Bit-Stuffing Algorithms for Crosstalk Avoidance in High-Speed Switching

Cheng-Shang Chang, Jay Cheng, Tien-Ke Huang, Xuan-Chao Huang, Duan-Shin Lee, and Chao-Yi Chen
Institute of Communications Engineering
National Tsing Hua University
Hsinchu 30013, Taiwan, R.O.C.
Email: cschang@ee.nthu.edu.tw; jcheng@ee.nthu.edu.tw; d915601@oz.nthu.edu.tw; d9761812@oz.nthu.edu.tw; lds@cs.nthu.edu.tw; g9764501@oz.nthu.edu.tw

Abstract—The crosstalk effect is one of the main problems in deep sub-micron designs of high-speed buses. To mitigate the crosstalk effect, there are several types of crosstalk avoidance codes proposed in the literature. In this paper, we are particularly interested in generating forbidden transition codes that do not have opposite transitions on any two adjacent wires. For this, we propose a *sequential bit-stuffing* algorithm and a *parallel bit-stuffing* algorithm. For the sequential bit-stuffing algorithm, we perform a worst-case analysis and a probabilistic analysis. We show by both theoretic analysis and simulations that the coding rate of the sequential bit-stuffing encoding scheme is quite close to the Shannon capacity. In particular, for a bus with $n = 10$ parallel wires, the difference is only 2.2%. Using a Markov chain analysis, we show that the coding rate of the parallel bit-stuffing algorithm is only slightly lower than that of the sequential bit-stuffing algorithm. The implementation complexity of the parallel bit-stuffing algorithm is linear with n. In comparison with the existing forbidden transition codes that use the Fibonacci representation in the literature, our bit-stuffing algorithms not only achieve higher coding rates but also have much lower implementation complexity.

Index Terms—Bit-stuffing, high-speed switching, bus encoding, crosstalk.

1 Introduction

Many electronic high-speed switch architectures, including shared medium switches and crossbar switches, rely on information exchange through high-speed buses. As the VLSI technology advances, it is possible to pack more wires in high-speed buses and attach more input/output ports to them. However, the propagation delay through long on-chip buses has become a serious issue in deep sub-micron designs [1] as the crosstalk effect due to the coupling capacitance between adjacent wires in the buses could be detrimental. To mitigate the crosstalk effect, it is suggested in the literature that one should avoid certain patterns and transitions on the wires [1], and there have been several bus encoding schemes proposed for this. Among them, we are particularly interested in designing coding schemes that avoid "opposite transitions" on any two adjacent wires [1]–[10]. Specifically, a transition on a wire is either a 0 followed by a 1 (a $0 \to 1$ transition) or a 1 followed by a 0 (a $1 \to 0$ transition), and opposite transitions on adjacent wires means that there is a $0 \to 1$ transition on one wire and there is a $1 \to 0$ transition on the other wire. A coding scheme without opposite transitions on any two adjacent wires is called a *forbidden transition code* in the literature [1]–[10].

The simplest way to avoid opposite transitions on any two adjacent wires is to have the even-numbered wires transmit 0 all the time and only use the odd-numbered wires for data transmission. Such a scheme is known as "ground shielding" in [2]. As only half of the wires are used for data transmission, its coding rate (throughput) is only 50%. In [3], Victor and Keutzer showed that there exist forbidden transition codes that achieve the coding rate $\log_2 \frac{1+\sqrt{5}}{2} \approx 0.6942$ when the number of wires is sufficiently large. A recursive construction of such a code was given in [4] by using the "Fibonacci representation." Such a Fibonacci representation for forbidden transition codes was also previously addressed in [5]. Recently, an explicit encoding scheme was proposed in [6] to map every input number to its Fibonacci representation for the forbidden transition code in [4]. An alternative mapping that can be used for both forbidden transition codes and forbidden overlap codes was shown in [10]. However, the overall complexity of the encoders in [6], [10] is $O(n^2)$ for a bus with n wires (as there are $n-1$ sequential stages in the encoder and each stage requires an $O(n)$-bit comparator). To further improve the coding rate, two-dimensional forbidden transition codes with block length larger than 1 were proposed in [7], and it was shown that the coding rate could be increased to more than 80%. However, no explicit constructions of such codes (and the associated encoders/decoders) were given in [7].

Our main contribution in this paper is to propose sim-

Part of the results in this paper were presented at the IEEE Annual Conference on Computer Communications (INFOCOM'10), San Diego, CA, USA, March 15–19, 2010.

ple *bit-stuffing* algorithms for generating forbidden transition codes and develop their associated analyses. Bit-stuffing algorithms are commonly used in networking to encode the data bit stream to avoid specific patterns [11], e.g., frame delimiters and tokens in token rings. They also have many applications in digital recording, e.g., runlength-limited encoding [12], [13], [14]. The success of the bit-stuffing algorithms is mainly due to their simplicity as they only have linear complexity. Despite the fact that there exist many previous works on bit-stuffing algorithms, it seems (to the best of our knowledge) that our work in this paper is the first to use the bit-stuffing algorithm for bus encoding to avoid opposite transitions on any two adjacent wires.

In this paper, we first propose a sequential bit-stuffing algorithm for the bus encoding problem. The idea of our sequential bit-stuffing algorithm is very simple. We consider a bus with n wires and index the n wires from 1 to n. When there is a $0 \to 1$ (resp., $1 \to 0$) transition on the $(i-1)^{\text{th}}$ wire and the previous bit on the i^{th} wire is 1 (resp., 0), then a redundant bit 1 (resp., 0) is inserted on the i^{th} wire by the encoder so that no opposite transitions can occur on any two adjacent wires. On the other hand, when the decoder receives a $0 \to 1$ (resp., $1 \to 0$) transition on the $(i-1)^{\text{th}}$ wire and the previous bit on the i^{th} wire is 1 (resp., 0), then the decoder knows the bit on the i^{th} wire is a *stuffed* bit and it is removed by the decoder. For our sequential bit-stuffing algorithm, we first show through the worst-case analysis that the asymptotic coding rate for any input data bit stream is at least $\frac{n+1}{2n}$ (more than 50%). For the probabilistic analysis, we assume that the input data bit stream is a sequence of independent and identically distributed (i.i.d.) Bernoulli random variables with equal probabilities of being 0 or 1. Under our sequential bit-stuffing algorithm, the sequence of the n-vectors transmitted on the n wires can be modeled as a finite, irreducible, and aperiodic (ergodic) Markov chain. Therefore, there exist unique steady state probabilities for the Markov chain and that in turn can be used to compute the asymptotic coding rate when n is small. For $1 \le n \le 10$, our numerical results show that the asymptotic coding rate for the i.i.d. Bernoulli input data bit stream is more than 82% and is quite close to the Shannon capacity [15]–[16]. In particular, for the case with $n = 10$, the difference is only 2.2%. Computing the steady state probabilities for large n appears to be numerically demanding as the number of states increases exponentially with n. For this, we find that the asymptotic coding rate can be approximated by $2\sqrt{2} - 2 \approx 0.8284$ for large n, which matches extremely well to our simulation results.

For the sequential bit-stuffing algorithm, scalability might be a problem as it only deals with one input data bit stream. In practice, it is quite common to have parallel data bit streams. To solve this problem, we take one step further by proposing a parallel bit-stuffing algorithm that can generate a forbidden transition code for a bus with n wires and n parallel data bit streams. Our first idea for parallel encoding is to allow data bit streams to be transmitted directly on odd-numbered wires and perform bit-stuffing on even-numbered wires. Like the ground shielding scheme in [2], such an idea provides the needed isolation on even-numbered wires so that bit-stuffing can be carried out in parallel. As in the sequential bit-stuffing algorithm, the implementation complexity of the parallel bit-stuffing algorithm is only $O(n)$ for a bus with n parallel wires. By assuming i.i.d. Bernoulli input data bit streams, we show that the coding rate for our parallel bit-stuffing algorithm is 0.8125, which is only slightly lower than the coding rate 0.8284 achieved by using the sequential bit-stuffing algorithm.

One problem that comes from the parallel bit-stuffing algorithm is that the coding rates for even-numbered wires are significantly lower than those for odd-numbered wires. When there are only a finite number of data bits in every data stream, one has to pad additional bits on odd-numbered wires. To solve the uneven rate problem, our second idea for the parallel bit-stuffing algorithm is *rate-balancing* that adds 2×2 crossbar switches before the encoder. The connection patterns of all the 2×2 crossbar switches are synchronized and set periodically between the "bar" state and the "cross" state. By so doing, every input data stream is connected alternatively to an even-numbered wire and an odd-numbered wire. We show by simulation that the parallel bit-stuffing algorithm with rate-balancing significantly reduces the number of padded bits.

One key message of this paper is that there exist very simple encoding/decoding schemes to generate forbidden transition codes that achieve coding rates near the Shannon capacity. As such, bus encoding for crosstalk avoidance in buses with a large number of wires might be feasible. This is quite contrary to the common belief in the literature. For example, in [8] it was shown that there do not exist linear forbidden transition codes and it was suggested that bus encoding for crosstalk avoidance should be the outermost code (along with error control coding and low-power coding). Furthermore, in the setting of high-performance processors like superscalar and very long instruction word (VLIW) architecture, a "segment-stuffing" algorithm was proposed in [9], where a segment of all 0's (or all 1's) is inserted whenever there are transition violations in the data segments. Our bit-stuffing algorithm improves the segment-stuffing algorithm in [9] by reducing the segment size to 1 bit and eliminating the need for shielding between adjacent segments.

As for the applicability of our work, we note there are some general issues for using bit-stuffing algorithms: (i) they are not *memoryless* and error propagation should be contained, (ii) they require additional buffer and delay for encoding/decoding and thus not suitable for delay-sensitive applications, and (iii) the codes are of variable lengths and thus not suitable for applications that require fixed-length codes. In view of all these, one possible application for our work is transmitting *variable length packets* through high-speed buses in packet switches. In such an application, bit errors are contained in a single packet. Also, buffering is required for a stored-and-forward network anyway and the additional encoding/decoding delay could be relatively small when compared with queueing delay.

The rest of this paper is organized as follows. In Section 2, we model a bus under the constraint that there are no opposite transitions on any two adjacent wires as a forbidden transition channel, and obtain the channel capacity of such a channel that serves as the fundamental limit on the coding rate of any forbidden transition code. In Section 3, we present our sequential bit-stuffing algorithm for generating forbidden transition codes. We identify a worst-case input data bit stream for the sequential bit-stuffing algorithm in Section 3.1 and perform a probabilistic analysis in Section 3.2. An approximate probabilistic analysis for a channel with a large number of wires is developed in Section 3.3. To cope with the scalability issue, we propose our parallel bit-stuffing algorithm in Section 4. In Section 4.1, we derive the coding rates of the parallel bit-stuffing algorithm. The idea of rate-balancing is proposed in Section 4.2. The paper is concluded in Section 5, where we address possible extensions of our work.

2 Forbidden Transition Channels

In this paper, we consider the discrete-time setting and index time by $t = 1, 2, \ldots$. We model a bus under the constraint that there are no opposite transitions on any two adjacent wires as a forbidden transition channel defined as follows.

A *forbidden transition channel* with n parallel wires, indexed from 1 to n, is a channel that is capable of transmitting n binary sequences through the n parallel wires as long as there are no opposite transitions on any two adjacent wires. Specifically, let $c_i(t)$, $i = 1, 2, \ldots, n$ and $t = 1, 2, \ldots$, be the bit transmitted on the i^{th} wire at time t in a forbidden transition channel with n parallel wires. Then for all $i = 2, 3, \ldots, n$ and $t = 2, 3, \ldots$, we have

$$\begin{bmatrix} c_{i-1}(t-1) & c_{i-1}(t) \\ c_i(t-1) & c_i(t) \end{bmatrix} \neq \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \text{ or } \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \quad (1)$$

Let $\bar{a}$ be the complement of a, i.e., $\bar{a} = 1$ if $a = 0$ and $\bar{a} = 0$ if $a = 1$. Then it is easy to see that the constraint in (1) is equivalent to the constraint that there exist no $2 \leq i \leq n$ and $t \geq 2$ such that

$$\bar{c}_{i-1}(t-1) = c_{i-1}(t) = c_i(t-1) = \bar{c}_i(t). \quad (2)$$

Denote $\mathbf{c}(t) = (c_n(t), c_{n-1}(t), \ldots, c_1(t))$, $t = 1, 2, \ldots$, as the n-vector transmitted on the n parallel wires at time t. Let $X_n(t)$, $t = 1, 2, \ldots$, be the number of sequences $(\mathbf{c}(1), \mathbf{c}(2), \ldots, \mathbf{c}(t))$ that satisfy the constraint in (1) up to time t. Then it is well known [15]–[16] that the channel capacity of a forbidden transition channel with n parallel wires is the entropy rate of these constrained sequences (see e.g., pp. 94 of [16]) and it is given by

$$C_n = \frac{1}{n} \lim_{t \to \infty} \frac{\log_2 X_n(t)}{t} \text{ bits/wire/time unit.} \quad (3)$$

Note that the channel capacity C_n serves as the fundamental limit on the coding rate of any forbidden transition code.

The channel capacity C_n is related to the *adjacency matrix* A_n defined by $(A_n)_{\mathbf{c},\mathbf{c}'} = 1$ if there exists no $2 \leq i \leq n$ such that $\bar{c}_{i-1} = c'_{i-1} = c_i = \bar{c}'_i$ and $(A_n)_{\mathbf{c},\mathbf{c}'} = 0$ otherwise, where $\mathbf{c} = (c_n, c_{n-1}, \ldots, c_1)$ and $\mathbf{c}' = (c'_n, c'_{n-1}, \ldots, c'_1) \in \{0,1\}^n$. In other words, if $\mathbf{c}(t-1) = \mathbf{c}$ and $\mathbf{c}(t) = \mathbf{c}'$, then $(A_n)_{\mathbf{c},\mathbf{c}'} = 1$ means that the constraint in (1) is satisfied at time t, and $(A_n)_{\mathbf{c},\mathbf{c}'} = 0$ means that the constraint in (1) is not satisfied at time t. We assume that the states $\mathbf{c} = (c_n, c_{n-1}, \ldots, c_1) \in \{0,1\}^n$ are ordered in the standard lexicographic order, i.e., in the increasing order of the integers $\sum_{i=1}^{n} c_i 2^{i-1} \in \{0, 1, \ldots, 2^n - 1\}$.

In the following theorem, we show that the channel capacity C_n can be expressed in terms of the maximum eigenvalue $\lambda_{n,\max}$ of the adjacency matrix A_n. Its proof is given in Appendix A. Similar theorems were also previously reported for the capacities of various constrained source coding problems (see e.g., [17], [18]).

Theorem 1 *The channel capacity C_n of a forbidden transition channel with n parallel wires is given by*

$$C_n = \frac{1}{n} \log_2 \lambda_{n,\max}, \quad (4)$$

where $\lambda_{n,\max}$ is the maximum eigenvalue of the adjacency matrix A_n.

In Table 1, we show the channel capacity C_n of a forbidden transition channel with n parallel wires for $1 \leq n \leq 10$.

For large n, the computation of the capacities becomes much more difficult. However, there are methods of obtaining bounds for the capacities of various constrained source coding problems in the literature. In particular, the Calkin-Wilf lower bound (see e.g., [19], [18], [20]) can be used here for bounding the capacity when $n \to \infty$. Specifically, for any positive integer p and nonnegative integer q,

$$C_\infty = \lim_{n \to \infty} C_n \geq \frac{\log_2 \lambda_{p+2q+1,\max} - \log_2 \lambda_{2q+1,\max}}{p}. \quad (5)$$

Choosing $p = q = 3$, we have from Table 1 and (4) that

$$C_\infty \geq \frac{\log_2 \lambda_{10,\max} - \log_2 \lambda_{7,\max}}{3} \geq 0.848. \quad (6)$$

TABLE 1
The channel capacity C_n given by (4) for $1 \leq n \leq 10$.

n	1	2	3	4	5
C_n	1	0.9163	0.8941	0.8826	0.8757

n	6	7	8	9	10
C_n	0.8712	0.8679	0.8654	0.8635	0.8620

3 Sequential Bit-Stuffing Algorithm

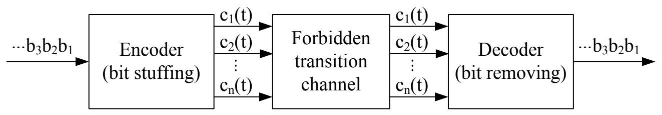

Fig. 1. A forbidden transition channel with n parallel wires and with a bit-stuffing encoder and a bit-removing decoder.

Consider a forbidden transition channel with n parallel wires (see Figure 1). Let $\{b_1, b_2, \ldots\}$ be the input data bit stream. If we transmit the input data bit stream directly through the forbidden transition channel such that the bit transmitted on the i^{th} wire at time t is $c_i(t) = b_{n(t-1)+i}$ for $i = 1, 2, \ldots, n$ and $t = 1, 2, \ldots$, then it is possible that the constraint in (1) cannot be satisfied and thus the input data bit stream cannot be reliably transmitted through the forbidden transition channel.

In the following, we propose a sequential bit-stuffing algorithm to encode the input data bits $b_1, b_2, \ldots$ so that the coded bits $c_i(t)$, $i = 1, 2, \ldots, n$ and $t = 1, 2, \ldots$, satisfy the constraint in (1) and hence the coded bits can be reliably transmitted through the forbidden transition channel. The idea of our sequential bit-stuffing algorithm is to add a redundant bit on the i^{th} wire at time t if $\bar{c}_{i-1}(t-1) = c_{i-1}(t) = c_i(t-1)$, and in that case the redundant bit $c_i(t)$ is set as $c_i(t) = c_{i-1}(t)$ so that the coded bits $c_i(t)$, $i = 1, 2, \ldots, n$ and $t = 1, 2, \ldots$, always satisfy the constraint in (1).

Algorithm 2 ***(Encoder: the sequential bit-stuffing algorithm)*** *Given an input data bit stream $\{b_1, b_2, \ldots\}$. Initially, set $c_i(1) = b_i$ for $i = 1, 2, \ldots, n$. For every time $t \geq 2$, generate the coded bit $c_i(t)$, $i = 1, 2, \ldots, n$, by the following bit-stuffing rule:*

(i) Set $c_1(t)$ as the next input data bit.

(ii) For $i = 2, 3, \ldots, n$,

- (a) *(Bit-stuffing condition) If $\bar{c}_{i-1}(t-1) = c_{i-1}(t) = c_i(t-1)$, then $c_i(t)$ is a stuffed bit and we set $c_i(t) = c_{i-1}(t)$.*
- (b) *Otherwise, set $c_i(t)$ as the next input data bit.*

Under the sequential bit-stuffing algorithm in Algorithm 2, it is clear that the coded bits $c_i(t)$, $i = 1, 2, \ldots, n$ and $t = 1, 2, \ldots$, satisfy the constraint in (1). Furthermore, for $i = 1$ or $t = 1$, the coded bit $c_i(t)$ is always a data bit. For $2 \leq i \leq n$ and $t \geq 2$, the coded bit $c_i(t)$ is a stuffed bit if and only if

$$\bar{c}_{i-1}(t-1) = c_{i-1}(t) = c_i(t-1) = c_i(t). \tag{7}$$

As we assume that there are no transmission errors in a forbidden transition channel as long as there are no forbidden transitions, it follows that the original input data bits $b_1, b_2, \ldots$ can be decoded from the coded bits $c_i(t)$, $i = 1, 2, \ldots, n$ and $t = 1, 2, \ldots$, by simply removing the coded bit $c_i(t)$ whenever $2 \leq i \leq n$, $t \geq 2$, and the stuffed bit condition $\bar{c}_{i-1}(t-1) = c_{i-1}(t) = c_i(t-1) = c_i(t)$ in (7) is satisfied. This is described in the following bit-removing algorithm.

Algorithm 3 ***(Decoder: the bit-removing algorithm)*** *Given received coded bits $c_i(t)$, $i = 1, 2, \ldots, n$ and $t = 1, 2, \ldots$. Initially, set $b_i = c_i(1)$ for $i = 1, 2, \ldots, n$. For every time $t \geq 2$, decode the received coded bit $c_i(t)$, $i = 1, 2, \ldots, n$, by the following bit-removing rule:*

(i) Decode $c_1(t)$ as the next data bit.

(ii) For $i = 2, 3, \ldots, n$,

- (a) *(Bit-removing condition) If $\bar{c}_{i-1}(t-1) = c_{i-1}(t) = c_i(t-1) = c_i(t)$, then $c_i(t)$ is a stuffed bit and it is discarded.*
- (b) *Otherwise, decode $c_i(t)$ as the next data bit.*

Unlike the fixed-length memoryless encoder that uses the Fibonacci representation in [3], [4], and [6], we note that our sequential bit-stuffing algorithm is a variable-length encoder with memory. This is because our sequential bit-stuffing algorithm requires the knowledge of the already coded bits in encoding the next input data bit, and it encodes an input data bit either by one bit (the data bit itself) or by two bits (duplicating the data bit twice by inserting a stuffed bit that is the same as the data bit when the bit-stuffing condition in Algorithm 2(ii-a) is met).

Under the sequential bit-stuffing algorithm, we have the following *no adjacent stuffed bits (both in space and in time)* property. This is one of the most important properties of our sequential bit-stuffing algorithm and will be used in the analysis of our sequential bit-stuffing algorithm in the rest of this paper.

Lemma 4 ***(No adjacent stuffed bits property)*** *If the coded bit $c_i(t)$ is a stuffed bit, then its four adjacent coded bits $c_i(t-1)$, $c_i(t+1)$, $c_{i-1}(t)$, and $c_{i+1}(t)$ (in the case that $i \leq n-1$) cannot be stuffed bits, i.e., they are all data bits.*

Proof. If the coded bit $c_i(t)$ is a stuffed bit, then it follows from (7) that $2 \leq i \leq n$, $t \geq 2$, and

$$\bar{c}_{i-1}(t-1) = c_{i-1}(t) = c_i(t-1) = c_i(t). \tag{8}$$

In the following, we show by contradiction that $c_i(t-1)$ cannot be a stuffed bit. The proof that $c_i(t+1)$, $c_{i-1}(t)$, and $c_{i+1}(t)$ (in the case that $i \leq n-1$) cannot be stuffed bits is similar.

Suppose that $c_i(t-1)$ is a stuffed bit. Then it follows from (7) (with t replaced by $t-1$) that $2 \leq i \leq n$, $t-1 \geq 2$, and

$$\bar{c}_{i-1}(t-2) = c_{i-1}(t-1) = c_i(t-2) = c_i(t-1). \tag{9}$$

As we have $c_i(t-1) = \bar{c}_{i-1}(t-1)$ in (8) and $c_i(t-1) = c_{i-1}(t-1)$ in (9), a contradiction is reached. ■

3.1 Worst-Case Analysis

In the following theorem, we derive a tight lower bound on the asymptotic coding rate of our sequential bit-stuffing encoding scheme.

Theorem 5 *The asymptotic coding rate R_n of our sequential bit-stuffing encoding scheme for any input data bit stream over a forbidden transition channel with n parallel wires is lower bounded by*

$$R_n \geq \frac{n+1}{2n} \text{ bits/wire/time unit.} \tag{10}$$

Furthermore, the lower bound in (10) is tight as there exists an input data bit stream that achieves the lower bound. In other words, the worst-case asymptotic coding rate R_n^* of our sequential bit-stuffing encoding scheme is given by $R_n^* = \frac{n+1}{2n}$.

Proof. (i) Consider an input data bit stream $\{b_1, b_2, \ldots\}$. From the sequential bit-stuffing algorithm in Algorithm 2, it is clear that the n coded bits at time $t = 1$ and all of the coded bits on the first wire are data bits, i.e., $c_i(t)$ is a data bit for $i = 1$ or $t = 1$. From the "no adjacent stuffed bits" property in Lemma 4, it is easy to see that at least one of the two coded bits $c_i(t)$ and $c_i(t+1)$ is a data bit for $i = 2, 3, \ldots, n$ and $t = 2, 3, \ldots$, and at least one of the two coded bits $c_i(t)$ and $c_{i+1}(t)$ is a data bit for $i = 2, 3, \ldots, n-1$ and $t = 2, 3, \ldots$. As such, among the $2(n-1)$ coded bits $c_2(t), c_2(t+1), c_3(t), c_3(t+1), \ldots, c_n(t), c_n(t+1)$, there are at least $n-1$ data bits for $t = 2, 3, \ldots$, and among the $n-1$ coded bits $c_2(t), c_3(t), \ldots, c_n(t)$, there are at least $\lfloor \frac{n-1}{2} \rfloor$ data bits for $t = 2, 3, \ldots$.

Let $R_n(t)$ be the number of data bits among the first nt coded bits $c_i(t')$, $i = 1, 2, \ldots, n$ and $t' = 1, 2, \ldots, t$. Then we have

$$R_n(t) \geq \begin{cases} n + (t-1) + \frac{t-1}{2}(n-1) \\ = \frac{(n+1)t}{2} + \frac{n-1}{2}, \quad \text{if } t \text{ is odd,} \\ n + (t-1) + \frac{t-2}{2}(n-1) + \lfloor \frac{n-1}{2} \rfloor \\ = \frac{(n+1)t}{2} + \lfloor \frac{n-1}{2} \rfloor, \text{ if } t \text{ is even.} \end{cases} \tag{11}$$

It follows from (11) that

$$R_n = \frac{1}{n} \lim_{t\to\infty} \frac{R_n(t)}{t} \geq \frac{n+1}{2n}.$$

(ii) Now, we give an input data bit stream that achieves the lower bound in (10). As the input data bits can be uniquely decoded from their coded bits under the sequential bit-stuffing encoding scheme, it suffices to show the coded bits of the input data bits.

Let $\mathbf{w} = (1,1,0,0,1,1,0,0,\ldots)$, $\mathbf{x} = (1,0,0,1,1,0,0,1,\ldots)$, $\mathbf{y} = (0,0,1,1,0,0,1,1,\ldots)$, and $\mathbf{z} = (0,1,1,0,0,1,1,0,\ldots)$ be four infinite binary periodic sequences with period 4. For $1 \leq i \leq n$, let

$$(c_i(1), c_i(2), \ldots) = \begin{cases} \mathbf{w}, & \text{if } i = 1 \pmod 4, \\ \mathbf{x}, & \text{if } i = 2 \pmod 4, \\ \mathbf{y}, & \text{if } i = 3 \pmod 4, \\ \mathbf{z}, & \text{if } i = 0 \pmod 4. \end{cases} \tag{12}$$

In Table 2, we show the coded bits $c_i(t)$ given by (12) for $1 \leq i \leq n$ and $1 \leq t \leq 12$, where $n = 12$. It is clear that $c_i(t)$ given by (12) is periodic with period 4 (both in space and in time).

TABLE 2
The coded bits $c_i(t)$ given by (12) for $1 \leq i \leq n$ and $1 \leq t \leq 12$, where $n = 12$. Note that the stuffed bits are *italicized*.

$i \setminus t$	1	2	3	4	5	6	7	8	9	10	11	12
1	1	1	0	0	1	1	0	0	1	1	0	0
2	1	0	*0*	1	*1*	0	*0*	1	*1*	0	*0*	1
3	0	*0*	1	*1*	0	*0*	1	*1*	0	*0*	1	*1*
4	0	1	*1*	0	*0*	1	*1*	0	*0*	1	*1*	0
5	1	*1*	0	*0*	1	*1*	0	*0*	1	*1*	0	*0*
6	1	0	*0*	1	*1*	0	*0*	1	*1*	0	*0*	1
7	0	*0*	1	*1*	0	*0*	1	*1*	0	*0*	1	*1*
8	0	1	*1*	0	*0*	1	*1*	0	*0*	1	*1*	0
9	1	*1*	0	*0*	1	*1*	0	*0*	1	*1*	0	*0*
10	1	0	*0*	1	*1*	0	*0*	1	*1*	0	*0*	1
11	0	*0*	1	*1*	0	*0*	1	*1*	0	*0*	1	*1*
12	0	1	*1*	0	*0*	1	*1*	0	*0*	1	*1*	0

It is easy to see that $c_i(t)$, $i = 1, 2, \ldots, n$ and $t = 1, 2, \ldots$, are valid coded bits as there are no opposite transitions on any two adjacent wires. Furthermore, $c_i(t)$ is a stuffed bit if and only if $i \geq 2$, $t \geq 2$, and $i+t$ is odd (note that the stuffed bits in Table 2 are *italicized*). As such, the number of data bits $R_n(t)$ among the first nt coded bits is given by the right-hand side of (11). It follows that

$$R_n = \frac{1}{n} \lim_{t\to\infty} \frac{R_n(t)}{t} = \frac{n+1}{2n}.$$

The proof is completed. ■

3.2 Probabilistic Analysis

In Section 3.1, we have given a worst-case performance analysis for the sequential bit-stuffing encoding scheme in Algorithm 2. However, in most situations, the performance of the sequential bit-stuffing encoding scheme is much better than the worst-case scenario. For this, we give a probabilistic performance analysis for the sequential bit-stuffing encoding scheme in this section.

Assume that the input data bit stream $\{b_1, b_2, \ldots\}$ is a sequence of i.i.d. Bernoulli random variables with equal probabilities of being 0 or 1. Let $c_i(t)$, $i = 1, 2, \ldots, n$ and $t = 1, 2, \ldots$, be the coded bits under the sequential bit-stuffing encoding scheme, and let $\mathbf{c}(t) = (c_n(t), c_{n-1}(t), \ldots, c_1(t))$ for $t = 1, 2, \ldots$. Then it is clear from Algorithm 2 that $\mathbf{c}(t)$ is a function of $\mathbf{c}(t-1)$ and the input data bits that have not been encoded by time t. As such, given $\mathbf{c}(t-1)$, $\mathbf{c}(t)$ is conditionally independent of $\mathbf{c}(1), \mathbf{c}(2), \ldots, \mathbf{c}(t-2)$. This shows that the stochastic process $\{\mathbf{c}(t), t \geq 1\}$ is a time-homogeneous Markov chain. The transition probability matrix P_n of the Markov chain $\{\mathbf{c}(t), t \geq 1\}$ is given by

$$(P_n)_{\mathbf{c},\mathbf{c}'} = P(\mathbf{c}(t) = \mathbf{c}' | \mathbf{c}(t-1) = \mathbf{c}), \tag{13}$$

for $\mathbf{c}, \mathbf{c}' \in \{0,1\}^n$ and for all $t = 2, 3, \ldots$.

As (i) $c_1(t)$ is a data bit for $t = 1, 2, \ldots$, (ii) the constraint in (1) and the stuffed bit condition in (7) are satisfied, and (iii) the input data bits are i.i.d. Bernoulli random variables with equal probabilities of being 0 or

TABLE 3
The values of $q(c_i, c_{i-1}, c'_i, c'_{i-1})$.

$c_i c_{i-1} \setminus c'_i c'_{i-1}$	00	01	10	11
00	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
01	1	$\frac{1}{2}$	0	$\frac{1}{2}$
10	$\frac{1}{2}$	0	$\frac{1}{2}$	1
11	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$

1, it then follows from the chain rule for conditional probability that

$$\begin{aligned}(P_n)_{\mathbf{c},\mathbf{c}'} &= P(\mathbf{c}(t) = \mathbf{c}' | \mathbf{c}(t-1) = \mathbf{c}) \\ &= P(c_1(t) = c'_1 | \mathbf{c}(t-1) = \mathbf{c}) \\ &\quad \times \prod_{i=2}^{n} P(c_i(t) = c'_i | \mathbf{c}(t-1) = \mathbf{c}, c_{i-1}(t) = c'_{i-1}, \\ &\qquad\qquad c_{i-2}(t) = c'_{i-2}, \ldots, c_1(t) = c'_1) \\ &= \frac{1}{2} \prod_{i=2}^{n} P(c_i(t) = c'_i | c_i(t-1) = c_i, \\ &\qquad\qquad c_{i-1}(t-1) = c_{i-1}, c_{i-1}(t) = c'_{i-1}) \\ &= \frac{1}{2} \prod_{i=2}^{n} q(c_i, c_{i-1}, c'_i, c'_{i-1}), \end{aligned} \tag{14}$$

where we have denoted $q(c_i, c_{i-1}, c'_i, c'_{i-1})$ as the conditional probability $P(c_i(t) = c'_i | c_i(t-1) = c_i, c_{i-1}(t-1) = c_{i-1}, c_{i-1}(t) = c'_{i-1})$ which is a function of $c_i, c_{i-1}, c'_i, c'_{i-1}$ and is independent of t, and the values of $q(c_i, c_{i-1}, c'_i, c'_{i-1})$ are shown in Table 3.

For $n \geq 2$ and $\mathbf{c} = (c_n, c_{n-1}, \ldots, c_1) \in \{0,1\}^n$, we denote $\mathbf{c}^{(n-1)} = (c_{n-1}, c_{n-2}, \ldots, c_1) \in \{0,1\}^{n-1}$. From (14), we can see that

$$(P_n)_{\mathbf{c},\mathbf{c}'} = q(c_n, c_{n-1}, c'_n, c'_{n-1}) (P_{n-1})_{\mathbf{c}^{(n-1)},\mathbf{c}'^{(n-1)}}. \tag{15}$$

By using (15), one can easily show by induction that there is a recursive expression for the transition probability matrix P_n. Specifically, we let $E_0 = F_0 = G_0 = H_0 = \frac{1}{2}$ and O_i be the zero matrix of size $2^i \times 2^i$ for $i \geq 0$. Then

$$P_1 = \begin{bmatrix} E_0 & F_0 \\ G_0 & H_0 \end{bmatrix}, \tag{16}$$

and P_n, $n \geq 2$, can be recursively obtained as follows:

$$P_n = \begin{bmatrix} E_{n-1} & F_{n-1} \\ G_{n-1} & H_{n-1} \end{bmatrix}, \tag{17}$$

where

$$E_{n-1} = \begin{bmatrix} \frac{1}{2} E_{n-2} & \frac{1}{2} F_{n-2} \\ G_{n-2} & \frac{1}{2} H_{n-2} \end{bmatrix}, \tag{18}$$

$$F_{n-1} = \begin{bmatrix} \frac{1}{2} E_{n-2} & \frac{1}{2} F_{n-2} \\ O_{n-2} & \frac{1}{2} H_{n-2} \end{bmatrix}, \tag{19}$$

$$G_{n-1} = \begin{bmatrix} \frac{1}{2} E_{n-2} & O_{n-2} \\ \frac{1}{2} G_{n-2} & \frac{1}{2} H_{n-2} \end{bmatrix}, \tag{20}$$

$$H_{n-1} = \begin{bmatrix} \frac{1}{2} E_{n-2} & F_{n-2} \\ \frac{1}{2} G_{n-2} & \frac{1}{2} H_{n-2} \end{bmatrix}. \tag{21}$$

It is clear from (16)–(21) that $(P_n)_{\mathbf{0}_n,\mathbf{c}} > 0$ and $(P_n)_{\mathbf{c},\mathbf{0}_n} > 0$ for all $\mathbf{c} \in \{0,1\}^n$. It follows that the Markov chain $\{\mathbf{c}(t), t \geq 1\}$ is finite, irreducible, and aperiodic as its state space $\{0,1\}^n$ is finite, and every state $\mathbf{c} \in \{0,1\}^n$ can be reached from the state $\mathbf{0}_n$ and vice versa. As such, it is well known [24] that there exist unique steady state probabilities $\boldsymbol{\pi}_n = (\pi_{n,\mathbf{0}_n}, \ldots, \pi_{n,\mathbf{1}_n})$ for the Markov chain $\{\mathbf{c}(t), t \geq 1\}$ that could be obtained by solving the following system of linear equations:

$$\boldsymbol{\pi}_n = \boldsymbol{\pi}_n P_n, \tag{22}$$

$$\boldsymbol{\pi}_n \mathbf{1}_n^T = \sum_{\mathbf{c} \in \{0,1\}^n} \pi_{n,\mathbf{c}} = 1. \tag{23}$$

In the following lemma, we show a recursive expression for the steady state probabilities $\boldsymbol{\pi}_n = (\pi_{n,\mathbf{0}_n}, \ldots, \pi_{n,\mathbf{1}_n})$. Its proof is given in Appendix B.

Lemma 6 *(i)* ***(State aggregation property)*** *Let* $\boldsymbol{\pi}_n = (\boldsymbol{\pi}_n^{(0)}, \boldsymbol{\pi}_n^{(1)})$*, where* $\boldsymbol{\pi}_n^{(0)} = (\pi_{n,0\mathbf{0}_{n-1}}, \ldots, \pi_{n,0\mathbf{1}_{n-1}})$ *and* $\boldsymbol{\pi}_n^{(1)} = (\pi_{n,1\mathbf{0}_{n-1}}, \ldots, \pi_{n,1\mathbf{1}_{n-1}})$*. Then we have*

$$\boldsymbol{\pi}_{n-1} = \boldsymbol{\pi}_n^{(0)} + \boldsymbol{\pi}_n^{(1)}. \tag{24}$$

(ii) ***(State splitting property)*** *For* $n \geq 2$, $\boldsymbol{\pi}_n = (\boldsymbol{\pi}_n^{(0)}, \boldsymbol{\pi}_n^{(1)})$ *could be recursively obtained from* $\boldsymbol{\pi}_{n-1}$ *as follows:*

$$\boldsymbol{\pi}_n^{(0)} = \boldsymbol{\pi}_{n-1} G_{n-1} (I_{n-1} - E_{n-1} + G_{n-1})^{-1}, \tag{25}$$

$$\boldsymbol{\pi}_n^{(1)} = \boldsymbol{\pi}_{n-1} [I_{n-1} - G_{n-1} (I_{n-1} - E_{n-1} + G_{n-1})^{-1}], \tag{26}$$

where E_{n-1} *and* G_{n-1} *are given in (18) and (20), and* I_{n-1} *is the identity matrix of size* $2^{n-1} \times 2^{n-1}$*.*

For example, if $n = 1$, then we have from (22), and (23) that

$$P_1 = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix} \text{ and } \boldsymbol{\pi}_1 = \left(\frac{1}{2}, \frac{1}{2}\right).$$

If $n = 2$, then we have

$$P_2 = \begin{bmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{2} & \frac{1}{4} & 0 & \frac{1}{4} \\ \frac{1}{4} & 0 & \frac{1}{4} & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{bmatrix},$$

and it follows from (25) and (26) in Lemma 6 that

$$\begin{aligned}\boldsymbol{\pi}_2 &= (\boldsymbol{\pi}_1 G_1 (I_1 - E_1 + G_1)^{-1}, \\ &\quad \boldsymbol{\pi}_1 (I_1 - G_1 (I_1 - E_1 + G_1)^{-1})) \\ &= \left(\frac{3}{10}, \frac{2}{10}, \frac{2}{10}, \frac{3}{10}\right).\end{aligned}$$

Similarly, if $n = 3$, then we have

$$P_3 = \begin{bmatrix} \frac{1}{8} & \frac{1}{8} & \frac{1}{8} & \frac{1}{8} & \frac{1}{8} & \frac{1}{8} & \frac{1}{8} & \frac{1}{8} \\ \frac{1}{4} & \frac{1}{8} & 0 & \frac{1}{8} & \frac{1}{4} & \frac{1}{8} & 0 & \frac{1}{8} \\ \frac{1}{4} & 0 & \frac{1}{8} & \frac{1}{4} & 0 & 0 & \frac{1}{8} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{8} & \frac{1}{8} & 0 & 0 & \frac{1}{8} & \frac{1}{8} \\ \frac{1}{8} & \frac{1}{8} & 0 & 0 & \frac{1}{8} & \frac{1}{8} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{8} & 0 & 0 & \frac{1}{4} & \frac{1}{8} & 0 & \frac{1}{4} \\ \frac{1}{8} & 0 & \frac{1}{8} & \frac{1}{4} & \frac{1}{8} & 0 & \frac{1}{8} & \frac{1}{4} \\ \frac{1}{8} & \frac{1}{8} & \frac{1}{8} & \frac{1}{8} & \frac{1}{8} & \frac{1}{8} & \frac{1}{8} & \frac{1}{8} \end{bmatrix}$$

and

$$\begin{aligned}\boldsymbol{\pi}_3 &= (\boldsymbol{\pi}_2 G_2 (I_2 - E_2 + G_2)^{-1}, \\ &\quad \boldsymbol{\pi}_2 (I_2 - G_2 (I_2 - E_2 + G_2)^{-1})) \\ &= \left(\frac{125}{710}, \frac{82}{710}, \frac{60}{710}, \frac{88}{710}, \frac{88}{710}, \frac{60}{710}, \frac{82}{710}, \frac{125}{710} \right).\end{aligned}$$

Given that $\mathbf{c}(t-1) = \mathbf{c}$ and $\mathbf{c}(t) = \mathbf{c}'$, where $t \geq 2$, let $(D_n)_{\mathbf{c},\mathbf{c}'}$ be the number of data bits among the coded bits $c'_1, c'_2, \ldots, c'_n$. Since $\pi_{n,\mathbf{c}} = \lim_{t\to\infty} P(\mathbf{c}(t) = \mathbf{c})$, we immediately see that the average total number of data bits D_n transmitted over the n wires per time unit is given by

$$\begin{aligned}D_n &= \lim_{t\to\infty} \sum_{\mathbf{c},\mathbf{c}' \in \{0,1\}^n} P(\mathbf{c}(t-1) = \mathbf{c}, \mathbf{c}(t) = \mathbf{c}')(D_n)_{\mathbf{c},\mathbf{c}'} \\ &= \lim_{t\to\infty} \sum_{\mathbf{c},\mathbf{c}' \in \{0,1\}^n} P(\mathbf{c}(t-1) = \mathbf{c})(P_n)_{\mathbf{c},\mathbf{c}'}(D_n)_{\mathbf{c},\mathbf{c}'} \\ &= \sum_{\mathbf{c},\mathbf{c}' \in \{0,1\}^n} \pi_{n,\mathbf{c}}(P_n)_{\mathbf{c},\mathbf{c}'}(D_n)_{\mathbf{c},\mathbf{c}'}. \qquad (27)\end{aligned}$$

Furthermore, the asymptotic coding rate is given by $R_n = \frac{D_n}{n}$ for $n \geq 1$, and the average number of data bits r_n transmitted over the n^{th} wire per time unit is given by $r_1 = D_1$ and $r_n = D_n - D_{n-1}$ for $n \geq 2$.

In the following theorem, we show that D_n is equal to the entropy rate $H(P_n) = -\sum_{\mathbf{c},\mathbf{c}' \in \{0,1\}^n} \pi_{n,\mathbf{c}}(P_n)_{\mathbf{c},\mathbf{c}'} \cdot \log_2(P_n)_{\mathbf{c},\mathbf{c}'}$ of the Markov chain $\{\mathbf{c}(t), t \geq 1\}$ [16]. Its proof is given in Appendix C.

Theorem 7 *For a forbidden transition channel with n parallel wires, the average data transmission rate D_n over the n wires is given by*

$$D_n = -\sum_{\mathbf{c},\mathbf{c}' \in \{0,1\}^n} \pi_{n,\mathbf{c}}(P_n)_{\mathbf{c},\mathbf{c}'} \log_2(P_n)_{\mathbf{c},\mathbf{c}'} = H(P_n). \quad (28)$$

Furthermore, for $n \geq 2$, the average data transmission rate r_n over the n^{th} wire is given by

$$r_n = -\sum_{\mathbf{c},\mathbf{c}' \in \{0,1\}^n} \pi_{n,\mathbf{c}}(P_n)_{\mathbf{c},\mathbf{c}'} \log_2 q(c_n, c_{n-1}, c'_n, c'_{n-1}). \quad (29)$$

For example, if $n = 1$, then we have from (28) and (29) that $D_1 = H(P_1) = 1$, $R_1 = D_1 = 1$, and $r_1 = D_1 = 1$. If $n = 2$, then we have $D_2 = H(P_2) = 1.8$, $R_2 = \frac{D_2}{2} = 0.9$, and $r_2 = D_2 - D_1 = 1.8 - 1 = 0.8$. If $n = 3$, then we have $D_3 = H(P_3) = \frac{187}{71} \approx 2.6338$, $R_3 = \frac{D_3}{3} \approx 0.8779$, and $r_3 = D_3 - D_2 = \frac{187}{71} - 1.8 \approx 0.8338$.

In Table 4, we show the Shannon capacity C_n in (4), the average data transmission rate D_n over the n wires in (28), the coding rate $R_n = \frac{D_n}{n}$ of the sequential bit-stuffing encoding scheme, the average data transmission rate r_n over the n^{th} wire in (29), and the approximation of r_n in (33) (in Section 3.3) for $1 \leq n \leq 10$. Several important observations can be drawn from these numerical results: (i) There is still some gap between the coding rate R_n achieved by the sequential bit-stuffing algorithm and the Shannon capacity C_n. This shows that the sequential bit-stuffing algorithm does not achieve the Shannon capacity. However, the difference is very small. For the case with $n = 10$, the difference is only 2.2%. (ii) It seems that the average data transmission rate r_n over the n^{th} wire converges to a constant near 0.8284, i.e., $\lim_{n\to\infty} r_n \approx 0.8284$ (this will be further addressed in Section 3.3).

TABLE 4
The Shannon capacity C_n in (4), the average data transmission rate D_n over the n wires in (28), the coding rate $R_n = \frac{D_n}{n}$ of the sequential bit-stuffing encoding scheme, the average data transmission rate r_n over the n^{th} wire in (29), and the approximation of r_n in (33) (in Section 3.3) for $1 \leq n \leq 10$.

n	1	2	3	4	5
C_n	1	0.9163	0.8941	0.8826	0.8757
D_n	1	1.8	2.6338	3.4613	4.2899
R_n	1	0.9	0.8779	0.8653	0.8580
r_n	1	0.8	0.8338	0.8275	0.8286
r_n by (33)	1	0.8	0.8333	0.8276	0.8286

n	6	7	8	9	10
C_n	0.8712	0.8679	0.8654	0.8635	0.8620
D_n	5.1183	5.9467	6.7752	7.6036	8.4320
R_n	0.8531	0.8495	0.8469	0.8448	0.8432
r_n	0.8284	0.8284	0.8284	0.8284	0.8284
r_n by (33)	0.8284	0.8284	0.8284	0.8284	0.8284

3.3 An Approximate Probabilistic Analysis

To calculate the average data rate r_n transmitted over the n^{th} wire given by (29), we need to compute the steady state probabilities $\boldsymbol{\pi}_n$, by either directly solving (22)–(23), or recursively applying (25)–(26), which is numerically demanding for large n as the state space grows exponentially with n.

In this section, we make the following *Markov chain and conditional independence assumption*, and give an approximate probabilistic analysis for the calculation of r_n.

(A1) The stochastic processes $\{c_{n-1}(t),\ t \geq 1\}$ and $\{(c_n(t), c_{n-1}(t)),\ t \geq 1\}$ are Markov chains. Furthermore, given $c_{n-1}(t-1)$, the coded bits $c_{n-1}(t)$ and $c_n(t-1)$ are conditionally independent.

Such a conditional independence assumption was previously used by Picard for analyzing Markov fields [25]. Note that the assumption in (A1) is true for $n = 2$ and is only an approximation for $n > 2$. Specifically, for $n > 2$, $c_{n-1}(t)$ and $(c_n(t), c_{n-1}(t))$ are deterministic functions of $\mathbf{c}(t)$. Thus, the stochastic processes $\{c_{n-1}(t),\ t \geq 1\}$ and $\{(c_n(t), c_{n-1}(t)),\ t \geq 1\}$ are hidden Markov chains [16] and they are not necessarily Markov chains. However, as will be seen shortly, the assumption in (A1) leads to a very good approximation for r_n in the following theorem. The proof of Theorem 8 is given in Appendix D.

Theorem 8 *Suppose that the assumption in (A1) is true.*

(i) The transition probability matrix $P'_n = [(P'_n)_{c_{n-1},c'_{n-1}}]_{c_{n-1},c'_{n-1} \in \{0,1\}}$ *of the Markov chain*

$\{c_{n-1}(t),\ t \geq 1\}$ is given by

$$P'_n = \begin{bmatrix} 1 - \frac{r_{n-1}}{2} & \frac{r_{n-1}}{2} \\ \frac{r_{n-1}}{2} & 1 - \frac{r_{n-1}}{2} \end{bmatrix}. \tag{30}$$

(ii) The transition probability matrix $P''_n = [(P''_n)_{c_n c_{n-1}, c'_n c'_{n-1}}]_{c_n c_{n-1}, c'_n c'_{n-1} \in \{0,1\}^2}$ of the Markov chain $\{(c_n(t), c_{n-1}(t)),\ t \geq 1\}$ is given by

$$P''_n = \begin{bmatrix} \frac{1}{2} - \frac{r_{n-1}}{4} & \frac{r_{n-1}}{4} & \frac{1}{2} - \frac{r_{n-1}}{4} & \frac{r_{n-1}}{4} \\ \frac{r_{n-1}}{2} & \frac{1}{2} - \frac{r_{n-1}}{4} & 0 & \frac{1}{2} - \frac{r_{n-1}}{4} \\ \frac{1}{2} - \frac{r_{n-1}}{4} & 0 & \frac{1}{2} - \frac{r_{n-1}}{4} & \frac{r_{n-1}}{2} \\ \frac{r_{n-1}}{4} & \frac{1}{2} - \frac{r_{n-1}}{4} & \frac{r_{n-1}}{4} & \frac{1}{2} - \frac{r_{n-1}}{4} \end{bmatrix} \tag{31}$$

and the steady state probabilities $\boldsymbol{\pi}''_n = (\pi''_{n,00}, \pi''_{n,01}, \pi''_{n,10}, \pi''_{n,11})$ are given by

$$\boldsymbol{\pi}''_n = \left(\frac{2 + r_{n-1}}{2(4 + r_{n-1})}, \frac{1}{4 + r_{n-1}}, \frac{1}{4 + r_{n-1}}, \frac{2 + r_{n-1}}{2(4 + r_{n-1})} \right). \tag{32}$$

(iii) Starting with $r_1 = 1$, r_n could be recursively obtained for $n \geq 2$ as follows:

$$r_n = \frac{4}{4 + r_{n-1}}. \tag{33}$$

The approximation of r_n in (33) is shown in Table 4. From Table 4, we can see that for $n = 2$, the approximation in (33) is the same as the exact value of r_n in (29). This is no coincidence as we have mentioned earlier that the assumption in (A1) is true for $n = 2$. Furthermore, the approximation of r_n in (33) matches very well to the exact value of r_n in (29) for $3 \leq n \leq 10$. As $n \to \infty$, the coding rate $r_\infty = \lim_{n\to\infty} r_n$ can be obtained by solving $r_\infty = \frac{4}{4+r_\infty}$. The result is $r_\infty = 2\sqrt{2} - 2 \approx 0.8284$, which matches extremely well to the exact value of r_n in (33) for $6 \leq n \leq 10$. Finally, we mention that we have computed r_n by using the approximation in (33) and by running simulations for n up to 3000, and the results all agree with 0.8284 (up to the fourth digit after the decimal point) for $6 \leq n \leq 3000$.

4 Parallel bit-stuffing algorithm

The sequential bit-stuffing algorithm only takes one data bit stream. This poses a scalability problem when there are parallel data bit streams. In this section, we address this scalability problem by proposing a parallel bit-stuffing algorithm.

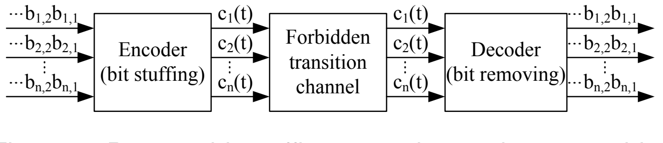

Fig. 2. Parallel bit-stuffing encoder and parallel bit-removing decoder for a forbidden transition channel with n parallel wires and n parallel data bit streams.

Parallel bit-stuffing algorithm:

The encoder of the parallel bit-stuffing algorithm takes the input of n data bit streams $\{b_{i,1}, b_{i,2}, \ldots\}$, $i = 1, 2, \ldots, n$, and converts them into codewords $(c_1(t), c_2(t), \ldots, c_n(t))$, $t = 1, 2, \ldots$, so that there are no opposite transitions on any two adjacent wires among all the n wires for all time t (see Figure 2). The parallel bit-stuffing algorithm is specified by using the following four rules:

(R1) (Initial condition) For $t = 0$, set

$$(c_1(0), c_2(0), \ldots, c_n(0)) = (0, 0, \ldots, 0).$$

(R2) (An odd-numbered wire) If i is an odd number, we simply set $c_i(t)$ to be the next data bit of the ith data bit stream.

(R3) (An internal even-numbered wire) If i is an even number and $i < n$, we need to consider the following three cases.

Case 1: (Bit-stuffing condition) If $\bar{c}_{i-1}(t-1) = c_{i-1}(t) = c_i(t-1)$, then $c_i(t)$ is a stuffed bit and we set $c_i(t) = c_i(t-1)$.

Case 2: (Bit-stuffing condition) If $\bar{c}_{i+1}(t-1) = c_{i+1}(t) = c_i(t-1)$, then $c_i(t)$ is a stuffed bit and we set $c_i(t) = c_i(t-1)$.

Case 3: Otherwise, we set $c_i(t)$ to be the next data bit of the ith data bit stream.

(R4) (The last boundary wire) If n is an even number and $i = n$, we need to consider the following two cases.

Case 1: (Bit-stuffing condition) If $\bar{c}_{n-1}(t-1) = c_{n-1}(t) = c_n(t-1)$, then $c_n(t)$ is a stuffed bit and we set $c_n(t) = c_n(t-1)$.

Case 2: Otherwise, we set $c_n(t)$ to be the next data bit of the nth data bit stream.

Note from the bit-stuffing conditions in Case 1 and Case 2 of (R3) and Case 1 of (R4) that there are no opposite transitions on any two adjacent wires. This is because we always stuff a bit (by setting $c_i(t) = c_i(t-1)$ in both cases) to prevent such a forbidden transition in (2).

As shown in the parallel bit-stuffing algorithm, bits transmitted on odd-numbered wires are always data bits. What the decoder needs to do is simply to remove the stuffed bits in the even-numbered wires. This can be easily done by observing that the coded bit $c_i(t)$ for an internal even-numbered wire i is a stuffed bit if and only if

$$c_i(t) = c_i(t-1) = c_{i-1}(t) = \bar{c}_{i-1}(t-1) \tag{34}$$

or

$$c_i(t) = c_i(t-1) = c_{i+1}(t) = \bar{c}_{i+1}(t-1). \tag{35}$$

Once we have the coded bits satisfy (34) and/or (35), we know that $c_i(t)$ is a stuffed bit and it should be removed. The case with the last boundary wire can be decoded similarly by using only (34) when n is an even number.

Clearly, the implementation complexity of the parallel bit-stuffing algorithm is only $O(n)$ for a bus with n parallel wires. This is the same as that for the implementation of the sequential bit-stuffing algorithm.

However, as both encoding/decoding for the parallel bit-stuffing algorithm can be implemented in parallel, the parallel bit-stuffing algorithm scales much better than the sequential bit-stuffing algorithm.

4.1 Coding rates

In this section, we compute the coding rate of the parallel bit-stuffing algorithm. Recall that the coding rate (throughput) of a wire is defined as the average number of data bits transmitted in one unit of time, and the coding rate of a bus with n parallel wires is defined as the average number of data bits transmitted per wire in one unit of time. As in the probabilistic analysis for the sequential bit-stuffing algorithm, we also assume that the data bits are *independent* Bernoulli random variables with equal probabilities of being 0 or 1.

As clearly stated in (R2) of the parallel bit-stuffing algorithm, every bit transmitted on an odd-numbered wire is a data bit. Thus, the coding rate for every odd-numbered wire is 100%, and we only need to compute the coding rate for even-numbered wires. For this, we need to consider two cases: (i) an internal even-numbered wire, i.e., $i = 2, 4, 6, \ldots, 2(\lceil n/2 \rceil - 1)$, and (ii) the last boundary wire when n is an even number. The coding rates for these cases are shown in the following theorem and its proof is deferred to Section 4.3.

Theorem 9 (i) *For an internal even-numbered wire, i.e.,* $i = 2, 4, 6, \ldots, 2(\lceil n/2 \rceil - 1)$, *the average number of data bits transmitted in one unit of time is* $5/8$.
(ii) *When* n *is an even number, the average number of bits transmitted on the last boundary wire is* $4/5$.

Let r_i, $i = 1, 2, \ldots, n$, be the coding rate of the i^{th} wire. If i is an odd number, one always transmits a data bit on that wire and thus $r_i = 1$. If i is an even number and $i < n$, we know from Theorem 9(i) that $r_i = 5/8$. Finally, if n is an even number and $i = n$, we have from Theorem 9(ii) that $r_i = 4/5$. Let $R_n^p = \frac{1}{n}\sum_{i=1}^{n} r_i$ be the coding rate of a bus with n wires under the parallel bit-stuffing algorithm. Then one can easily compute

$$R_n^p = \begin{cases} \frac{13}{16} + \frac{3}{16n}, & \text{if } n \text{ is odd,} \\ \frac{13}{16} + \frac{7}{40n}, & \text{if } n \text{ is even.} \end{cases} \tag{36}$$

In Table 5, we show the Shannon capacity C_n for a forbidden transition channel with n wires, the coding rate R_n of the sequential bit-stuffing algorithm, and the coding rate R_n^p of the parallel bit-stuffing algorithm for $1 \le n \le 10$. Observe that the coding rate of the parallel bit-stuffing algorithm is less than that of the sequential bit-stuffing algorithm for all $1 \le n \le 10$. However, the difference is very small. The difference is less than 1.6% for $1 \le n \le 10$. Even when $n \to \infty$, the difference between $R_\infty \approx 0.8284$ and $R_\infty^p = 0.8125$ is still less than 2%. The advantage of the parallel bit-stuffing algorithm is that it allows encoding/decoding to be performed in parallel. As such, it scales much better than the sequential bit-stuffing algorithm.

TABLE 5
The Shannon capacity C_n, the coding rate R_n of the sequential bit-stuffing algorithm, and the coding rate R_n^p of the parallel bit-stuffing algorithm for $1 \le n \le 10$.

n	1	2	3	4	5
C_n	1	0.9163	0.8941	0.8826	0.8757
R_n	1	0.9	0.8779	0.8653	0.8580
R_n^p	1	0.9	0.875	0.8562	0.85

n	6	7	8	9	10
C_n	0.8712	0.8679	0.8654	0.8635	0.8620
R_n	0.8531	0.8495	0.8469	0.8448	0.8432
R_n^p	0.8417	0.8393	0.8344	0.8333	0.83

In Figure 3, we also compare the theoretical coding rates of various schemes, including the ground shielding scheme in [2] and the Fibonacci representation scheme in [3]. Note that in the ground shielding scheme, one always transmits 0 on every even-numbered wire. Since our parallel bit-stuffing algorithm achieves the coding rate of $5/8$ on every even-numbered wire, our scheme is significantly better than the ground shielding scheme (for the average-case). On the other hand, our parallel bit-stuffing algorithm also performs much better than the Fibonacci representation scheme in [3]. Note that the best hardware implementation complexity for implementing the Fibonacci representation scheme for a bus of n wires is $O(n^2)$ (see e.g., [6], [26]). This is also much more complex than the $O(n)$ complexity in our scheme.

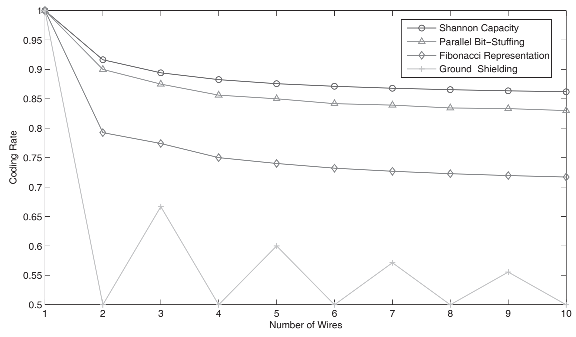

Fig. 3. The comparison of coding rates of various schemes.

Even though the coding rate for an even-numbered wire of our parallel bit-stuffing algorithm is much better than that of the ground shielding scheme, its worse case is as bad as that of the ground shielding scheme. For instance, consider the scenario that the sequence of data bits on the first (resp., third) wire is $\{1, 0, 1, 0, \ldots\}$ (resp., $\{0, 1, 0, 1, \ldots\}$). Suppose that the first data bit on the second wire is 0. Then all of the subsequent bits on the second wire are needed to be stuffed with 0. This is the same as transmitting a stream of 0's on the second wire in the ground shielding scheme, and that results in a coding rate near 0 on the second wire. This worst-case scenario shows how serious the problem of uneven coding rates could be. We will provide a simple solution in the next section.

4.2 Rate balancing

The problem of our parallel bit-stuffing algorithm is that the coding rate of an even-numbered wire is lower than that of an odd-numbered wire. When transmitting parallel finite sequences of data bits over the forbidden transition channel, one has to add padded bits on a wire once it completes the transmission of its data bits before others. As commented in the previous section, the worst case for this could be very serious as it is possible to have an even-numbered wire that has a coding rate near 0.

To further understand the problem of uneven coding rates, we perform a series of experiments by simulating the transmission of a packet of 1500 bytes (a typical Ethernet packet size) over a bus of 32 wires under the parallel bit-stuffing algorithm. For this experiment, we partition the 1500×8 data bits evenly over the 32 wires and each wire thus needs to transmit 375 data bits. Once a wire completes the transmission of its data bits, it starts to add padded bits until all the wires complete the transmission of their 375 data bits. Define the transmission time of a packet as the number of transmitted bits (including both data bits and padded bits) on a wire when all the wires complete the transmission of their 375 data bits. In Figure 4, we show the simulation result for the cumulative distribution function (CDF) for the transmission time of a packet. From Figure 4, we see that 99% of our experiments have the transmission time less than 541. This leads to an empirical coding rate of $375/541 \approx 0.6931$ for transmitting a packet of 1500 bytes.

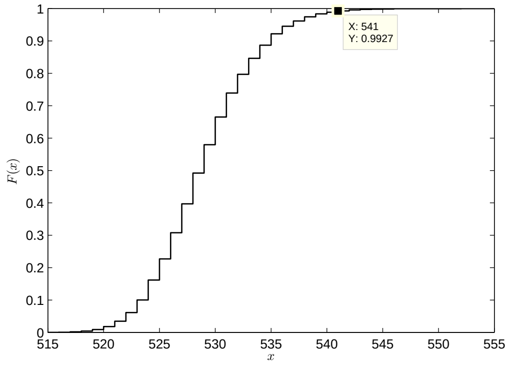

Fig. 4. The CDF $F(x)$ of the transmission time x of a packet of 1500 bytes over a bus of 32 wires under the parallel bit-buffering algorithm.

Our idea for solving the problem of uneven coding rates is *rate-balancing*. As shown in Figure 5, we add a 2×2 crossbar switch for every two wires before the encoder, and append a 2×2 crossbar switch for every two wires after the decoder. The connection patterns of all these 2×2 crossbar switches are synchronized and they alternate periodically (with a period of 2) between the "bar" state and the "cross" state. In the bar state, the first (resp., second) input wire of a crossbar switch is connected to the first (resp., second) output wire of that switch. On the other hand, in the cross state, the first (resp., second) input wire of a 2×2 crossbar switch is connected to the second (resp., first) output wire of that switch. By so doing, every input/output stream is alternatively connected to an even-numbered wire and an odd-numbered wire as shown in Figure 5. As such, the coding rate of an internal wire is $(1+5/8)/2 = 0.8125$. Moreover, as the connection patterns are synchronized, all the data bits can still be encoded and decoded correctly.

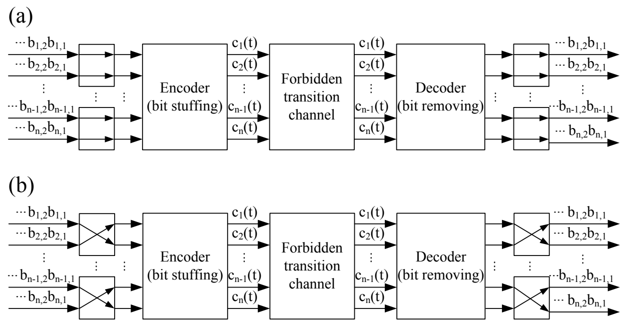

Fig. 5. Parallel bit-stuffing encoder and parallel bit-removing decoder with rate-balancing for a forbidden transition channel with n parallel wires, where n is an even number in this case. (a) At time $t = 1, 3, \ldots$, all the 2×2 crossbar switches are set to the bar state. (b) At time $t = 2, 4, \ldots$, all the 2×2 crossbar switches are set to the cross state.

In Figure 6, we report the simulation result for the parallel bit-stuffing algorithm with rate-balancing under the same setting in Figure 4. From Figure 6, we see that 99% of our experiments have the transmission time less than 486. This leads to an empirical coding rate of $375/486 \approx 0.7716$ for transmitting a packet of 1500 bytes. Clearly, the improvement is quite significant when comparing with the original parallel bit-stuffing algorithm (without rate-balancing). We note that the empirical coding rate 0.7716 is still lower than the theoretical coding rate 0.8125. This is due to the fact the theoretical coding rate is derived in the asymptotic regime where the packet length is assumed to be infinite.

We note that there are many schemes for adding padded bits. Clearly, adding random bits is not good as it might cause transitions on the same wire. Here in our experiments, whenever the next data bit is a padded bit, we simply set the coded bit to be the same as the previous coded bit. By so doing, we reduce the number of transitions on the same wire.

Also, there are other rate-balancing schemes. For instance, one could use an $n \times n$ crossbar switch in front of the encoder and run a set of n periodic connections as in the load-balanced switches (see e.g., [27]–[30]). This certainly will mitigate the boundary effect and achieve a better result for rate-balancing. However, this is at the cost of a much more complicated switch design than

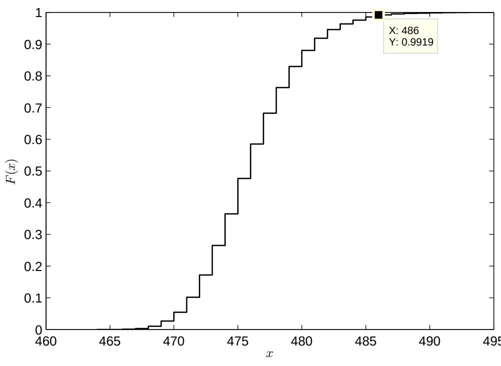

Fig. 6. The CDF $F(x)$ of the transmission time x of a packet of 1500 bytes over a bus of 32 wires under the parallel bit-buffering algorithm with *rate-balancing*.

using simple 2×2 crossbar switches. Another possible approach is to alternate bit-stuffing between even-numbered wires and odd-numbered wires. But this also complicates the hardware design.

4.3 Proof of Theorem 9

(i) First, we show that for an internal even-numbered wire, i.e., $i = 2, 4, 6, \ldots, 2(\lceil n/2 \rceil - 1)$, the average number of data bits transmitted in one unit of time is $5/8$.

Let $\mathbf{c}(t) = (c_{i-1}(t), c_i(t), c_{i+1}(t))$. According to the parallel bit-stuffing algorithm, the code vector $\mathbf{c}(t)$ is a function of $\mathbf{c}(t-1)$ and the input data bit streams. As the input data bit streams consist of i.i.d. Bernoulli random variables, the code vector at time t is independent of all the code vectors before time $t-1$ once the code vector at time $t-1$ is given. This shows that $\mathbf{c}(t)$ is a time-homogeneous Markov chain. Now we calculate the transition probabilities of the Markov chain $\mathbf{c}(t)$. Let $\mathbf{c} = (c_1, c_2, c_3)$ and $\mathbf{c}' = (c_1', c_2', c_3')$. Denote the transition probability from the state $\mathbf{c}$ to $\mathbf{c}'$ by $P_{\mathbf{c},\mathbf{c}'}$, i.e.,

$$P_{\mathbf{c},\mathbf{c}'} = P(\mathbf{c}(t) = \mathbf{c}' | \mathbf{c}(t-1) = \mathbf{c}).$$

Clearly, we have $P_{\mathbf{c},\mathbf{c}'} = 0$ if there is a forbidden transition as described in (1). Specifically, $P_{\mathbf{c},\mathbf{c}'} = 0$ whenever

$$\bar{c}_1 = c_1' = c_2 = \bar{c}_2' \tag{37}$$

or

$$\bar{c}_3 = c_3' = c_2 = \bar{c}_2'. \tag{38}$$

On the other hand, we have from the chain rule that

$$\begin{aligned} P_{\mathbf{c},\mathbf{c}'} &= P(\mathbf{c}(t) = \mathbf{c}' | \mathbf{c}(t-1) = \mathbf{c}) \\ &= P(c_{i-1}(t) = c_1' | \mathbf{c}(t-1) = \mathbf{c}) \\ &\times \; P(c_{i+1}(t) = c_3' | \mathbf{c}(t-1) = \mathbf{c}, c_{i-1}(t) = c_1') \\ &\times \; P(c_i(t) = c_2' | \mathbf{c}(t-1) = \mathbf{c}, c_{i-1}(t) = c_1', \\ &\qquad c_{i+1}(t) = c_3'). \end{aligned} \tag{39}$$

Recall that both $c_{i-1}(t)$ and $c_{i+1}(t)$ are data bits. Since we assume that the data bits are i.i.d. Bernoulli random variables with equal probabilities of being 0 or 1, we then have

$$P(c_{i-1}(t) = c_1' | \mathbf{c}(t-1) = \mathbf{c}) = P(c_{i-1}(t) = c_1') = \frac{1}{2}, \tag{40}$$

and

$$\begin{aligned} &P(c_{i+1}(t) = c_3' | \mathbf{c}(t-1) = \mathbf{c}, c_{i-1}(t) = c_1') \\ &= P(c_{i+1}(t) = c_3') = \frac{1}{2}. \end{aligned} \tag{41}$$

To compute

$$P(c_i(t) = c_2' | \mathbf{c}(t-1) = \mathbf{c}, c_{i-1}(t) = c_1', c_{i+1}(t) = c_3'),$$

we note that its value is also $1/2$ if $c_i(t)$ is a data bit, and its value is 1 if $c_i(t)$ is a stuffed bit. By using (R3), we know that c_2' in this conditional probability is a stuffed bit whenever

$$c_2' = c_2 = c_1' = \bar{c}_1 \tag{42}$$

or

$$c_2' = c_2 = c_3' = \bar{c}_3. \tag{43}$$

Otherwise, it is a data bit (excluding the cases for the forbidden transitions in (37) and (38)). We enumerate all the cases in Table 6, where we define a function $g(\mathbf{c}, \mathbf{c}')$, in which "1" indicates a data bit, "0" indicates a stuffed bit, and "F" indicates a forbidden transition ($P_{\mathbf{c},\mathbf{c}'} = 0$). Thus, we have from (39)–(41) that

$$P_{\mathbf{c},\mathbf{c}'} = \left\{ \begin{array}{ll} 0, & \text{if } g(\mathbf{c}, \mathbf{c}') = F, \\ (\frac{1}{2})^2 (\frac{1}{2})^{g(\mathbf{c},\mathbf{c}')}, & \text{if } g(\mathbf{c}, \mathbf{c}') = 0 \text{ or } 1. \end{array} \right. \tag{44}$$

Order the eight states by (0,0,0), (0,0,1), (0,1,0), (0,1,1), (1,0,0), (1,0,1), (1,1,0), (1,1,1). Using (44) and Table 6, we then have the the following transition probability matrix:

$$P = \begin{bmatrix} \frac{1}{8} & \frac{1}{8} & \frac{1}{8} & \frac{1}{8} & \frac{1}{8} & \frac{1}{8} & \frac{1}{8} & \frac{1}{8} \\ \frac{1}{4} & \frac{1}{8} & 0 & \frac{1}{8} & \frac{1}{4} & \frac{1}{8} & 0 & \frac{1}{8} \\ \frac{1}{8} & 0 & \frac{1}{8} & \frac{1}{4} & 0 & 0 & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{8} & \frac{1}{8} & \frac{1}{8} & \frac{1}{8} & 0 & 0 & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & 0 & 0 & \frac{1}{8} & \frac{1}{8} & \frac{1}{8} & \frac{1}{8} \\ \frac{1}{4} & \frac{1}{4} & 0 & 0 & \frac{1}{4} & \frac{1}{8} & 0 & \frac{1}{8} \\ \frac{1}{8} & 0 & \frac{1}{8} & \frac{1}{4} & \frac{1}{8} & 0 & \frac{1}{8} & \frac{1}{4} \\ \frac{1}{8} & \frac{1}{8} & \frac{1}{8} & \frac{1}{8} & \frac{1}{8} & \frac{1}{8} & \frac{1}{8} & \frac{1}{8} \end{bmatrix}.$$

Let

$$\pi = (\pi_{000}, \pi_{001}, \pi_{010}, \pi_{011}, \pi_{100}, \pi_{101}, \pi_{110}, \pi_{111})$$

TABLE 6
The values of the function $g(\mathbf{c}, \mathbf{c}')$.

$\mathbf{c} \setminus \mathbf{c}'$	000	001	010	011	100	101	110	111
000	1	1	1	1	1	1	1	1
001	0	1	F	1	0	1	F	1
010	1	F	1	0	F	F	0	0
011	1	1	1	1	F	F	0	0
100	0	0	F	F	1	1	1	1
101	0	0	F	F	0	1	F	1
110	1	F	1	0	1	F	1	0
111	1	1	1	1	1	1	1	1

be the steady state probability vector. Then one can solve π by

$$\sum_{\mathbf{c}} \pi_{\mathbf{c}} = 1 \text{ and } \pi = \pi P.$$

By symmetry, we know that $\pi_{c_1c_2c_3}$ and $\pi_{\bar{c}_1\bar{c}_2\bar{c}_3}$ are identical. Also, as the bits transmitted on the $(i-1)^{\text{th}}$ wire and the $(i+1)^{\text{th}}$ wire are data bits, we have for all c_1 and c_3 that

$$\pi_{c_1 0 c_3} + \pi_{c_1 1 c_3} = \frac{1}{4}.$$

From these, we can then solve

$$\pi = \left(\frac{1}{6}, \frac{1}{8}, \frac{1}{12}, \frac{1}{8}, \frac{1}{8}, \frac{1}{12}, \frac{1}{8}, \frac{1}{6}\right).$$

With the steady state probability vector, we can then compute the average number of data bits transmitted on the i^{th} wire by

$$\sum_{\mathbf{c}} \pi_{\mathbf{c}} \sum_{\mathbf{c}'} P_{\mathbf{c},\mathbf{c}'} g(\mathbf{c}, \mathbf{c}') = \frac{5}{8},$$

where $i = 2, 4, 6, \ldots, 2(\lceil n/2 \rceil - 1)$.

(ii) When n is an even number, we show in the following that the average number of bits transmitted on the last boundary wire is $4/5$.

For this, we only need to consider the states of the last two wires, i.e., $(c_{n-1}(t), c_n(t))$. Order the four states by (0,0), (0,1), (1,0) and (1,1). Analog to the argument used in Theorem 9(i) for an internal even-numbered wire, it is easy to obtain the following transition probability matrix

$$P = \begin{bmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{bmatrix}.$$

One can also observe the following from the transition probability matrix: (i) a data bit is transmitted if the transition probability is $1/4$, (ii) a stuffed bit is transmitted if the transition probability is $1/2$, and (iii) a forbidden transition if the transition probability is 0.

Then one can solve the steady state probability vector

$$\pi = (\pi_{00}, \pi_{01}, \pi_{10}, \pi_{11}) = \left(\frac{3}{10}, \frac{1}{5}, \frac{1}{5}, \frac{3}{10}\right),$$

and use that to show the average number of data bits transmitted on the n^{th} wire is $4/5$.

5 Conclusion

Motivated by the design of high-speed switching fabrics, in this paper we have proposed a sequential bit-stuffing algorithm and a parallel bit-stuffing algorithm for generating forbidden transition codes to mitigate the crosstalk effect between adjacent wires in long on-chip buses. We also have developed the associated analysis for these two algorithms. We have shown by both theoretic analysis and simulations that the coding rates of these two bit-stuffing algorithms are quite close to the Shannon capacity, and hence are much better than those of the existing forbidden transition codes in the literature, including the Fibonacci representation in [3], [4], and [6].

In this paper, we assume i.i.d. Bernouilli random variables with equal probabilities of being 0 or 1. This is often a valid assumption on scrambled data, but on internal bus systems this can be quite different. One way to fix this is to apply a distribution transformer [13] that converts the original data bit-streams into an i.i.d. Bernouilli random variables with equal probabilities of being 0 or 1. Another issue is error propagation in the bit-stuffing algorithm. To address such a problem, one has to put a limit on the maximum packet length, e.g., 1500 bytes in our numerical examples. When the maximum packet length is exceeded during the decoding process, we know there is an error and that packet is then discarded. Discarded packets have to be retransmitted by an upper layer protocol.

There are several research directions that are worth further investigation.

(i) Bounding the capacity of the sequential bit-stuffing algorithm: In this paper, we provided an approximate probabilistic analysis for the capacity of the sequential bit-stuffing algorithm. There are recent efforts in obtaining bounds by linear programming and convex programming for the bit-stuffing algorithms on various applications (see e.g., [21], [22]). It would be of interest to apply their approaches to obtain tight bounds for our setting.

(ii) Bounding the number of padded bits of the parallel bit-stuffing algorithm: Unlike the fixed-length memoryless encoder by using the the ground shielding scheme in [2] and Fibonacci representation scheme in [3], [4], and [6], our parallel bit-stuffing algorithm is a variable-length encoder with memory. As such, padded bits have to be added in our scheme. For determining the size of both input/output buffers before the parallel encoder, one possible future research topic is to derive tight bounds for the number of padded bits.

Acknowledgment

This work was supported in part by Ministry of Science and Technology (MOST), Taiwan, under Grant Numbers 102-2221-E-007-006-MY3, 103-2622-E-009-012 and 103-2218-E-007-022.

References

[1] P. P. Sotiriadis, "Interconnect modeling and optimization in deep submicron technologies," Ph.D. Dissertation, Massachusetts Institute of Technology, Cambridge, MA, USA, 2002.

[2] J. D. Z. Ma and L. He, "Formulae and applications of interconnect estimation considering shield insertion and net ordering," in *Proceedings IEEE/ACM International Conference on Computer-Aided Design (ICCAD'01)*, San Jose, CA, USA, November 4–8, 2001, pp. 327–332.

[3] B. Victor and K. Keutzer, "Bus encoding to prevent crosstalk delay," in *Proceedings IEEE/ACM International Conference on Computer-Aided Design (ICCAD'01)*, San Jose, CA, USA, November 4–8, 2001, pp. 57–63.

[4] M. Mutyam, “Preventing crosstalk delay using Fibonacci representation,” in *Proceedings 17th International Conference on VLSI Design (VLSID’04)*, Mumbai, India, January 5–9, 2004, pp. 685–688.

[5] B. E. Moision, A. Orlitsky, and P. H. Siegel, “On codes that avoid specified differences,” *IEEE Transactions on Information Theory*, vol. 47, pp. 433–442, January 2001.

[6] C. Duan, C. Zhu, and S. P. Khatri, “Forbidden transition free crosstalk avoidance CODEC design,” in *Proceedings 45th Annual Design Automation Conference (DAC’08)*, Anaheim, CA, USA, June 8–13, 2008, pp. 986–991.

[7] X. Wu, Z. Yan, and Y. Xie, “Two-dimensional crosstalk avoidance codes,” in *Proceedings IEEE Workshop on Signal Processing Systems (SiPS’08)*, Washington DC, USA, October 8–10, 2008, pp. 106–111.

[8] S. R. Sridhara and N. R. Shanbhag, “Coding for system-on-chip networks: a unified framework,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 13, pp. 655–667, June 2005.

[9] W.-W. Hsieh, P.-Y. Chen, C.-Y. Wang, and T.-T. Hwang, “A bus-encoding scheme for crosstalk elimination in high-performance processor design,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, pp. 2222–2227, December 2007.

[10] C.-S. Chang, J. Cheng, T.-K. Huang and D.-S. Lee, ”Explicit constructions of memoryless crosstalk avoidance codes via C-transform,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 9, pp. 2030–2033, September 2014.

[11] L. L. Peterson and B. S. Davie, *Computer Networks: A Systems Approach*, 4th edition, San Francisco, CA: Morgan Kaufmann Publishers, 2007.

[12] R. M. Roth, P. H. Siegel, and J. K. Wolf, “Efficient coding schemes for the hard-suqare constraint,” *IEEE Transactions on Information Theory*, vol. 47, pp. 1166–1176, Mar. 2001.

[13] S. Halevy, J. Chen, R. M. Roth, P. H. Siegel, and J. K. Wolf, “Improved bit-stuffing bounds on two-dimensional constraints,” *IEEE Transactions on Information Theory*, vol. 50, pp. 824–838, May 2004.

[14] S. Aviran, P. H. Siegel, and J. K. Wolf, “An improvement to the bit stuffing algorithm,” *IEEE Transactions on Information Theory*, vol. 51, pp. 2885–2891, August 2005.

[15] C. E. Shannon, “A mathematical theory of communication,” *Bell System Technical Journal*, vol. 27, pp. 379–423 (Part I), 623–656 (Part II), July, October 1948.

[16] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, New York, NY: John Wiley & Sons, 1991.

[17] E. K. Orcutt and W. M. Marcellin, “Redundant multitrack (d, k) codes,” *IEEE Transactions on Information Theory*, vol. 39, pp. 1744–1750, 1993.

[18] W. Weeks and R.E. Blahut, “The capacity and coding gain of certain checkerboard codes,” *IEEE Transactions on Information Theory*, vol. 44, pp. 1193–1203, 1998.

[19] N. J. Calkin and H. S. Wilf, “The number of independent sets in a grid graph,” *SIAM J. Discr. Math*, vol. 11, pp. 54–60, 1998.

[20] J. Justesen and S. Forchhammer, *Two-dimensional Information Theory and Coding*, Cambridge: Cambridge University Press, 2010.

[21] I. Tal and R. M. Roth, “Bounds on the rate of 2-D bit-stuffing encoders,” *IEEE Transactions Information Theory*, vol. 56, pp. 2561–2567, 2010.

[22] I. Tal and R. M. Roth, “Convex Programming Upper Bounds on the Capacity of 2-D Constraints,” *IEEE Transactions Information Theory*, vol. 57, pp. 381–391, 2011.

[23] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge, UK: Cambridge University Press, 1985.

[24] S. I. Resnick, *Adventures in Stochastic Processes*, Boston, MA: Birkhaüser, 1992.

[25] D. K. Pickard, “Unilateral Markov fields,” *Advances in Applied Pprobability*, vol. 12, pp. 655–671, 1980.

[26] X. Wu and Z. Yan, “Efficient CODEC designs for crosstalk avoidance codes based on numeral systems,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, pp. 548–558, 2011.

[27] C.-S. Chang, D.-S. Lee and Y.-S. Jou, “Load balanced Birkhoff-von Neumann switches, part I: one-stage buffering,” *Computer Communications*, vol. 25, pp. 611–622, April 2002.

[28] I. Keslassy, S.-T. Chuang, K. Yu, D. Miller, M. Horowitz, O. Solgaard, and N. McKeown, “Scaling internet routers using optics,” in *Proceedings ACM SIGCOMM 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM’03)*, Karlsruhe, Germany, August 25–29, 2003, pp. 189–200.

[29] Y. Shen, S. Jiang, S. S. Panwar, and H. J. Chao, “Byte-focal: a practical load balanced switch,” in *Proceedings 2005 IEEE Workshop on High Performance Switching and Routing (HPSR’05)*, Hong Kong, P. R. China, May 12–14, 2005, pp. 6–12.

[30] J. J. Jaramillo, F. Milan, and R. Srikant, “Padded frames: a novel algorithm for stable scheduling in load-balanced switches,” *IEEE/ACM Transactions on Networking*, vol. 16, pp. 1212–1225, October 2008.

Cheng-Shang Chang (S’85-M’86-M’89-SM’93-F’04) received the B.S. degree from National Taiwan University, Taipei, Taiwan, in 1983, and the M.S. and Ph.D. degrees from Columbia University, New York, NY, USA, in 1986 and 1989, respectively, all in Electrical Engineering. From 1989 to 1993, he was employed as a Research Staff Member at the IBM Thomas J. Watson Research Center, Yorktown Heights, N.Y. Since 1993, he has been with the Department of Electrical Engineering at National Tsing Hua University, Taiwan, R.O.C., where he is a Tsing Hua Chair Professor. His current research interests are concerned with network science, high speed switching, communication network theory, and mathematical modeling of the Internet. Dr. Chang received an IBM Outstanding Innovation Award in 1992, an IBM Faculty Partnership Award in 2001, and Outstanding Research Awards from the National Science Council, Taiwan, in 1998, 2000 and 2002, respectively. He also received Outstanding Teaching Awards from both the college of EECS and the university itself in 2003. He was appointed as the first Y. Z. Hsu Scientific Chair Professor in 2002 and elected to an IEEE Fellow in 2004. Dr. Chang received the Academic Award from the Ministry of Education and the Merit NSC Research Fellow Award from the National Science Council in 2011. He is the author of the book “Performance Guarantees in Communication Networks” and the coauthor of the book ”Principles, Architectures and Mathematical Theory of High Performance Packet Switches.” He served as an editor for Operations Research from 1992 to 1999 and an editor for IEEE/ACM Transactions on Networking from 2007 to 2009. He is currently serving as an editor-at-large for IEEE/ACM Transactions on Networking and an editor for IEEE Transactions on Network Science and Engineering. Dr. Chang is a member of IFIP Working Group 7.3.

Jay Cheng received the B.S. and M.S. degrees from National Tsing Hua University, Hsinchu, Taiwan, R.O.C., in 1993 and 1995, respectively, and the Ph.D. degree from Cornell University, Ithaca, NY, USA, in 2003, all in Electrical Engineering. In August 2003, he joined the Department of Electrical Engineering at National Tsing Hua University, Hsinchu, Taiwan, R.O.C., where he is currently a Professor. Since October 2004, Dr. Cheng has also been affiliated with the Institute of Communications Engineering at National Tsing Hua University, Hsinchu, Taiwan, R.O.C. His current research interests include mathematical logic, foundations of mathematics, game theory, network science, optical queueing theory, high-speed switching theory, and information theory.

Tien-Ke Huang (S'06-M'10) received the B.S. and Ph.D. degrees in 2002 and 2010, respectively, both from National Tsing Hua University, Hsinchu, Taiwan, R.O.C. From 2011 to 2012, he was a Postdoctoral Researcher with National Tsing Hua University. In August 2012, he joined Metanoia Communications Inc., as a Senior Engineer for DSL systems development. Since August 2014, he has been with Realtek Semiconductor Corp. His research interests are in communication theory and information theory.

Xuan-Chao Huang received the B.S., M.S., and Ph.D. degrees from National Tsing Hua University, Hsinchu, Taiwan, R.O.C., in 2005, 2008, and 2011, respectively. From 2011 to 2013, he was a postdoctoral research fellow in National Tsing Hua University. From 2013 to present, he serves as a software engineer in Gorilla Inc.. His research interests are in optical queueing theory, high speed switching, and network clustering algorithm.

Duan-Shin Lee (S'89-M'90-SM'98) received the B.S. degree from National Tsing Hua University, Taiwan, in 1983, and the MS and Ph.D. degrees from Columbia University, New York, in 1987 and 1990, all in electrical engineering. He worked as a research staff member at the C&C Research Laboratory of NEC USA, Inc. in Princeton, New Jersey from 1990 to 1998. He joined the Department of Computer Science of National Tsing Hua University in Hsinchu, Taiwan, in 1998. Since August 2003, he has been a professor. He received a best paper award from the Y.Z. Hsu Foundation in 2006. His current research interests are high-speed switch and router design, social networks, network science and data engineering. He is a senior IEEE member.

Chao-Yi Chen received the B.S.degree in Electrical Engineering from National Tsing Hua University, Hsinchu, Taiwan, in 2008, and he received the M.S. degree in Communications Engineering from National Tsing Hua University, Hsinchu, Taiwan, in 2010. He have been with USI, Nantou, Taiwan, as a Signal Integrity Engineer since 2010. He received the 10th Golden Silicon Awards from MXIC, Taiwan, in 2010, and he also received the EMC Design Game Awards from BSMI, Taiwan, in 2011 and 2014, respectively.

Appendix A
Proof of Theorem 1

Let $X_{n,\mathbf{c}}(t)$, $\mathbf{c} = (c_n, c_{n-1}, \ldots, c_1)$ and $t = 1, 2, \ldots$, be the number of sequences $(\mathbf{c}(1), \mathbf{c}(2), \ldots, \mathbf{c}(t))$ that satisfy the constraint in (1) up to time t and $\mathbf{c}(t) = \mathbf{c}$, i.e., the sequence $(\mathbf{c}(1), \mathbf{c}(2), \ldots, \mathbf{c}(t))$ ends in the state $\mathbf{c}$ at time t. Clearly, we have $X_{n,\mathbf{c}}(1) = 1$ for all $\mathbf{c} \in \{0,1\}^n$ and $X_n(t) = \sum_{\mathbf{c}\in\{0,1\}^n} X_{n,\mathbf{c}}(t)$ for all $t = 1, 2, \ldots$.

Let $\mathbf{0}_n = (0, 0, \ldots, 0)$ be the row vector of size 2^n whose entries are all equal to 0, and let $\mathbf{1}_n = (1, 1, \ldots, 1)$ be the row vector of size 2^n whose entries are all equal to 1. Let $\mathbf{X}_n(t) = (X_{n,\mathbf{0}_n}(t), \ldots, X_{n,\mathbf{1}_n}(t))$ be the row vector of size 2^n whose i^{th} entry, $i = 1, 2, \ldots, 2^n$, is given by $X_{n,\mathbf{c}}(t)$ in which $\mathbf{c} = (c_n, c_{n-1}, \ldots, c_1)$ satisfies $\sum_{j=1}^{n} c_j 2^{j-1} = i - 1$. Then it is easy to see that $\mathbf{X}_n(t) = \mathbf{X}_n(t-1)A_n$ for $t = 2, 3, \ldots$, and hence we have

$$\mathbf{X}_n(t) = \mathbf{X}_n(1)A_n^{t-1}, \text{ for } t = 1, 2, \ldots. \tag{45}$$

As such, we have from $X_n(t) = \sum_{\mathbf{c}\in\{0,1\}^n} X_{n,\mathbf{c}}(t) = \mathbf{X}_n(t)\mathbf{1}_n^T$, (45), and $\mathbf{X}_n(1) = \mathbf{1}_n$ that

$$\begin{aligned} X_n(t) &= \mathbf{X}_n(t)\mathbf{1}_n^T = (\mathbf{X}_n(1)A_n^{t-1})\mathbf{1}_n^T = \mathbf{1}_n A_n^{t-1}\mathbf{1}_n^T \\ &= \sum_{\mathbf{c},\mathbf{c}'\in\{0,1\}^n} (A_n^{t-1})_{\mathbf{c},\mathbf{c}'}. \end{aligned} \tag{46}$$

Recall that for an $m \times m$ matrix M, the maximum-row-sum matrix norm $|||M|||_\infty$ of M is defined as [23, Definition 5.6.5]

$$|||M|||_\infty = \max_{1\le i\le m} \sum_{j=1}^{m} |M_{i,j}|, \tag{47}$$

and its spectral radius $\rho(M) = \max\{|\lambda| : \lambda \text{ is an eigenvalue of } M\}$ [23, Definition 5.6.8] is given by [23, Corollary 5.6.14]

$$\rho(M) = \lim_{k\to\infty} |||M^k|||_\infty^{1/k}. \tag{48}$$

Furthermore, if M is nonnegative, i.e., $M_{i,j} \ge 0$ for all $1 \le i, j \le m$, then $\rho(M)$ is an eigenvalue of M [23, Theorem 8.3.1] and hence we have

$$\rho(M) = \max\{\lambda : \lambda \text{ is an eigenvalue of } M\}. \tag{49}$$

As A_n is a nonnegative matrix, we see from (46) and (47) that

$$|||A_n^{t-1}|||_\infty \le X_n(t) \le 2^n \cdot |||A_n^{t-1}|||_\infty. \tag{50}$$

It then follows from (3), (50), (48), and (49) that

$$\begin{aligned} C_n &= \frac{1}{n}\lim_{t\to\infty}\frac{\log_2 X_n(t)}{t} = \frac{1}{n}\lim_{t\to\infty}\frac{\log_2 |||A_n^{t-1}|||_\infty}{t} \\ &= \frac{1}{n}\log_2\left(\lim_{t\to\infty}|||A_n^{t-1}|||_\infty^{1/t}\right) = \frac{1}{n}\log_2\rho(A_n) \\ &= \frac{1}{n}\log_2\lambda_{n,\max}, \end{aligned} \tag{51}$$

where $\lambda_{n,\max} = \max_{1\le i\le 2^n}\lambda_{n,i}$ is the maximum eigenvalue of the adjacency matrix A_n.

Appendix B
Proof of Lemma 6

(i) Note that from (22) and (17), we have

$$\boldsymbol{\pi}_n^{(0)} = \boldsymbol{\pi}_n^{(0)}E_{n-1} + \boldsymbol{\pi}_n^{(1)}G_{n-1}, \tag{52}$$

$$\boldsymbol{\pi}_n^{(1)} = \boldsymbol{\pi}_n^{(0)}F_{n-1} + \boldsymbol{\pi}_n^{(1)}H_{n-1}. \tag{53}$$

By adding (52) and (53), and using (18)–(21), and (17), we can see that

$$\begin{aligned} \boldsymbol{\pi}_n^{(0)} + \boldsymbol{\pi}_n^{(1)} &= \boldsymbol{\pi}_n^{(0)}(E_{n-1}+F_{n-1}) + \boldsymbol{\pi}_n^{(1)}(G_{n-1}+H_{n-1}) \\ &= (\boldsymbol{\pi}_n^{(0)} + \boldsymbol{\pi}_n^{(1)})\begin{bmatrix} E_{n-2} & F_{n-2} \\ G_{n-2} & H_{n-2} \end{bmatrix} \\ &= (\boldsymbol{\pi}_n^{(0)} + \boldsymbol{\pi}_n^{(1)})P_{n-1}. \end{aligned} \tag{54}$$

Also, it is clear from (23) that

$$\begin{aligned} (\boldsymbol{\pi}_n^{(0)} + \boldsymbol{\pi}_n^{(1)})\mathbf{1}_{n-1}^T &= \sum_{\mathbf{c}^{(n-1)}\in\{0,1\}^{n-1}} (\pi_{n,0\mathbf{c}(n-1)} + \pi_{n,1\mathbf{c}(n-1)}) \\ &= \sum_{\mathbf{c}\in\{0,1\}^n} \pi_{n,\mathbf{c}} = 1. \end{aligned} \tag{55}$$

It follows from (54), (55), and the uniqueness of $\boldsymbol{\pi}_{n-1}$ that

$$\boldsymbol{\pi}_{n-1} = \boldsymbol{\pi}_n^{(0)} + \boldsymbol{\pi}_n^{(1)}.$$

(ii) From (52) and (24), we can see that

$$\begin{aligned} \boldsymbol{\pi}_n^{(0)}(I_{n-1} - E_{n-1} + G_{n-1}) &= \boldsymbol{\pi}_n^{(1)}G_{n-1} + \boldsymbol{\pi}_n^{(0)}G_{n-1} \\ &= \boldsymbol{\pi}_{n-1}G_{n-1}. \end{aligned} \tag{56}$$

From (18) and (20), we have

$$I_{n-1} - E_{n-1} + G_{n-1} = \begin{bmatrix} I_{n-2} & -\frac{1}{2}F_{n-2} \\ -\frac{1}{2}G_{n-2} & I_{n-2} \end{bmatrix}. \tag{57}$$

Note that from (17), we have

$$P_{n-1} = \begin{bmatrix} E_{n-2} & F_{n-2} \\ G_{n-2} & H_{n-2} \end{bmatrix}.$$

It follows that each row sum of F_{n-2} and each row sum of G_{n-2} are less than 1 since each row sum of P_{n-1} is equal to 1 and each row of E_{n-2} and each row of H_{n-2} contain at least one positive entry. Therefore, we see from (57) and the well-known Gerschgorin's Disk Theorem [23, Theorem 6.1.1] that each eigenvalue of $I_{n-1} - E_{n-1} + G_{n-1}$ is contained in the following disk:

$$\left\{ z \in \mathbf{C} : |z - 1| < \frac{1}{2} \right\}.$$

As such, the matrix $I_{n-1} - E_{n-1} + G_{n-1}$ has only nonzero eigenvalues and hence is nonsingular. As a result, (25) follows from (56) and the nonsingularity of the matrix $I_{n-1} - E_{n-1} + G_{n-1}$. Finally, (26) then follows from (24) and (25).

Appendix C Proof of Theorem 7

Suppose that $\mathbf{c}(t-1)=\mathbf{c}$ and $\mathbf{c}(t)=\mathbf{c}'$, where $t\geq 2$. As c_1' is a data bit and c_i', $i=2,3,\dots,n$, is a stuffed bit if and only if $\bar{c}_{i-1}=c_i=c_{i-1}'=c_i'$ under our sequential bit-stuffing algorithm, we can see from the values of $q(c_i,c_{i-1},c_i',c_{i-1}')$ in Table 3 that

$$(D_n)_{\mathbf{c},\mathbf{c}'}=1+\sum_{i\in I(\mathbf{c},\mathbf{c}')}\log_2\left(\frac{1}{q(c_i,c_{i-1},c_i',c_{i-1}')}\right),\quad (58)$$

where $I(\mathbf{c},\mathbf{c}')=\{2\leq i\leq n: q(c_i,c_{i-1},c_i',c_{i-1}')\neq 0\}$. Note that if $q(c_i,c_{i-1},c_i',c_{i-1}')=0$, then we have from (14) that $(P_n)_{\mathbf{c},\mathbf{c}'}=0$. As such, it follows from (58), the convention $0\log_2 0=0$, and (14) that

$$\begin{aligned}&(P_n)_{\mathbf{c},\mathbf{c}'}(D_n)_{\mathbf{c},\mathbf{c}'}\\&=(P_n)_{\mathbf{c},\mathbf{c}'}\left(1+\sum_{i=2}^n\log_2\left(\frac{1}{q(c_i,c_{i-1},c_i',c_{i-1}')}\right)\right)\\&=-(P_n)_{\mathbf{c},\mathbf{c}'}\log_2\left(\frac{1}{2}\prod_{i=2}^n q(c_i,c_{i-1},c_i',c_{i-1}')\right)\\&=-(P_n)_{\mathbf{c},\mathbf{c}'}\log_2(P_n)_{\mathbf{c},\mathbf{c}'}.\end{aligned}\quad (59)$$

Therefore, (28) follows from (27) and (59).

For $n\geq 2$ and $\mathbf{c}=(c_n,c_{n-1},\dots,c_1)\in\{0,1\}^n$, we denote $\mathbf{c}^{(n-1)}=(c_{n-1},c_{n-2},\dots,c_1)\in\{0,1\}^{n-1}$. From (15), Table 3, and (24) in Lemma 6, we can see that

$$\begin{aligned}&\sum_{c_n,c_n'\in\{0,1\}}\pi_{n,\mathbf{c}}(P_n)_{\mathbf{c},\mathbf{c}'}\\&=\sum_{c_n\in\{0,1\}}\pi_{n,\mathbf{c}}[(P_n)_{\mathbf{c},0\mathbf{c}'^{(n-1)}}+(P_n)_{\mathbf{c},1\mathbf{c}'^{(n-1)}}]\\&=\sum_{c_n\in\{0,1\}}\pi_{n,\mathbf{c}}(P_{n-1})_{\mathbf{c}^{(n-1)},\mathbf{c}'^{(n-1)}}\\&\qquad\times[q(c_n,c_{n-1},0,c_{n-1}')+q(c_n,c_{n-1},1,c_{n-1}')]\\&=\sum_{c_n\in\{0,1\}}\pi_{n,\mathbf{c}}(P_{n-1})_{\mathbf{c}^{(n-1)},\mathbf{c}'^{(n-1)}}\\&=\pi_{n-1,\mathbf{c}^{(n-1)}}(P_{n-1})_{\mathbf{c}^{(n-1)},\mathbf{c}'^{(n-1)}}.\end{aligned}\quad (60)$$

It follows from (28), (15), and (60) that

$$\begin{aligned}D_n=&-\sum_{\mathbf{c},\mathbf{c}'\in\{0,1\}^n}\pi_{n,\mathbf{c}}(P_n)_{\mathbf{c},\mathbf{c}'}\log_2(P_{n-1})_{\mathbf{c}^{(n-1)},\mathbf{c}'^{(n-1)}}\\&-\sum_{\mathbf{c},\mathbf{c}'\in\{0,1\}^n}\pi_{n,\mathbf{c}}(P_n)_{\mathbf{c},\mathbf{c}'}\log_2 q(c_n,c_{n-1},c_n',c_{n-1}')\\=&-\sum_{\mathbf{c}^{(n-1)},\mathbf{c}'^{(n-1)}\in\{0,1\}^{n-1}}\pi_{n-1,\mathbf{c}^{(n-1)}}(P_{n-1})_{\mathbf{c}^{(n-1)},\mathbf{c}'^{(n-1)}}\\&\qquad\times\log_2(P_{n-1})_{\mathbf{c}^{(n-1)},\mathbf{c}'^{(n-1)}}\\&-\sum_{\mathbf{c},\mathbf{c}'\in\{0,1\}^n}\pi_{n,\mathbf{c}}(P_n)_{\mathbf{c},\mathbf{c}'}\log_2 q(c_n,c_{n-1},c_n',c_{n-1}')\\=&D_{n-1}\\&-\sum_{\mathbf{c},\mathbf{c}'\in\{0,1\}^n}\pi_{n,\mathbf{c}}(P_n)_{\mathbf{c},\mathbf{c}'}\log_2 q(c_n,c_{n-1},c_n',c_{n-1}').\end{aligned}\quad (61)$$

Therefore, (29) follows from $r_n=D_n-D_{n-1}$ and (61).

Appendix D Proof of Theorem 8

(i) Let $D_{n-1}(t)$ (resp., $S_{n-1}(t)$) be the event that $c_{n-1}(t)$ is a data (resp., stuffed) bit. Then we have

$$\lim_{t\to\infty}P(D_{n-1}(t))=r_{n-1},\quad (62)$$

$$\lim_{t\to\infty}P(S_{n-1}(t))=\lim_{t\to\infty}(1-P(D_{n-1}(t)))=1-r_{n-1}.\quad (63)$$

As the sequential bit-stuffing algorithm is symmetric, we also have

$$\lim_{t\to\infty}P(c_{n-1}(t)=0)=\lim_{t\to\infty}P(c_{n-1}(t)=1)=\frac{1}{2},\quad (64)$$

$$\begin{aligned}&\lim_{t\to\infty}P(c_{n-1}(t-1)=c_{n-1}(t)=0)\\&=\lim_{t\to\infty}P(c_{n-1}(t-1)=c_{n-1}(t)=1).\end{aligned}\quad (65)$$

Note that

$$\begin{aligned}&P(c_{n-1}(t)=c_{n-1}(t-1))\\&=P(D_{n-1}(t))P(c_{n-1}(t)=c_{n-1}(t-1)|D_{n-1}(t))\\&\quad+P(S_{n-1}(t))P(c_{n-1}(t)=c_{n-1}(t-1)|S_{n-1}(t)).\end{aligned}\quad (66)$$

Given that $c_{n-1}(t)$ is a stuffed bit, we know from the bit-stuffing rule that $c_{n-1}(t)=c_{n-1}(t-1)$, and so we have

$$P(c_{n-1}(t)=c_{n-1}(t-1)|S_{n-1}(t))=1.\quad (67)$$

Given that $c_{n-1}(t)$ is a data bit, $c_{n-1}(t)$ is a Bernoulli random variable with equal probabilities of being 0 or 1, and is conditionally independent of $c_{n-1}(t-1)$, and thus we have

$$\begin{aligned}&P(c_{n-1}(t)=c_{n-1}(t-1)|D_{n-1}(t))\\&=\sum_{c_{n-1}\in\{0,1\}}P(c_{n-1}(t-1)=c_{n-1}|D_{n-1}(t))\\&\qquad\times P(c_{n-1}(t)=c_{n-1}|D_{n-1}(t),c_{n-1}(t-1)=c_{n-1})\\&=\sum_{c_{n-1}\in\{0,1\}}\frac{1}{2}P(c_{n-1}(t-1)=c_{n-1})=\frac{1}{2}.\end{aligned}\quad (68)$$

As such, it follows from (66)–(68) and (62)–(63) that

$$\begin{aligned}&\lim_{t\to\infty}P(c_{n-1}(t)=c_{n-1}(t-1))\\&=\frac{1}{2}\cdot r_{n-1}+1\cdot(1-r_{n-1})=1-\frac{r_{n-1}}{2}.\end{aligned}\quad (69)$$

Therefore, we have from (64), (65), and (69) that

$$\begin{aligned}(P_n')_{0,0}=&\lim_{t\to\infty}P(c_{n-1}(t)=0|c_{n-1}(t-1)=0)\\&\lim_{t\to\infty}\frac{P(c_{n-1}(t)=c_{n-1}(t-1)=0)}{P(c_{n-1}(t-1)=0)}\\&=\frac{\frac{1}{2}(1-\frac{r_{n-1}}{2})}{\frac{1}{2}}=1-\frac{r_{n-1}}{2}.\end{aligned}$$

Similarly, we have $(P_n')_{1,1}=1-\frac{r_{n-1}}{2}$. Finally, $(P_n')_{0,1}=1-(P_n')_{0,0}=\frac{r_{n-1}}{2}$ and $(P_n')_{1,0}=1-(P_n')_{1,1}=\frac{r_{n-1}}{2}$.

(ii) From the assumption in (A1), we immediately see that

$$\begin{aligned}
&(P''_n)_{c_n c_{n-1}, c'_n c'_{n-1}} \\
&= \lim_{t\to\infty} P(c_n(t) = c'_n, c_{n-1}(t) = c'_{n-1} \\
&\qquad\qquad |c_n(t-1) = c_n, c_{n-1}(t-1) = c_{n-1}) \\
&= \lim_{t\to\infty} P(c_{n-1}(t) = c'_{n-1} \\
&\qquad\qquad |c_n(t-1) = c_n, c_{n-1}(t-1) = c_{n-1}) \\
&\qquad \times P(c_n(t) = c'_n | c_{n-1}(t) = c'_{n-1}, \\
&\qquad\qquad c_n(t-1) = c_n, c_{n-1}(t-1) = c_{n-1}) \\
&= \lim_{t\to\infty} P(c_{n-1}(t) = c'_{n-1} | c_{n-1}(t-1) = c_{n-1}) \\
&\qquad \times q(c_n, c_{n-1}, c'_n, c'_{n-1}) \\
&= P'_{c_{n-1}, c'_{n-1}} q(c_n, c_{n-1}, c'_n, c'_{n-1}).
\end{aligned} \tag{70}$$

It is easy to see that (31) follows from (70), (30), and Table 3. By solving $\boldsymbol{\pi}''_n = \boldsymbol{\pi}''_n P''_n$ subject to the constraint that $\pi''_{n,00} + \pi''_{n,01} + \pi''_{n,10} + \pi''_{n,11} = 1$, we obtain (32).

(iii) From (31), (32), and Table 3, we see that

$$\begin{aligned}
r_n &= -\sum_{c_n, c_{n-1} \in \{0,1\}} \pi''_{n, c_n c_{n-1}} \sum_{c'_n, c'_{n-1} \in \{0,1\}} (P''_n)_{c_n c_{n-1}, c'_n c'_{n-1}} \\
&\qquad\qquad \times \log_2 q(c_n, c_{n-1}, c'_n, c'_{n-1}) \\
&= (\pi''_{n,00} + \pi''_{n,11}) \cdot 1 + (\pi''_{n,01} + \pi''_{n,10}) \cdot \left(1 - \frac{r_{n-1}}{2}\right) \\
&= \frac{4}{4 + r_{n-1}}.
\end{aligned}$$

The proof is completed.