of codewords (cardinality).

Conversely, by prohibiting "10" on $d_{2k}d_{2k-1}$ and "01" on $d_{2k+1}d_{2k}$, we can produce a different set of FTF-CACs. The maximum cardinality of FTF-CACs is fm+2, slightly lower than the cardinality of FPF-CACs. When the bus size is large, the area overhead of FTF-CACs over an uncoded bus approaches 44% as well. Table 1 lists the codewords of one set of the 2, 3,4 and 5 bit FTF-CACs.

2-bit	3-bits	4-bits	5 bit	
00	000	0000	00000	10100
01	001	0001	00001	10101
11	100	0100	00100	10111
	101	0101	00101	11100
	111	0111	00111	11101
		1100	10000	11111
		1101	10001	
		1111		

Table1: FPF-CAC codewords for 2,3,4 and 5 bit busses

Recently, we have showed that there exists a deterministic mathematical mapping for FPF-CACs using FNS, and proposed two different coding algorithms as well as the corresponding CODEC implementations [1]. In this paper, we present the mathematical framework for FTF-CAC design using FNS, along with an algorithm for its CODEC.

Both the FTF-CACs and FPF-CACs were proposed earlier and algorithms were given to generate codewords for them [7, 6]. However, the CODEC design was not thoroughly addressed in the original papers. Since then, there have been more research results published on designing CODECs for these CACs. Most of the designs were based on bus partitioning techniques, which require additional wires on the bus.

More importantly, none of them addressed the fundamental issue of how to map datawords to codewords. This is partially due to the fact that the CACs are non-linear codes, and it is difficult to find a mapping using conventional mathematical expressions. [5] Showed that a brute force lookup- table implementation is impractical, as the CODEC size grows exponentially with the bus size. Figure 3 (obtained from [5]) shows this exponential growth with increasing bus size.

$$d_k d_{k-1} = \begin{cases} 00 & \text{if } r_{k+1} < f_k, \\ 01 & \text{if } r_{k+1} < f_{k+1}, \\ 11 & \text{otherwise} \end{cases}$$

$$r_{k-1} = \begin{cases} r_{k+1} & \text{if } r_{k+1} < f_k, \\ r_{k+1} - f_k & \text{if } r_{k+1} < f_{k+1}, \\ r_{k+1} - f_{k+1} & \text{otherwise} \end{cases}$$

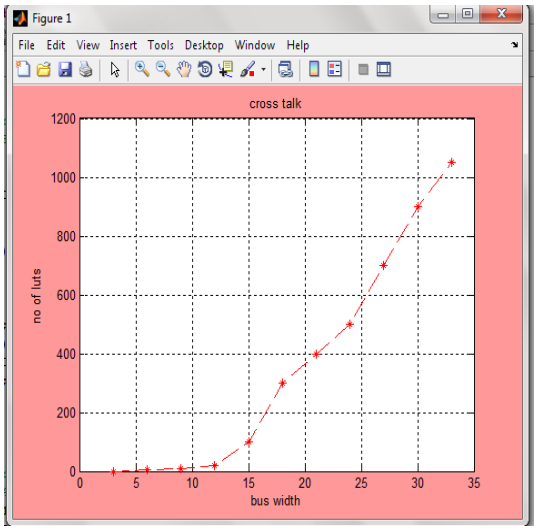

Fig 3. Encoder size with a brute-force implementation

Above Equation can be implemented using *two* adders and *two* MUXes. Since the single-bit stage implementation requires *three* adders and *two* MUXes to encode two bits, this two-bit stage implementation is simpler. We should point out that this simplification is only achieved when "10" is the prohibited pattern. We can not reduce the implementation complexity if"01"is the prohibited pattern in the two bit implementation. For comparison, the FPF CODEC proposed in [1] need one adder, one comparator and one MUX for each stage, almost twice the complexity of the FTF CODEC with two-bit implementation. The decoders are identical for both FTF and FPF.

To evaluate the complexity of the CODECs, we synthesized the CODEC in a 90nm process [5]. Figure 4 plots the equivalent gate counts of the encoder for input bus widths from 4 to 32. For 12-bit input data width, the equivalent number of 2-input gates is 245. This is nearly two orders of magnitude lower than the gate count reported in Figure 3. For the input data width of 32-bit, the equivalent gate count is 2247. The growth of the encoder sizes is quadratic with respect to the bus size, as we expected.

For comparison, the gate counts of the FPF-CAC encoders obtained through synthesis [1] are also plotted in Figure 3.

The FPF-CAC encoder gate count for a 32-bit bus is 2640, which is 17.6% bigger than the FTF-CAC encoder for the same bus size. On average, the gate count for the FPF encoder is ~ 17% higher that the FTF-CAC encoder. The decoders for both codes are identical.

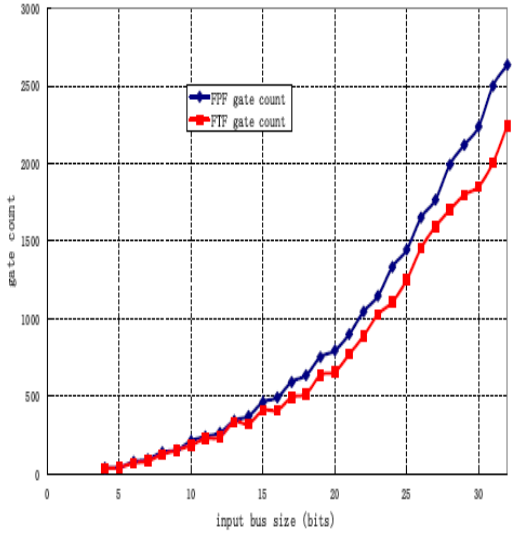

Fig 4.Encoder gate count comparison

Without pipelining, the overall delay of the encoder is the summation of all the stage delays. This total delay can be be significant. Fortunately, our design allows pipeline stages to be easily inserted between stages.

The speed of the encoder is determined by the slowest stage, the MSB stage. It is reported in [16] that a 64-bit adder has a total delay of less than 250ps using a 65nm process. With additional delay from the MUX, we estimate that the slowest stage of our CODEC, the MSB stage, has a delay of no more than 300ps.

The complexity and speed can be further improved by applying bus partitioning. The total area has the quadratic relation with the number of input bits and therefore partitioning the bus will reduce the total area by close to 50%.

Unlike FPF CODECs, which require either two shield wires between the group boundaries or some group complement logic for bus partitioning, the FTF code only needs one grounded wire between two groups.

Fig 5. Simulation Result for Encoder

IV. CONCLUSIONS

In this work, we present a CODEC design for the FPFCAC based on the Fibonacci numeral system. Our analysis show that all numbers can be represented by FPF vectors in Fibonacci numeral system.

In this paper, we establish a framework for the CAC CODEC design based on numeral systems, and devise efficient CODECs for OLCs, FPCs, and FOCs by choosing appropriate numeral systems and constants. The results are summarized in Table I.

Our CODEC design has been verified through actual implementation. Encoders for bus size from 4 to 32 bit are implemented in a 90 nm CMOS process and the results show a 17% reduction in gate count over the FPF-CAC encoder for the same bus size [1]. The design can achieve high speed through pipelining.

Implementation results show that our CODECs all have area and delay that increase quadratically with the bus width. Used together with partial coding, our efficient CODECs help make CACs a viable option in combating crosstalk delay, which is a bottleneck in deep sub-micron system-on-chip designs.

V. ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers for their comments which were very helpful in improving the quality and presentation of this paper.

REFERENCES:

[1] K. Kim, K. Baek, N. Shanbhag, C. Liu, and S.-M. Kang, SCoupling driven

[2] S.R. Sridhara, A. Ahmed, and N. R. Shanbhag, "Area and Energy-Efficient Crosstalk Avoidance Codes for On-Chip busses", Proc. of ICCD, 2004, pp 12-17.

[3] C. Duan, A.Tirumala and S.P.Khatri, "Analysis and Avoidance of Cross-talk in On-chip Bus", HotInterconnects, 2001,pp 133-138.

[4] Bret Victor and K. Keutzer,"Bus Encoding to Prevent Crosstalk Delay", ICCAD, 2001, pp 57-63.

[5] C. Duan, K. Gulati and S. P. Khatri, "Memory-based Cross-talk Canceling CODECs for On-chip busses", ISCAS 2006, pp 4-9.

[6] Signal encoding scheme for low-power interface design,"T Proc. of IEEE/ACM International Conference on Computer-Aided Design, Nov 2000.

[7] P. P. Sotiriadis, "Interconnect modeling and optimization in deep submicron technologies," Ph.D. dissertation, Dept. Elect. Eng. Comput. Sci., Massachusetts Inst. Technol., Cambridge, 2002.

[8] K. Hirose and H. Yasuura, "A bus delay reduction technique considering crosstalk," in Proc. Des. Autom. Test Eur. Conf. Exhibition, 2000, pp. 441–445.

[9] B. Victor, "Bus encoding to prevent crosstalk delay," M.S. thesis, Dept. Elect. Eng. Comput. Sci., Univ. California, Berkeley, 2001.

[10] B. Victor and K. Keutzer, "Bus encoding to prevent crosstalk delay," in Proc. IEEE/ACM Int. Conf. Comput.-Aided Des., 2001, pp. 57–63.

[11] P. P. Sotiriadis and A. Chandrakasan, "Reducing bus delay in submicron technology using coding," in Proc. Conf. Asia South Pacific Des. Autom., 2001, pp. 109–114.

Authors Profile:

K.Vijaya Lakshmi is pursuing M.Tech in Nalanda institute of engineering and technology with specialization embedded systems.

I.V.G Manohar M.Tech

I.V.G Manohar is an associate professor of electronics and communication engineering department in Nalanda institute of engineering and technology

L.Srinivas is an assistant professor and Head of the ECE department in nalanda institute of engineering and technology. He got his M.Tech from Narasaraopet Engineering College in 2010.His Area of interest is communication field

◈◈◈