Coding Rate Analysis of Forbidden Overlap Codes in High Speed Buses

CHENG-SHANG CHANG, JAY CHENG, TIEN-KE HUANG, DUAN-SHIN LEE and CHENG-YU CHEN, National Tsing Hua University

One of the main problems in deep sub-micron designs of high speed buses is the propagation delay due to the crosstalk effect. To alleviate the crosstalk effect, there are several types of crosstalk avoidance codes proposed in the literature. In this paper, we analyze the coding rates of forbidden overlap codes (FOCs) that avoid "$010 \rightarrow 101$" transition and "$101 \rightarrow 010$" transition on any three adjacent wires in a bus. We first compute the maximum achievable coding rate of FOCs and the maximum coding rate of *memoryless* FOCs. Our numerical results show that there is a significant gap between the maximum coding rate of memoryless FOCs and the maximum achievable rate. We then analyze the coding rates of FOCs generated from the bit-stuffing algorithm. Our worse-case analysis yields a tight lower bound of the coding rate of the bit-stuffing algorithm. Under the assumption of Bernoulli inputs, we use a Markov chain model to compute the coding rate of a bus with n wires under the bit-stuffing algorithm. The main difficulty of solving such a Markov chain model is that the number of states grows exponentially with respect to the number of wires n. To tackle the problem of the curse of dimensionality, we derive an approximate analysis that leads to a recursive closed-form formula for the coding rate over the n^{th} wire. Our approximations match extremely well with the numerical results from solving the original Markov chain for $n \leq 10$ and the simulation results for $n \leq 3000$. Our analysis of coding rates of FOCs could be helpful in understanding the tradeoff between propagation delay and coding rate among various crosstalk avoidance codes in the literature. In comparison with the forbidden transition codes (FTCs) that have shorter propagation delay than that of FOCs, our numerical results show that the coding rates of FOCs are much higher than those of FTCs.

Categories and Subject Descriptors: G.3 [**Probability and Statistics**]: —*Markov processes*; C.4 [**Performance of Systems**]: —*Performance attributes*

General Terms: Performance

Additional Key Words and Phrases: Bus encoding, bit-stuffing algorithm, maximum achievable rate

ACM Reference Format:

C.-S. Chang, J. Cheng, T.-K. Huang and D.-S. Lee, 2015. Coding Rate Analysis of Forbidden Overlap Codes in High Speed Buses. jn 2, 3, Article 1 (March 2015), 25 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

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.
Author's addresses: C.-S. Chang, J. Cheng, T.-K. Huang, D.-S. Lee and C.-Y. 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, lds@cs.nthu.edu.tw, s100060021@m100.nthu.edu.tw.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2015 ACM 0000-0000/2015/03-ART1 $15.00
DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

Table I. The normalized delay of the i^{th} wire T_i/τ_0 for $i \neq 1, n$

bits	000	001	010	011	100	101	110	111
000	0	0	$1+2\lambda$	$1+\lambda$	0	0	$1+\lambda$	1
001	0	0	$1+3\lambda$	$1+2\lambda$	0	0	$1+2\lambda$	$1+\lambda$
010	$1+2\lambda$	$1+3\lambda$	0	0	$1+3\lambda$	$1+4\lambda$	0	0
011	$1+\lambda$	$1+2\lambda$	0	0	$1+2\lambda$	$1+3\lambda$	0	0
100	0	0	$1+3\lambda$	$1+2\lambda$	0	0	$1+2\lambda$	$1+\lambda$
101	0	0	$1+4\lambda$	$1+3\lambda$	0	0	$1+3\lambda$	$1+2\lambda$
110	$1+\lambda$	$1+2\lambda$	0	0	$1+2\lambda$	$1+3\lambda$	0	0
111	1	$1+\lambda$	0	0	$1+\lambda$	$1+2\lambda$	0	0

1. INTRODUCTION

High speed buses are commonly used in many computing and networking systems for information exchange. As the VLSI technology advances, it is possible to pack more wires in high speed buses. However, according to the International Technology Roadmap for Semiconductors (ITRS) [ITRS 2003], even though the gate delay decreases with the shrinking feature size, the global wire delay *increases* as the feature size shrinks. Therefore, the propagation delay through long on-chip buses has become a bottleneck in the overall system performance as the VLSI technology advances into the deep submicrometer (DSM) regime, and such an issue has been identified as one of the Grand Challenges in DSM designs [ITRS 2005].

The propagation delay of a DSM bus is mainly due to the crosstalk effect that arises from the coupling capacitance between adjacent wires in the bus, which in turn depends on the transition patterns on the wires of the bus [Victor 2001; Sridhara 2006]. Note that the speed of a DSM bus is limited by its worst-case propagation delay. To mitigate the crosstalk effect so as to reduce the worst-case propagation delay and hence increase the speed of a DSM bus, it is suggested in the literature that one should avoid certain transition patterns or bit patterns on the wires of the bus. Specifically, for a bus of n parallel wires, it was shown in [Sotiriadis 2002] that the delay of the i^{th} wire, denoted by T_i, can be modelled by the following equation:

$$T_i = \begin{cases} \tau_0[(1+\lambda)\Delta_1^2 - \lambda\Delta_1\Delta_2], & \text{if } i = 1, \\ \tau_0[(1+2\lambda)\Delta_i^2 - \lambda\Delta_i(\Delta_{i-1}+\Delta_{i+1})], & \text{if } i \neq 1, n, \\ \tau_0[(1+\lambda)\Delta_n^2 - \lambda\Delta_n\Delta_{n-1}], & \text{if } i = n, \end{cases} \tag{1}$$

where λ is the ratio of the coupling capacitance between adjacent wires and the loading capacitance between the i^{th} wire and the ground, τ_0 is the delay of a transition on a single wire, and

$$\Delta_i = \begin{cases} 1, & \text{a transition from 0 to 1 on the } i^{th} \text{ wire}, \\ -1, & \text{a transition from 1 to 0 on the } i^{th} \text{ wire}, \\ 0, & \text{no transition on the } i^{th} \text{ wire}. \end{cases} \tag{2}$$

Using (1), one can calculate the normalized delay of the i^{th} wire T_i/τ_0 for $i \neq 1, n$ in Table I. The three bits in the first column denote the *previous* bit values transmitted on the first three wires (the first wire, the second wire and the third wire) and the three bits in the first row denote the *current* bit values transmitted on the first three wires. For instance, if $1, 0, 1$ are the previously transmitted bit values on the first three wires, and they are followed by $0, 1, 0$ on the first three wires, then the delay of transmitting 1 on the second wire is $\tau_0(1+4\lambda)$.

In view of Table I, there are many ways to reduce the maximum delay by avoiding certain transition patterns or bit patterns on the wires of the bus. For instance, one can eliminate $(1+4\lambda)$ from the table if one can construct a set of codewords that avoid the following two types of transitions: $101 \to 010$ and

$010 \to 101$. Codewords that have this property are known as forbidden overlap codes (FOCs) [Wu and Yan 2011] and the maximum delay of FOCs in each wire is bounded by $\tau_0(1 + 3\lambda)$. Similarly, one can avoid the two types of transitions, $10 \to 01$ and $01 \to 10$, to eliminate all the normalized delay larger than $1 + 2\lambda$. Codewords that have this property are known as forbidden transition codes (FTCs) [Wu and Yan 2011; Duan et al. 2008] and the maximum delay of FTCs in each wire is thus bounded by $\tau_0(1 + 2\lambda)$. Alternatively, one can achieve the same delay bound by avoiding the following two patterns in the codewords: 010 and 101. Such codewords are called forbidden pattern codes (FPCs). One-lambda codes (OLCs) [Wu and Yan 2011] can be deduced by further eliminating all the transitions that have delay larger than $\tau_0(1 + 2\lambda)$. All these codes, including FOCs, FTCs, FPCs, and OLCs, are known as crosstalk avoidance codes.

One important performance issue for crosstalk avoidance codes is the tradeoff between coding rate and maximum delay. The coding rate (or throughput) of a crosstalk avoidance code is generally defined as the ratio of the number of (useful) data bits to the number of coded bits over a long period of time. As mentioned before, one can add more restrictions on a set of codewords to reduce the maximum delay and thus increase the bus speed. However, on the other hand, these restrictions also reduce the coding rates of crosstalk avoidance codes. Thus, analyzing the coding rate of a crosstalk avoidance code is crucial in understanding such a tradeoff.

There are several previous works on the coding rates of FTCs in the literature. In particular, the coding rate of the "ground shielding" scheme [Ma and He 2001] (that only transmits data on odd-numbered wires and transmits 0 all the time on even-numbered wires) is 0.5. In [Victor 2001] and [Victor and Keutzer 2001], Victor and Keutzer showed that there exist FTCs that achieve the coding rate $\log_2 \frac{1+\sqrt{5}}{2} \approx 0.6942$ when the number of wires is sufficiently large. Recursive constructions of FTC codes with the coding rate $\log_2 \frac{1+\sqrt{5}}{2}$ were given in [Mutyam 2004; Moision et al. 2001] by using the "Fibonacci representation." By using the Fibonacci numeral system, Duan, Zhu, and Khatri [2008] developed an *explicit* construction of a set of *memoryless* FTCs that achieve the coding rate $\log_2 \frac{1+\sqrt{5}}{2}$. To further improve the coding rate, two-dimensional FTCs with block length larger than 1 were proposed in [Wu et al. 2008], and it was shown that the coding rate could be increased to more than 0.8. However, no explicit constructions of such codes were given in [Wu et al. 2008]. Mutyam [2012] used the transition signaling technique [Stan and Burleson 1994] to construct FTCs that also achieve the coding rate $\log_2 \frac{1+\sqrt{5}}{2}$. In our recent paper [Chang et al. 2015], we proposed and analyzed a bit-stuffing algorithm that can generate FPCs with coding rates larger than 0.82. We also showed that the difference between such a coding rate and the maximum achievable coding rate is only 2.2% for a bus with $n = 10$ parallel wires.

Among all the families of crosstalk avoidance codes, we focus on analyzing the coding rates of FOCs in this paper. Though the restrictions on codewords for FOCs are less stringent than those for FPCs. The analysis for FOCs are much more difficult as the dependency among coded bits in FOCs is much more complicated than that in FPCs. Perhaps, this is one of the reasons that there are few analytical works on FOCs in the literature. In [Wu and Yan 2011], a suboptimal memoryless FOC with (asymptotic) coding rate 0.7925 was constructed. In our recent paper [Chang et al. 2014], we showed an explicit construction of a set of memoryless FOCs that has the largest set of codewords. Such an optimal memoryless FOC has the (asymptotic) coding rate 0.8791. In this paper, we first compute the maximum achievable rate of FOCs in Section 2. For a bus with 10 wires, the maximum achievable rate is 0.9729 and the coding rate of the optimal memoryless FOC is only 0.8977. Clearly, there is still a significant gap between the coding rate of the optimal memoryless FOC and its maximum achievable rate. Analogous to [Chang et al. 2015], we propose using the bit-stuff algorithm in Section 3 to generate FOCs that can achieve higher coding rates. For this bit-stuffing algorithm, we perform a worst-case analysis

Table II. Coding rates of FOCs obtained in this paper: the maximum achievable rate C_n in (7), the maximum coding rate of memoryless FOCs R_n^M, the coding rate per wire R_n for the bit-stuffing algorithm, the coding rate r_n over the n^{th} wire for the bit-stuffing algorithm in (27), and the approximation of r_n in (32) (in Section 6.) for $1 \leq n \leq 10$.

n	1	2	3	4	5	6	7	8	9	10
C_n	1	1	0.9861	0.9816	0.9787	0.9768	0.9754	0.9743	0.9735	0.9729
R_n^M	1	1	0.9358	0.9251	0.9170	0.9099	0.9057	0.9024	0.8998	0.8977
R_n	1	1	0.9815	0.9759	0.9723	0.9698	0.9681	0.9668	0.9657	0.9649
r_n	1	1	0.9444	0.9591	0.9579	0.9575	0.9576	0.9576	0.9576	0.9576
r_n by (32)	1	1	0.9444	0.9588	0.9578	0.9573	0.9575	0.9575	0.9575	0.9575

in Section 4 and a probabilistic analysis in Section 5. Our worse-case analysis shows that the coding rate of our bit-stuffing algorithm is lower bounded by $(2n+2)/3n$ for a bus with $n \geq 4$ wires. Moreover, there exists an input data stream that achieves the lower bound and thus the lower bound is tight. By assuming that the input data stream is a sequence of independent and identically distributed (i.i.d.) Bernoulli random variables (r.v.'s) with equal probabilities of being 0 and 1, the stochastic process of coded bits can be modelled by a Markov chain. By solving the steady state probabilities of the Markov chain, we can then compute the coding rate of our bit-stuffing algorithm. In particular, for a bus with 10 wires, the coding rate of the bit-stuffing algorithm is 0.9649, which is within 0.8% difference when compared with the maximum achievable rate 0.9729.

In Section 6, we tackle the problem of the curse of dimensionality in the Markov chain model that we use in our probabilistic analysis for the coding rate of the bit-stuffing algorithm. In such a Markov chain model, the number of states grows exponentially with respect to the number of wires n. Thus, it is difficult to compute the coding rate of the bit-stuffing algorithm for a large number of wires. To tackle the problem of the curse of dimensionality, our idea is to propose an approximate analysis that limits the dependency of coded bits in adjacent wires. By using various conditional independence properties, we are able to approximate the original Markov chain model for n wires by a Markov chain with only 8 states that in turn can be solved explicitly. Our approximate analysis then leads to a recursive closed-form formula for the coding rate over the n^{th} wire. In particular, for a bus with 10 wires, the coding rate of the 10^{th} wire from our approximate analysis is 0.9575, which matches extremely well with the exact result 0.9576 from solving the original Markov chain model. We also perform simulations for n up to 3000, and the results all agree with 0.9576 for $6 \leq n \leq 3000$.

In Table II, we summarize our numerical results for the coding rates of various FOCs obtained in this paper. In comparison with the numerical results in [Chang et al. 2015] for the coding rates of various FTCs in Table III, it is clear that the coding rates of FOCs are significantly higher than their counterparts of FTCs. For instance, let us compare the coding rates for a bus with 10 wires. The maximum achievable rate (denoted by C_n in both tables) of FOCs is 0.9729, while that of FTCs is only 0.8630. The maximum coding rate of memoryless codes (denoted by R_n^M in both tables) of FOCs is 0.8977, while that of FTCs is only 0.7170. The coding rate per wire from the bit-stuffing algorithm (denoted by R_n in both tables) of FOCs is 0.9649, while that of FTCs is only 0.8432. The coding rate on the 10^{th} wire from the bit-stuffing algorithm (denoted by r_n in both tables) of FOCs is 0.9676, while that of FTCs is only 0.8484. The reason that the coding rates of FOCs are higher than their counterparts of FTCs is because the constraints for the codewords of FTCs are more stringent than those of FOCs. However, on the other hand, the maximum delay of FOCs is only bounded above $\tau_0(1+3\lambda)$ while the maximum delay of FTCs is bounded above by $\tau_0(1+2\lambda)$.

In Table IV, we compare various known crosstalk avoidance codes in the literature, including the "ground shielding" scheme (GS) in [Ma and He 2001], the memoryless FTC (FTC-m) in [Victor 2001; Victor and Keutzer 2001], the sequential bit-stuffing algorithm for FTC (FTC-sb) in [Chang et al.

Table III. Coding rates of FTCs in [Chang et al. 2015]: the maximum achievable rate C_n, the maximum coding rate of memoryless FTCs R_n^M, the coding rate per wire R_n for the bit-stuffing algorithm in [Chang et al. 2015], the coding rate r_n over the n^{th} wire for the bit-stuffing algorithm in [Chang et al. 2015] for $1 \leq n \leq 10$.

n	1	2	3	4	5	6	7	8	9	10
C_n	1	0.9163	0.8941	0.8826	0.8757	0.8712	0.8679	0.8654	0.8635	0.8620
R_n^M	1	0.7925	0.7740	0.7500	0.7401	0.7321	0.7268	0.7227	0.7195	0.7170
R_n	1	0.9	0.8779	0.8653	0.8580	0.8531	0.8495	0.8469	0.8448	0.8432
r_n	1	0.8	0.8338	0.8275	0.8286	0.8284	0.8284	0.8284	0.8284	0.8284

Table IV. Comparison of various known crosstalk avoidance codes in the literature.

Code	GS	FTC-m	FTC-sb	FTC-pb	FTC-2D	FOC-m	FOC-sb	FOC-pb	FPC	OLC
Rate	0.5	0.6942	0.8284	0.8125	0.8250	0.8791	0.9576	0.9544	0.6942	0.4050
Delay$\times\tau_0$	$1+2\lambda$	$1+2\lambda$	$1+2\lambda$	$1+2\lambda$	$1+2\lambda$	$1+3\lambda$	$1+3\lambda$	$1+3\lambda$	$1+2\lambda$	$1+\lambda$
Complexity	$O(1)$	$O(n^2)$	$O(n)$	$O(n)$	unknown	$O(n^2)$	$O(n)$	$O(n)$	$O(n^2)$	$O(n^2)$
Memoryless	Y	Y	N	N	Y	Y	N	N	Y	Y

2015], the parallel bit-stuffing algorithm for FTC (FTC-pb) in [Chang et al. 2015], the two-dimensional FTC (FTC-2D) in [Wu et al. 2008], the memoryless FOC (FOC-m) in [Chang et al. 2014], the sequential bit-stuffing algorithm for FOC (FOC-sb) in this paper, the parallel bit-stuffing algorithm for FOC (FOC-pb) in the conclusion section of this paper, the FPC (FPC) in [Wu and Yan 2011], and the OLC (OLC) in [Wu and Yan 2011]. In the second row of Table IV, we compare the asymptotic coding rates for these schemes (for a large number of wires). Clearly, the coding rates of the three FOC schemes are significantly larger than the other schemes. In particular, the OLC has the smallest coding rate 0.4050 (which is computed by using the recursive equation in [Wu and Yan 2011] for $n = 2500$). In the third row we compare the normalized delay of these schemes. The delays of the FOC schemes are longer than the other schemes. The OLC has the smallest delay among all these schemes. In the fourth row, we compare the implementation complexity of the encoders/decoders of these schemes (as a function of the number of wires). Clearly, the ground shielding (GS) scheme (that only transmits data on odd-numbered wires and transmits 0 all the time on the even-numbered wires) is the simplest scheme and has the $O(1)$ implementation complexity in encoding and decoding. As no explicit constructions (and the associated encoders/decoders) were given for of FTC-2D in [Wu et al. 2008]. Its implementation complexity for a large number of wires is unknown. The encoders and decoders for FTC-m, FOC-m, FPC, and OLC are all based on numerical systems (e.g., the Fibonacci representation) and thus the implementation complexity of their encoders 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). On the other hand, the implementation complexity of the encoders/decoders of the bit-stuffing algorithms is only $O(n)$. In terms of the implementation complexity, the bit-stuffing algorithms are more scalable than the numerical systems. However, the codes generated by the bit-stuffing algorithms are not *memoryless*, as shown in the last row of Table IV. In view of this, error propagation should be contained in bit-stuffing algorithms and that might limit their applicability. As discussed in [Chang et al. 2015], one possible application for using bit-stuffing algorithms for crosstalk avoidance codes is transmitting *variable length packets* through high-speed buses in packet switches. In such an application, bit errors are contained in a single packet.

2. FORBIDDEN OVERLAP CHANNELS

In this section, we model a bus under the constraint that there is no "$010 \rightarrow 101$" transition or "$101 \rightarrow 010$" transition on any three adjacent wires as a forbidden overlap channel defined as follows.

A *forbidden overlap 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 "$010 \to 101$" transition and "$101 \to 010$" transition on any three adjacent wires. Specifically, let $c_i(t)$ be the bit transmitted on the i^{th} wire at time t in a forbidden overlap channel with n parallel wires for $i = 1, 2, \ldots, n$ and $t = 1, 2, \ldots$. Then for the reliable transmissions of the bits $c_i(t)$, $i = 1, 2, \ldots, n$ and $t = 1, 2, \ldots$, over the forbidden overlap channel, we must have

$$\begin{bmatrix} c_{i-2}(t-1) & c_{i-2}(t) \\ c_{i-1}(t-1) & c_{i-1}(t) \\ c_i(t-1) & c_i(t) \end{bmatrix} \neq \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \text{ or } \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}, \tag{3}$$

for $i = 3, 4, \ldots, n$ and $t = 2, 3, \ldots$. Let $\bar{a}$ be the complement of a, i.e., $\bar{a} = 1$ if $a = 0$ and $\bar{a} = 0$ if $a = 1$. It is easy to see that the constraint in (3) is equivalent to the constraint that there are no $3 \leq i \leq n$ and $t \geq 2$ such that

$$\bar{c}_{i-2}(t-1) = c_{i-2}(t) = c_{i-1}(t-1) = \bar{c}_{i-1}(t) = \bar{c}_i(t-1) = c_i(t). \tag{4}$$

In general, for two n-dimensional binary vectors $\mathbf{c} = (c_n, c_{n-1}, \ldots, c_1) \in \{0,1\}^n$ and $\mathbf{c}' = (c'_n, c'_{n-1}, \ldots, c'_1) \in \{0,1\}^n$, we say these two vectors satisfy the forbidden overlap constraint if the constraint in (4) is satisfied when we replace $c_i(t-1)$ by c_i and $c_i(t)$ by c'_i for all $i = 1, 2, \ldots, n$.

2.1 Maximum achievable rate

In this section, we first perform the analysis for the maximum achievable rate of the forbidden overlap channel. 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 (3) up to time t. Then it is well-known (see e.g., [Shannon 1948; Cover and Thomas 1991; Orcutt and Marcellin 1993; Weeks and Blahut 1998]) that the maximum achievable rate of a forbidden overlap channel with n parallel wires is the entropy rate of these constrained sequences (see e.g., pp. 94 of [Cover and Thomas 1991]) and this is given by

$$C_n = \frac{1}{n} \lim_{t \to \infty} \frac{\log_2 X_n(t)}{t} \text{ bits/wire/time unit.} \tag{5}$$

Note that the maximum achievable rate in (5) is achieved when every sequence is selected with an equal probability. To compute $X_n(t)$, let us order the n-dimensional binary vector $\mathbf{c} = (c_n, c_{n-1}, \ldots, c_1) \in \{0,1\}^n$ in the standard lexicographic order, i.e., in the increasing order of the integers $\sum_{i=1}^{n} c_i 2^{i-1} \in \{0, 1, \ldots, 2^n - 1\}$. Denote by $\mathbf{0}_n = (0, 0, \ldots, 0)$ (resp., $\mathbf{1}_n = (1, 1, \ldots, 1)$) the n-vector whose entries are all equal to 0 (resp., 1). Also, let $X_{n,\mathbf{c}}(t)$ be the number of sequences $(\mathbf{c}(1), \mathbf{c}(2), \ldots, \mathbf{c}(t))$ that satisfy the constraint in (3) up to time t and end in the state $\mathbf{c}$ at time t, i.e., $\mathbf{c}(t) = \mathbf{c}$, and $\mathbf{X}_n(t) = (X_{n,\mathbf{0}_n}(t), \ldots, X_{n,\mathbf{1}_n}(t))$ be the 2^n-vector whose entries are the numbers $X_{n,\mathbf{c}}(t)$, $\mathbf{c} \in \{0,1\}^n$, ordered in the standard lexicographic order with respect to the states $\mathbf{c}$. From the definition of $X_{n,\mathbf{c}}(t)$, clearly we have

$$X_n(t) = \sum_{\mathbf{c} \in \{0,1\}^n} X_{n,\mathbf{c}}(t) = \mathbf{X}_n(t) \mathbf{1}_{2^n}^T,$$

where $\mathbf{1}_{2^n}^T$ is the 2^n-column vector with all its elements being 1. Furthermore, it is straightforward to see that $\mathbf{X}_n(t)$ can be derived recursively by

$$\mathbf{X}_n(t) = \mathbf{X}_n(t-1) \mathbf{A}_n, \tag{6}$$

where $\mathbf{A}_n$ is the $2^n \times 2^n$ adjacency matrix associated with the forbidden overlap channel with n parallel wires, i.e., $(\mathbf{A}_n)_{\mathbf{c},\mathbf{c}'} = 1$ if the forbidden overlap constraint is satisfied for the two n-vectors $\mathbf{c}$ and $\mathbf{c}'$, and $(\mathbf{A}_n)_{\mathbf{c},\mathbf{c}'} = 0$ otherwise.

In view of the recursive equation in (6), it follows from the Perron-Frobenius theorem (see e.g., [Horn and Johnson 1985]) that the maximum achievable rate C_n in (5) can also be given by

$$C_n = \frac{1}{n} \log_2 \lambda_{n,\max}, \tag{7}$$

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

As in [Weeks and Blahut 1998] for the study of the capacity of certain checkerboard codes, we first identify a recursive expression for the adjacency matrix $\mathbf{A}_n$ in the following lemma. Its proof is given in Appendix A.

LEMMA 2.1. *Let $\mathbf{O}_i$ be the zero matrix of size $2^i \times 2^i$ for $i \geq 0$. Then the adjacency matrix $\mathbf{A}_n$ associated with the forbidden overlap channel is a symmetric matrix and it can be given recursively by*

$$\mathbf{A}_n = \begin{bmatrix} \mathbf{A}_{n-1} & \mathbf{B}_{n-1} \\ \mathbf{B}_{n-1}^T & \mathbf{A}_{n-1} \end{bmatrix}, \text{ for } n \geq 1, \tag{8}$$

where $\mathbf{A}_0 = \mathbf{B}_0 = 1$ and

$$\mathbf{B}_{n-1} = \begin{bmatrix} \mathbf{A}_{n-2} & \mathbf{B}_{n-2} \\ \mathbf{C}_{n-2} & \mathbf{A}_{n-2} \end{bmatrix}, \text{ for } n \geq 2, \tag{9}$$

in which $\mathbf{C}_0 = 1$ and

$$\mathbf{C}_{n-2} = \begin{bmatrix} \mathbf{A}_{n-3} & \mathbf{O}_{n-3} \\ \mathbf{B}_{n-3}^T & \mathbf{A}_{n-3} \end{bmatrix}, \text{ for } n \geq 3. \tag{10}$$

In the second row of Table II, we use the recursion in Lemma 2.1 and (7) to compute the maximum achievable rate C_n of a forbidden overlap channel with n parallel wires for $1 \leq n \leq 10$.

2.2 Maximum coding rate of memoryless forbidden overlap codes

Note that the maximum achievable rate C_n serves as the fundamental limit on the coding rate of any FOC and it is achieved with an infinite code length ($t \to \infty$ in (5)). In general, the implementation complexity of encoders/decoders increases with respect to the length of codes. The simplest one is the *memoryless* codes that has the code length 1. An explicit construction of the memoryless FOC that has the largest set of codewords was recently reported in [Chang et al. 2014]. The key insight of the construction in [Chang et al. 2014] is that the forbidden overlap constraint in (4) for *memoryless* FOCs is equivalent to the constraint that there are no 3 consecutive 1's in each "modified" codeword when the even numbered bits of the original codeword are *inverted*. As such, one can first construct a set of modified codewords by using a greedy numeral system, called the C-transform in [Chou et al. 2006; Cheng et al. 2008], to generate binary representations for integers that do not have 3 consecutive 1's. Then invert every even numbered bit to construct the desired FOC.

The maximum number of n-bit codewords that do not have 3 consecutive 1's, denoted by N_n, can be derived from the recursive equation

$$N_n = N_{n-1} + N_{n-2} + N_{n-3}, \tag{11}$$

with $N_1 = 2$, $N_2 = 4$ and $N_3 = 7$. This is because we can obtain an n-bit codeword by adding a prefix 0 in front of the $(n-1)$-bit codewords, a prefix 10 in front of the $(n-2)$-bit codewords and a prefix 110 in front of the $(n-3)$-bit codewords. The maximum coding rate of the n-bit memoryless FOCs, denoted by R_n^M, is then $\frac{1}{n}\log_2(N_n)$. Letting $n \to \infty$ yields the maximum asymptotic coding rate 0.8791 for memoryless FOCs. In the third row of Table II, we also show the maximum coding rate of memoryless FOCs for $1 \leq n \leq 10$. Clearly there is still a significant gap between the maximum coding rate of memoryless FOCs and the maximum achievable rate. In the next section, we will use the bit-stuffing algorithm to generate FOCs that have much higher coding rates than those of the memoryless codes.

3. THE BIT-STUFFING ALGORITHM

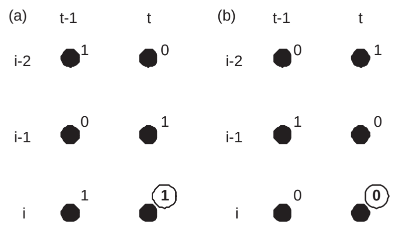

Fig. 1. Illustration of the bit-stuffing algorithm: (a) If the coded bits $c_{i-2}(t-1) = 1$, $c_{i-2}(t) = 0$, $c_{i-1}(t-1) = 0$, $c_{i-1}(t) = 1$ and $c_i(t-1) = 1$, then the coded bit $c_i(t) = 1$ is a stuffed bit. (b) If the coded bits $c_{i-2}(t-1) = 0$, $c_{i-2}(t) = 1$, $c_{i-1}(t-1) = 1$, $c_{i-1}(t) = 0$ and $c_i(t-1) = 0$, then the coded bit $c_i(t) =$ is a stuffed bit.

In this section, we use the bit-stuffing encoding scheme for generating forbidden overlap codes. The idea of our bit-stuffing algorithm is to add a redundant bit on the i^{th} wire at time t if $\bar{c}_{i-2}(t-1) = c_{i-2}(t) = c_{i-1}(t-1) = \bar{c}_{i-1}(t) = \bar{c}_i(t-1)$, 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 forbidden overlap constraint in (3). There are two cases for this and they are illustrated in Figure 1: (a) If the coded bits $c_{i-2}(t-1) = 1$, $c_{i-2}(t) = 0$, $c_{i-1}(t-1) = 0$, $c_{i-1}(t) = 1$ and $c_i(t-1) = 1$, then the coded bit $c_i(t) = 1$ is a stuffed bit. (b) If the coded bits $c_{i-2}(t-1) = 0$, $c_{i-2}(t) = 1$, $c_{i-1}(t-1) = 1$, $c_{i-1}(t) = 0$ and $c_i(t-1) = 0$, then the coded bit $c_i(t) =$ is a stuffed bit.

This bit-stuffing algorithm is described below.

ALGORITHM 1: Encoder–Bit-stuffing algorithm

```
Given an input data bit stream $\{b_1, b_2, \ldots\}$. Initially at time $t = 1$, set $c_i(1) = b_i$ for $i = 1, 2, \ldots, n$. For every time $t \geq 2$, generate the coded bit $c_i(t)$ by the following bit-stuffing rule for $i = 1, 2, \ldots, n$:
(i) Set $c_1(t)$ and $c_2(t)$ as the next two input data bits.
(ii) For $i = 3, 4, \ldots, n$,
        (a). (Bit-stuffing condition) If $\bar{c}_{i-2}(t-1) = c_{i-2}(t) = c_{i-1}(t-1) = \bar{c}_{i-1}(t) = \bar{c}_i(t-1)$, then we set
    $c_i(t) = c_{i-1}(t)$, i.e., $c_i(t)$ is a stuffed bit.
        (b). Otherwise, set $c_i(t)$ as the next input data bit.
```

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

ALGORITHM 2: Decoder–the bit-removing algorithm

```
Given received coded bits $c_i(t)$, $i = 1, 2, \dots, n$ and $t = 1, 2, \dots$. Initially, set $b_i = c_i(1)$ for $i = 1, 2, \dots, n$. For every time $t \geq 2$, decode the received coded bit $c_i(t)$, $i = 1, 2, \dots, n$, by the following bit-removing rule:
(i) Decode $c_1(t)$ and $c_2(t)$ as the next data bit.
(ii) For $i = 3, \dots, n$,
    (a). (Bit-removing condition) If $\bar{c}_{i-2}(t-1) = c_{i-2}(t) = c_{i-1}(t-1) = \bar{c}_{i-1}(t) = \bar{c}_i(t-1)$, then $c_i(t)$ is a stuffed bit and it is discarded.
    (b). Otherwise, decode $c_i(t)$ as the next data bit.
```

In the following lemma, we show an interesting property of the bit-stuffing algorithm, called *no adjacent stuffed bits* property in this paper. This property states that a set of "neighboring" bits of a stuffed bit must be data bits (see the illustration in Figure 2). Such a property will be used to derive a lower bound on the coding rate of the bit-stuffing algorithm.

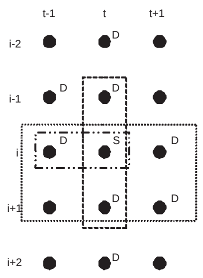

Fig. 2. If the coded bit $c_i(t)$ is a stuffed bit (marked with a "S"), then the coded bits $c_{i-2}(t)$, $c_{i-1}(t-1)$, $c_{i-1}(t)$, $c_i(t-1)$, $c_i(t+1)$, $c_{i+1}(t)$ and $c_{i+1}(t+1)$, and $c_{i+2}(t)$ are all data bits (marked with a "D").

LEMMA 3.1. ***(No adjacent stuffed bits property)*** *Under the bit-stuffing algorithm, if the coded bit $c_i(t)$ is a stuffed bit, then the coded bits $c_{i-2}(t)$, $c_{i-1}(t-1)$, $c_{i-1}(t)$, $c_i(t-1)$, $c_i(t+1)$, $c_{i+1}(t)$ and $c_{i+1}(t+1)$ (in the case that $i \leq n-1$), and $c_{i+2}(t)$ (in the case that $i \leq n-2$) 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 (4) that $3 \leq i \leq n$, $t \geq 2$, and

$$\bar{c}_{i-2}(t-1) = c_{i-2}(t) = c_{i-1}(t-1) = \bar{c}_{i-1}(t) = \bar{c}_i(t-1) = \bar{c}_i(t). \tag{12}$$

In the following, we show by contradiction that $c_{i-2}(t)$ cannot be a stuffed bit. The proof that $c_{i-1}(t-1)$, $c_{i-1}(t)$, $c_i(t-1)$, $c_i(t+1)$, $c_{i+1}(t)$ and $c_{i+1}(t)$ (in the case that $i \leq n-1$), and $c_{i+2}(t)$ (in the case that $i \leq n-2$) cannot be stuffed bits is similar.

Suppose that $c_{i-2}(t)$ is a stuffed bit. Then it follows from (4) (with i replaced by $i-2$) that $3 \leq i-2 \leq n$, $t \geq 2$, and

$$\bar{c}_{i-4}(t-1) = c_{i-4}(t) = c_{i-3}(t-1) = \bar{c}_{i-3}(t) = \bar{c}_{i-2}(t-1) = \bar{c}_{i-2}(t). \tag{13}$$

As we have $\bar{c}_{i-2}(t-1) = c_{i-2}(t)$ in (12) and $\bar{c}_{i-2}(t-1) = \bar{c}_{i-2}(t)$ in (13), a contradiction is reached. ■

From Lemma 3.1, we obtain the following properties on the number of stuffed bits in certain blocks of coded bits (see the illustration in Figure 2).

LEMMA 3.2. *Under the bit-stuffing algorithm, we have that*
(i) For any $3 \leq i \leq n$ and $t \geq 2$, there is at most one stuffed bit among $c_i(t)$ and $c_i(t+1)$.

(ii) For any $3 \leq i \leq n-2$ and $t \geq 2$, there is at most one stuffed bit among $c_i(t)$, $c_{i+1}(t)$, and $c_{i+2}(t)$.

(iii) For any $3 \leq i \leq n-1$ and $t \geq 2$, there are at most two stuffed bits among $c_i(t)$, $c_i(t+1)$, $c_i(t+2)$, $c_{i+1}(t)$, $c_{i+1}(t+1)$, and $c_{i+1}(t+2)$.

Proof. The proof of (i) and (ii) is omitted as they follow immediately from Lemma 3.1.

(iii) Consider the coded bits $c_i(t)$, $c_i(t+1)$, $c_i(t+2)$, $c_{i+1}(t)$, $c_{i+1}(t+1)$, and $c_{i+1}(t+2)$. In the following, we show that given $c_i(t)$ is a stuffed bit, there are at most two stuffed bits among the six coded bits. The proof for other cases that one of the coded bits $c_i(t+1)$, $c_i(t+2)$, $c_{i+1}(t)$, $c_{i+1}(t+1)$, and $c_{i+1}(t+2)$ is a stuffed bit is similar.

If $c_i(t)$ is a stuffed bit, then it follows from Lemma 3.1 that $c_i(t+1)$, $c_{i+1}(t)$, $c_{i+1}(t+1)$ are all data bits. Now, there is at most one stuffed bit among the two coded bits $c_i(t+2)$ and $c_{i+1}(t+2)$ by Lemma 3.1. Therefore, we have at most two stuffed bits among the coded bits $c_i(t)$, $c_i(t+1)$, $c_i(t+2)$, $c_{i+1}(t)$, $c_{i+1}(t+1)$, and $c_{i+1}(t+2)$. ■

4. WORST-CASE ANALYSIS

In the following theorem, we derive a tight lower bound on the coding rate of our bit-stuffing encoding scheme for forbidden overlap channels. The coding rate is defined as the (asymptotic) ratio of the number of data bits to the total number of coded bits.

THEOREM 4.1. *The coding rate R_n of our bit-stuffing encoding scheme for any input data bit stream over a forbidden overlap channel with n parallel wires satisfies*

$$R_1 = R_2 = 1 \text{ and } R_n \geq \begin{cases} \frac{5}{6}, & \text{if } n = 3, \\ \frac{2n+2}{3n}, & \text{if } n \geq 4, \end{cases} \text{ bits/wire/time unit.} \tag{14}$$

Furthermore, the lower bound in (14) is tight as there exists an input data bit stream that achieves the lower bound. In other words, the worst case coding rate R_n^* of our bit-stuffing encoding scheme for forbidden overlap channels is given by $R_1^* = R_2^* = 1$, $R_3^* = \frac{5}{6}$, and $R_n^* = \frac{2n+2}{3n}$ for $n \geq 4$.

Proof. (i) Consider a forbidden overlap channel with n wires and with an input data bit stream $\{b_1, b_2, \ldots\}$. From the bit-stuffing algorithm, it is clear that the n coded bits at time $t = 1$ and all of the coded bits on the first two wires are data bits, i.e., $c_i(t)$ is a data bit for $i = 1, 2$ or $t = 1$. It follows immediately that $R_1 = R_2 = 1$.

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 $t' = 1, 2, \ldots, t$. For the case that $n = 3$, we have from Lemma 3.2(i) that at least one of the two coded bits $c_3(t')$ and $c_3(t'+1)$ is a data bit for $t' = 2, 3, \ldots$. Thus, we have

$$R_3(t) \geq 3 + 2(t-1) + \left\lfloor \frac{t-1}{2} \right\rfloor. \tag{15}$$

For the case that $n = 2k+1$, where $k \geq 2$, it follows from Lemma 3.2(iii) that there are at least four data bits among $c_i(t)$, $c_i(t+1)$, $c_i(t+2)$, $c_{i+1}(t)$, $c_{i+1}(t+1)$, and $c_{i+1}(t+2)$, for $i = 3, 5, \ldots, 2k-3$ and $t = 2, 5, \ldots$, and from Lemma 3.2(ii) that there are at least two data bits among $c_{2k-1}(t)$, $c_{2k}(t)$, $c_{2k+1}(t)$ for $t = 2, 3, \ldots$. Thus, we have

$$\begin{aligned} R_{2k+1}(t) &\geq (2k+1) + 2(t-1) + 4(k-2) \left\lfloor \frac{t-1}{3} \right\rfloor + 2(t-1) \\ &= (2k+1) + 4(t-1) + 4(k-2) \left\lfloor \frac{t-1}{3} \right\rfloor. \end{aligned} \tag{16}$$

Table V. A worst case for $n = 3$ and $t = 1, 2, \ldots, 12$. Note that the stuffed bits are in boldface.

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

Similarly, for the case that $n = 2k$, where $k \geq 2$, we have

$$R_{2k}(t) \geq 2k + 2(t-1) + 4(k-1)\left\lfloor \frac{t-1}{3} \right\rfloor. \tag{17}$$

It follows from (15)–(17) that

$$R_n = \frac{1}{n}\lim_{t\to\infty}\frac{R_n(t)}{t} \geq \begin{cases} \frac{5}{6}, & \text{if } n = 3, \\ \frac{2n+2}{3n}, & \text{if } n \geq 4. \end{cases}$$

(ii) Now, we give input data bit streams that achieve the lower bounds in (14) for $n \geq 3$. As the input data bits can be uniquely decoded from their coded bits under the bit-stuffing encoding scheme, it suffices to show the coded bits of the input data bits.

For the case that $n = 3$, let $\mathbf{c}(t) = (c_3(t), c_2(t), c_1(t))$ be given by $\mathbf{c}(t) = (0, 1, 0)$ for $t = 1, 3, \ldots$, and $\mathbf{c}(t) = (1, 0, 0)$ for $t = 2, 4, \ldots$. In Table V, we show the coded bits $c_i(t)$ for $i = 1, 2, 3$ and $t = 1, 2, \ldots, 12$. It is easy to see that $c_i(t)$, $i = 1, 2, 3$ and $t = 1, 2, \ldots$, are valid coded bits as there are no forbidden overlaps on the three wires. Furthermore, $c_i(t)$ is a stuffed bit if and only if $i = 3$ and $t = 2, 4, \ldots$. As such, the number of data bits $R_3(t)$ among the first $3t$ coded bits is given by the right-hand side of (15), and it follows that

$$R_3 = \frac{1}{3}\lim_{t\to\infty}\frac{R_3(t)}{t} = \frac{5}{6}.$$

For the case that $n \geq 4$, let

$$\begin{aligned} \mathbf{x} &= (0, 1, 0, 0, 1, 0, 0, 1, 0, \ldots), \\ \mathbf{y} &= (1, 0, 0, 1, 0, 0, 1, 0, 0, \ldots), \\ \mathbf{z} &= (0, 0, 1, 0, 0, 1, 0, 0, 1, \ldots) \end{aligned}$$

be three infinite binary periodic sequences with period 3. For $1 \leq i \leq n$, let

$$(c_i(1), c_i(2), \ldots) = \begin{cases} \mathbf{x}, & \text{if } i = 1 \pmod 3, \\ \mathbf{y}, & \text{if } i = 2 \pmod 3, \\ \mathbf{z}, & \text{if } i = 0 \pmod 3. \end{cases} \tag{18}$$

In Table VI, we show the coded bits $c_i(t)$ given by (18) for $1 \leq i \leq n$ and $1 \leq t \leq 12$, where $n = 9$. It is clear that $c_i(t)$ given by (18) is periodic with period 3 (both in space and in time).

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 forbidden overlaps on any three adjacent wires. Furthermore, $c_i(t)$ is a stuffed bit if and only if $i \geq 3$, $t \geq 2$, and $i + t = 2 \pmod 3$ (note that the stuffed bits in Table VI are in boldface). As such, the number of data bits $R_n(t)$ among the first nt coded bits satisfies (16), (17), and

$$R_n(t) \leq \begin{cases} n + 4(t-1) + 2(n-5)\left\lfloor \frac{t-1}{3} \right\rfloor + 2(n-2), & \text{if } n = 2k+1, \text{ where } k \geq 2, \\ n + 2(t-1) + 2(n-2)\left\lfloor \frac{t-1}{3} \right\rfloor + 2(n-2), & \text{if } n = 2k, \text{ where } k \geq 2, \end{cases} \tag{19}$$

It follows that

$$R_n = \frac{1}{n}\lim_{t\to\infty}\frac{R_n(t)}{t} = \frac{2n+2}{3n}$$

Table VI. The coded bits $c_i(t)$ given by (18) for $1 \leq i \leq n$ and $1 \leq t \leq 12$, where $n = 9$. Note that the stuffed bits are in boldface.

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

for $n \geq 4$. The proof is completed. ■

5. PROBABILISTIC ANALYSIS

In this section, we give a probabilistic analysis for the coding rate of our bit-stuffing algorithm. For our probabilistic analysis, we 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*. Under such an assumption, it is easy to see that the stochastic process $\{\mathbf{c}(t), t \geq 1\}$ is a time-homogeneous Markov chain. Let $\mathbf{P}_n$ be the transition probability matrix of the Markov chain $\{\mathbf{c}(t), t \geq 1\}$. Now we specify the transition probability $(\mathbf{P}_n)_{\mathbf{c},\mathbf{c}'} = P(\mathbf{c}(t) = \mathbf{c}'|\mathbf{c}(t-1) = \mathbf{c})$ for $\mathbf{c}, \mathbf{c}' \in \{0,1\}^n$. As the input data bits are i.i.d. Bernoulli random variables with equal probabilities of being 0 or 1, it is easy to see from the bit-stuffing algorithm that

$$\mathbf{P}_1 = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix}, \mathbf{P}_2 = \begin{bmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{bmatrix}. \tag{20}$$

For $n \geq 3$, we have from the chain rule and the bit-stuffing rule that

$$\begin{aligned}(\mathbf{P}_n)_{\mathbf{c},\mathbf{c}'} =& P(\mathbf{c}(t) = \mathbf{c}'|\mathbf{c}(t-1) = \mathbf{c}) \\ =& \prod_{i=1}^{n} P(c_i(t) = c_i'|c_j(t-1) = c_j', j = 1, 2, \ldots, i-1, \mathbf{c}(t-1) = \mathbf{c}) \\ =& \frac{1}{4}\prod_{i=3}^{n} P(c_i(t) = c_i'|c_i(t-1) = c_i, c_{i-1}(t-1) = c_{i-1}, c_{i-1}(t) = c_{i-1}', \\ & \qquad c_{i-2}(t-1) = c_{i-2}, c_{i-2}(t) = c_{i-2}') \\ =& \frac{1}{4}\prod_{i=3}^{n} q(c_i, c_{i-1}, c_{i-2}, c_i', c_{i-1}', c_{i-2}'),\end{aligned} \tag{21}$$

where

$$\begin{aligned}& q(c_i, c_{i-1}, c_{i-2}, c_i', c_{i-1}', c_{i-2}') \\ &= P(c_i(t) = c_i'|c_i(t-1) = c_i, c_{i-1}(t-1) = c_{i-1}, c_{i-1}(t) = c_{i-1}', c_{i-2}(t-1) = c_{i-2}, c_{i-2}(t) = c_{i-2}')\end{aligned} \tag{22}$$

is a function of $c_i, c_{i-1}, c_{i-2}, c_i', c_{i-1}', c_{i-2}'$ and it is independent of time t. The values of $q(c_i, c_{i-1}, c_{i-2}, c_i', c_{i-1}', c_{i-2}')$ are shown in Table VII. Note that there are two entries of $q(c_i, c_{i-1}, c_{i-2}, c_i', c_{i-1}', c_{i-2}')$ that

Table VII. The values of $q(c_i, c_{i-1}, c_{i-2}, c'_i, c'_{i-1}, c'_{i-2})$.

$c_i c_{i-1} c_{i-2} \backslash c'_i c'_{i-1} c'_{i-2}$	000	001	010	011	100	101	110	111
000	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
001	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
010	$\frac{1}{2}$	1	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	0	$\frac{1}{2}$	$\frac{1}{2}$
011	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
100	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
101	$\frac{1}{2}$	$\frac{1}{2}$	0	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	1	$\frac{1}{2}$
110	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
111	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$

are equal to 0. These two entries correspond to the forbidden overlap constraint in (3). Also, there are two entries of $q(c_i, c_{i-1}, c_{i-2}, c'_i, c'_{i-1}, c'_{i-2})$ that are equal to 1. In each of these two entries, the coded bit c'_i is a *stuffed* bit. The rest of the entries are all equal to 1/2 (from the assumption of Bernoulli i.i.d. random variables with equal probabilities of being 0 and 1) as the coded bit c'_i is a *data* bit in each of these entries.

Since the state with all its values being 0 is able to communicate with any other states, i.e., $(\mathbf{P}_n)_{\mathbf{0}_n,\mathbf{c}} > 0$ and $(\mathbf{P}_n)_{\mathbf{c},\mathbf{0}_n} > 0$ for all $\mathbf{c} \in \{0,1\}^n$, it follows that the Markov chain $\{\mathbf{c}(t), t \geq 1\}$ is irreducible and aperiodic. As its state space $\{0,1\}^n$ is finite, it is well-known [Nelson 1995] that there exist unique steady state probabilities $\boldsymbol{\pi}_n = (\pi_{n,\mathbf{0}_n}, \dots, \pi_{n,\mathbf{1}_n})$ for the Markov chain $\{\mathbf{c}(t), t \geq 1\}$ that could be obtained by solving the following system of linear equations:

$$\boldsymbol{\pi}_n = \boldsymbol{\pi}_n \mathbf{P}_n, \tag{23}$$

$$\sum_{\mathbf{c}\in\{0,1\}^n} \pi_{n,\mathbf{c}} = 1. \tag{24}$$

Once we solve the steady state probabilities of the Markov chain, we can use those to compute the coding rate. In particular, the total coding rate D_n over the n wires is equal to the entropy rate $H(\mathbf{P}_n)$ of the Markov chain $\{\mathbf{c}(t), t \geq 1\}$ [Cover and Thomas 1991], that is,

$$D_n = H(\mathbf{P}_n) = -\sum_{\mathbf{c},\mathbf{c}'\in\{0,1\}^n} \pi_{n,\mathbf{c}} (\mathbf{P}_n)_{\mathbf{c},\mathbf{c}'} \log_2 (\mathbf{P}_n)_{\mathbf{c},\mathbf{c}'}. \tag{25}$$

Using (25), one can then compute the coding rate per wire

$$R_n = \frac{D_n}{n}. \tag{26}$$

Also, the coding rate r_n over the n^{th} wires is given by $r_1 = r_2 = 1$ and

$$r_n = -\sum_{\mathbf{c},\mathbf{c}'\in\{0,1\}^n} \pi_{n,\mathbf{c}} (\mathbf{P}_n)_{\mathbf{c},\mathbf{c}'} \log_2 q(c_n, c_{n-1}, c_{n-2}, c'_n, c'_{n-1}, c'_{n-2}), \text{ for } n \geq 3, \tag{27}$$

where the values of $q(c_n, c_{n-1}, c_{n-2}, c'_n, c'_{n-1}, c'_{n-2})$ are given in Table VII.

In Table II, we show the maximum achievable rate C_n in (7), the coding rate per wire $R_n = \frac{D_n}{n}$, the coding rate r_n over the n^{th} wire in (27), and the approximation of r_n in (32) (this will be given in Section 6) for $1 \leq n \leq 10$. Several observations can be drawn from these numerical results: (i) There is still some gap between the coding rate R_n of the bit-stuffing encoding scheme and the maximum achievable rate C_n. This shows that the bit-stuffing encoding scheme does not achieve the maximum achievable rate when n is small. However, the difference is very small. For the case with $n = 10$, the difference is only 0.8%. (ii) It seems that the coding rate r_n over the n^{th} wire converges to a constant near 0.9576, i.e., $\lim_{n\to\infty} r_n \approx 0.9576$ (this will be further addressed in Section 6).

6. APPROXIMATE PROBABILISTIC ANALYSIS

In the previous section, we derive a Markov chain model to compute the coding rate of the bit-stuffing algorithm. However, the number of states grows exponentially with respect to the number of wires n. Thus, it is difficult to compute the coding rate of the bit-stuffing algorithm for a large number of wires. To tackle the problem of the curse of dimensionality, our idea is to propose an approximate analysis that limits the dependency of coded bits in adjacent wires as in [Chang et al. 2015]. Note from the bit-stuffing condition in our bit-stuffing algorithm, whether the coded bit $c_i(t)$ is a stuffed bit or a data bit depends on the other five coded bits $c_{i-2}(t-1)$, $c_{i-1}(t-1)$, $c_i(t-1)$, $c_{i-2}(t)$ and $c_{i-1}(t)$. In Figure 3, we show the dependency graph of coded bits on the five wires: the $(n-4)^{th}$ wire, the $(n-3)^{th}$ wire, the $(n-2)^{th}$ wire, the $(n-1)^{th}$ wire and the n^{th} wire. If we neglect the dependency from the coded bits on the $(n-4)^{th}$ wire and the $(n-3)^{th}$ wire, then it is much easier to analyze the remaining dependency graph (the subgraph under the dotted line in Figure 3) for the coded bits on the $(n-2)^{th}$ wire, the $(n-1)^{th}$ wire and the n^{th} wire. In view of the remaining dependency graph, we have the following properties:

(P1). The stochastic process $\{c_{n-2}(t),\ t \geq 1\}$ is a Markov chain. Specifically, given $c_{n-2}(t-1)$, the coded bit $c_{n-2}(t)$ is conditionally independent of the coded bits $c_{n-2}(\tau), \tau = 1, 2, \ldots, t-2$.

(P2). The stochastic process $\{(c_{n-1}(t), c_{n-2}(t)),\ t \geq 1\}$ is a Markov chain. Specifically, given $(c_{n-1}(t-1), c_{n-2}(t-1))$, the coded bits $(c_{n-1}(t), c_{n-2}(t))$ are conditionally independent of the coded bits $(c_{n-1}(\tau), c_{n-2}(\tau)), \tau = 1, 2, \ldots, t-2$.

(P3). The stochastic process $\{(c_n(t),\ c_{n-1}(t), c_{n-2}(t)),\ t \geq 1\}$ is a Markov chain. Specifically, given $(c_n(t-1), c_{n-1}(t-1), c_{n-2}(t-1))$, the coded bits $(c_n(t), c_{n-1}(t), c_{n-2}(t))$ are conditionally independent of the coded bits $(c_n(\tau), c_{n-1}(\tau), c_{n-2}(\tau)), \tau = 1, 2, \ldots, t-2$.

(P4). Given $c_{n-2}(t-1)$, the coded bits $c_{n-2}(t)$ and $c_{n-1}(t-1)$ are conditionally independent.

(P5). Given $(c_{n-1}(t-1), c_{n-2}(t-1))$, the coded bits $(c_{n-1}(t), c_{n-2}(t))$ and $c_n(t-1)$ are conditionally independent.

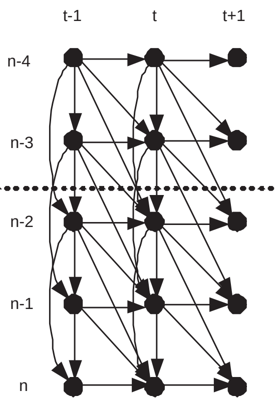

Fig. 3. The dependency graph of coded bits. The approximate probability analysis is to neglect the dependency above the dotted line.

We note that the approximate analysis for FPCs in [Chang et al. 2015] neglects the dependency from the coded bits above the $(n-2)^{th}$ wire. In comparison with the approximate analysis for FPCs in [Chang et al. 2015], our approximate analysis for FOCs is much more difficult as it adds another level of dependency in the dependency graph of coded bits in Figure 3. Also, these five properties (P1)-(P5) are in fact true for $n = 3$. However, it is only an approximation for $n > 3$. In fact, the stochastic processes $\{c_{n-2}(t),\ t \geq 1\}$, $\{(c_{n-1}(t), c_{n-2}(t)),\ t \geq 1\}$, and $\{(c_n(t), c_{n-1}(t), c_{n-2}(t)),\ t \geq 1\}$ are hidden Markov chains (Markov modulated processes) and they are not necessarily Markov chains. However, as will be seen shortly, these five properties lead to a very good approximation for the coding rate r_n for the n^{th} wire when n is large.

Our analysis consists of the following five steps:

(S1) Derive the transition probability matrix $\mathbf{P}'_n = [(\mathbf{P}'_n)_{c_{n-2},c'_{n-2}}]_{c_{n-2},c'_{n-2}\in\{0,1\}}$ for the Markov chain $\{c_{n-2}(t),\ t \geq 1\}$. This is given by

$$\mathbf{P}'_n = \begin{bmatrix} 1 - \frac{r_{n-2}}{2} & \frac{r_{n-2}}{2} \\ \frac{r_{n-2}}{2} & 1 - \frac{r_{n-2}}{2} \end{bmatrix}. \tag{28}$$

The detailed derivation is shown in Appendix B. Note that r_{n-2} is the coding rate over the $(n-2)^{th}$ wire and intuitively it can be viewed as a summarized statistic for the dependency of the coded bits above the $(n-2)^{th}$ wire. Now the Markov chain $\{c_{n-2}(t),\ t \geq 1\}$ here is an approximation of the original hidden Markov chain $\{c_{n-2}(t),\ t \geq 1\}$.

(S2) Use the conditional independence property in (P4) to derive the transition probability matrix $\mathbf{P}''_n = [(\mathbf{P}''_n)_{c_{n-1}c_{n-2},c'_{n-1}c'_{n-2}}]_{(c_{n-1},c_{n-2}),(c'_{n-1},c'_{n-2})\in\{0,1\}^2}$ for the Markov chain $\{(c_{n-1}(t), c_{n-2}(t)),\ t \geq 1\}$. This is given by

$$\mathbf{P}''_n = \begin{bmatrix} \frac{1}{2} - \frac{r_{n-2}}{4} & \frac{r_{n-2}}{4} & \frac{1}{2} - \frac{r_{n-2}}{4} & \frac{r_{n-2}}{4} \\ \frac{r_{n-2}}{4} + \frac{1-r_{n-1}}{r_{n-1}} & \frac{1}{2} - \frac{r_{n-2}}{4} & \frac{r_{n-2}}{4} - \frac{1-r_{n-1}}{r_{n-1}} & \frac{1}{2} - \frac{r_{n-2}}{4} \\ \frac{1}{2} - \frac{r_{n-2}}{4} & \frac{r_{n-2}}{4} - \frac{1-r_{n-1}}{r_{n-1}} & \frac{1}{2} - \frac{r_{n-2}}{4} & \frac{r_{n-2}}{4} + \frac{1-r_{n-1}}{r_{n-1}} \\ \frac{r_{n-2}}{4} & \frac{1}{2} - \frac{r_{n-2}}{4} & \frac{r_{n-2}}{4} & \frac{1}{2} - \frac{r_{n-2}}{4} \end{bmatrix}. \tag{29}$$

The detailed derivation is shown in Appendix C. Again, r_{n-1} is the coding rate over the $(n-1)^{th}$ wire and intuitively it can be viewed as a summarized statistic for the dependency of the coded bits above the $(n-1)^{th}$ wire. Now the Markov chain $\{(c_{n-1}(t), c_{n-2}(t)),\ t \geq 1\}$ here is an approximation of the original hidden Markov chain $\{(c_{n-1}(t), c_{n-2}(t)),\ t \geq 1\}$.

(S3) Use the conditional independence property in (P5) to derive the transition probability matrix

$$\mathbf{P}'''_n = [(\mathbf{P}'''_n)_{c_n c_{n-1} c_{n-2}, c'_n c'_{n-1} c'_{n-2}}]_{(c_n,c_{n-1},c_{n-2}),(c'_n,c'_{n-1},c'_{n-2})\in\{0,1\}^3}$$

for the Markov chain $\{(c_n(t), c_{n-1}(t), c_{n-2}(t)),\ t \geq 1\}$. This is given by

$$\mathbf{P}'''_n = \left[\begin{matrix} \frac{1}{4} - \frac{r_{n-2}}{8} & \frac{r_{n-2}}{8} & \frac{1}{4} - \frac{r_{n-2}}{8} & \frac{r_{n-2}}{8} \\ \frac{r_{n-2}}{8} + \frac{1-r_{n-1}}{2r_{n-1}} & \frac{1}{4} - \frac{r_{n-2}}{8} & \frac{r_{n-2}}{8} - \frac{1-r_{n-1}}{2r_{n-1}} & \frac{1}{4} - \frac{r_{n-2}}{8} \\ \frac{1}{4} - \frac{r_{n-2}}{8} & \frac{r_{n-2}}{4} - \frac{1-r_{n-1}}{r_{n-1}} & \frac{1}{4} - \frac{r_{n-2}}{8} & \frac{r_{n-2}}{8} + \frac{1-r_{n-1}}{2r_{n-1}} \\ \frac{r_{n-2}}{8} & \frac{1}{4} - \frac{r_{n-2}}{8} & \frac{r_{n-2}}{8} & \frac{1}{4} - \frac{r_{n-2}}{8} \\ \frac{1}{4} - \frac{r_{n-2}}{8} & \frac{r_{n-2}}{8} & \frac{1}{4} - \frac{r_{n-2}}{8} & \frac{r_{n-2}}{8} \\ \frac{r_{n-2}}{8} + \frac{1-r_{n-1}}{2r_{n-1}} & \frac{1}{4} - \frac{r_{n-2}}{8} & 0 & \frac{1}{4} - \frac{r_{n-2}}{8} \\ \frac{1}{4} - \frac{r_{n-2}}{8} & \frac{r_{n-2}}{8} - \frac{1-r_{n-1}}{2r_{n-1}} & \frac{1}{4} - \frac{r_{n-2}}{8} & \frac{r_{n-2}}{8} + \frac{1-r_{n-1}}{2r_{n-1}} \\ \frac{r_{n-2}}{8} & \frac{1}{4} - \frac{r_{n-2}}{8} & \frac{r_{n-2}}{8} & \frac{1}{4} - \frac{r_{n-2}}{8} \end{matrix}\right.$$

$$\left.\begin{matrix} \frac{1}{4}-\frac{r_{n-2}}{8} & \frac{r_{n-2}}{8} & \frac{1}{4}-\frac{r_{n-2}}{8} & \frac{r_{n-2}}{8} \\ \frac{r_{n-2}}{8}+\frac{1-r_{n-1}}{2r_{n-1}} & \frac{1}{4}-\frac{r_{n-2}}{8} & \frac{r_{n-2}}{8}-\frac{1-r_{n-1}}{2r_{n-1}} & \frac{1}{4}-\frac{r_{n-2}}{8} \\ \frac{1}{4}-\frac{r_{n-2}}{8} & 0 & \frac{1}{4}-\frac{r_{n-2}}{8} & \frac{r_{n-2}}{8}+\frac{1-r_{n-1}}{2r_{n-1}} \\ \frac{r_{n-2}}{8} & \frac{1}{4}-\frac{r_{n-2}}{8} & \frac{r_{n-2}}{8} & \frac{1}{4}-\frac{r_{n-2}}{8} \\ \frac{1}{4}-\frac{r_{n-2}}{8} & \frac{r_{n-2}}{8} & \frac{1}{4}-\frac{r_{n-2}}{8} & \frac{r_{n-2}}{8} \\ \frac{r_{n-2}}{8}+\frac{1-r_{n-1}}{2r_{n-1}} & \frac{1}{4}-\frac{r_{n-2}}{8} & \frac{r_{n-2}}{4}-\frac{1-r_{n-1}}{r_{n-1}} & \frac{1}{4}-\frac{r_{n-2}}{8} \\ \frac{1}{4}-\frac{r_{n-2}}{8} & \frac{r_{n-2}}{8}-\frac{1-r_{n-1}}{2r_{n-1}} & \frac{1}{4}-\frac{r_{n-2}}{8} & \frac{r_{n-2}}{8}+\frac{1-r_{n-1}}{2r_{n-1}} \\ \frac{r_{n-2}}{8} & \frac{1}{4}-\frac{r_{n-2}}{8} & \frac{r_{n-2}}{8} & \frac{1}{4}-\frac{r_{n-2}}{8} \end{matrix}\right] . \tag{30}$$

The detailed derivation is shown in Appendix D. Now the Markov chain $\{(c_n(t), c_{n-1}(t), c_{n-2}(t)),\ t \geq 1\}$ here is an approximation of the original hidden Markov chain $\{(c_n(t), c_{n-1}(t), c_{n-2}(t)),\ t \geq 1\}$.

(S4) Solve the steady state probabilities $\boldsymbol{\pi}'''_n = (\pi'''_{n,000}, \pi'''_{n,001}, \ldots, \pi'''_{n,111})$ of the Markov chain $\{(c_n(t), c_{n-1}(t), c_{n-2}(t)),\ t \geq 1\}$ from the system of linear equations $\boldsymbol{\pi}'''_n = \boldsymbol{\pi}'''_n \mathbf{P}'''_n$ and $\pi'''_{n,000} + \pi'''_{n,001} + \cdots + \pi'''_{n,111} = 1$. They are given by

$$\boldsymbol{\pi}'''_n = \left(\frac{1}{4} - \frac{r_{n-1}}{8}, \frac{r_{n-1}}{4} - \frac{r_{n-1}^2}{r_{n-1}r_{n-2} + 4(3r_{n-1}-1)}, \frac{r_{n-1}^2}{r_{n-1}r_{n-2} + 4(3r_{n-1}-1)}, \frac{1}{4} - \frac{r_{n-1}}{8}, \right.$$
$$\left. \frac{1}{4} - \frac{r_{n-1}}{8}, \frac{r_{n-1}^2}{r_{n-1}r_{n-2} + 4(3r_{n-1}-1)}, \frac{r_{n-1}}{4} - \frac{r_{n-1}^2}{r_{n-1}r_{n-2} + 4(3r_{n-1}-1)}, \frac{1}{4} - \frac{r_{n-1}}{8}\right). \tag{31}$$

(S5) Compute the coding rate r_n from the steady state probabilities of the Markov chain $\{(c_n(t), c_{n-1}(t), c_{n-2}(t)),\ t \geq 1\}$. This then leads to the following recursive equation for r_n:

$$r_n = \frac{r_{n-1}r_{n-2}\left(1 - \frac{r_{n-1}}{2}\right) + 4(3r_{n-1}-1) + 2r_{n-1}(1-r_{n-1})}{r_{n-1}r_{n-2} + 4(3r_{n-1}-1)}, \tag{32}$$

where $r_1 = 1$ and $r_2 = 1$. The detailed derivation is shown in Appendix E.

The approximation of r_n in (32) is shown in the last row of Table II. From Table II, we can see that for $n = 3$, the approximation in (32) is the same as the exact value of r_n in (27). This is no coincidence as we have mentioned earlier that the five properties (P1)-(P5) are true for $n = 3$. Furthermore, the approximation of r_n in (32) matches very well to the exact value of r_n in (27) for $4 \leq n \leq 10$. As $n \to \infty$, the coding rate $r_\infty = \lim_{n\to\infty} r_n$ can be obtained by solving

$$r_\infty = \frac{r_\infty^2\left(1 - \frac{r_\infty}{2}\right) + 4(3r_\infty - 1) + 2r_\infty(1-r_\infty)}{r_\infty^2 + 4(3r_\infty - 1)}$$

with the constraint that $2/3 \leq r_\infty \leq 1$. The result is $r_\infty \approx 0.9575$, which matches very well to the exact value of r_n in (27) for $6 \leq n \leq 10$. Moreover, we also perform simulations for n up to 3000. In our simulations, the input streams are all i.i.d. Bernoulli random variables with equal probabilities of being 0 and 1. The total number coded bits in each wire is 10001000, i.e., $t = 1, 2, \ldots, 10001000$. Thus, the total number of coded bits on the 3000 wires is approximately 3×10^{10}. We run our simulations on a server with an Intel Xeon processor (3.00Ghz) and the Ubuntu Linux operating system. The total simulation time is 17 days, 3 hours and 7 mins. To obtain the 95% confidence interval, we discard the simulation results for the first 1000 coded bits (to reduce the transient effect) and then apply the standard batch means method [Alexopoulos and Seila 1996] by viewing the average coding rate for each block of 1000 coded bits as an "independent" observation. We then compute the sample mean and the standard deviation of these "independent" observations and use those to compute the confidence interval. In Table VIII, we show the simulation results for the coding rate per wire R_n (see the fourth row of Table II) for $3 \leq n \leq 10$. The simulation results match the theoretical results (that are obtained from solving the

Table VIII. The simulation results for the coding rate per wire R_n, $3 \leq n \leq 10$.

n	3	4	5	6	7	8	9	10
R_n (theory)	0.9815	0.9759	0.9723	0.9698	0.9681	0.9668	0.9657	0.9649
Sample mean	0.9815	0.9759	0.9723	0.9698	0.9681	0.9668	0.9658	0.9649
Standard deviation	0.0023	0.0022	0.0021	0.0020	0.0019	0.0018	0.0017	0.0016

Markov chain numerically). The coding rates for $2000 \leq n \leq 3000$ are all 0.9576 with 95% confidence intervals smaller than 0.0002. The simulation results also show that our approximation $r_\infty \approx 0.9575$ is very good.

7. CONCLUSION

In this paper, we analyzed the coding rates of for various FOCs, including the maximum achievable rate, the memoryless FOCs, and the FOCs generated from the bit stuffing algorithm. For the coding rate of the bit-stuffing algorithm, we performed a worst-case analysis and a probabilistic analysis (under the assumption of Bernoulli inputs). To cope with the problem of the curse of dimensionality, we proposed an approximate probabilistic analysis that limits the dependency of coded bits in adjacent wires. In comparison with the approximate analysis for FTCs in [Chang et al. 2015], our approximate analysis is much more difficult as it adds another level of dependency among coded bits. Also, our numerical results obtained in this paper could be useful for further understanding the tradeoff between maximum delay and coding rate in the designs of DSM buses. In particular, our numerical results show that the coding rates for various FOCs are significantly higher than their counterparts for FTCs in [Chang et al. 2015].

One possible extension of the bit-stuffing algorithm is to use parallel encoding/decoding as in [Chang et al. 2015]. Consider a bus with n parallel wires and n data bit streams. Index the n wires from 1 to n and let $S = \{3, 6, 9, \ldots, 3\lfloor n/3 \rfloor\}$. For a bus $i \notin S$, we simply set $c_i(t)$ to be the next data bit of the i^{th} data bit stream. On the other hand, for a bus $i \in S$, we set $c_i(t) = c_i(t-1)$ as a stuffed bit if one of the following three bit-stuffing conditions holds: (i) $c_{i-2}(t-1) = \bar{c}_{i-2}(t) = \bar{c}_{i-1}(t-1) = c_{i-1}(t) = c_i(t-1)$, (ii) $\bar{c}_{i-1}(t-1) = c_{i-1}(t) = c_i(t-1) = \bar{c}_{i+1}(t-1) = c_{i+1}(t)$, (iii) $c_{i+2}(t-1) = \bar{c}_{i+2}(t) = \bar{c}_{i+1}(t-1) = c_{i+1}(t) = c_i(t-1)$. Otherwise, we also set $c_i(t)$ to be the next data bit of the i^{th} data bit stream.

The coding rate of a wire $i \notin S$ is certainly 1. To compute the coding rate of a wire $i \in S$, let $\hat{c}(t) = (c_{i-2}(t), c_{i-1}(t), c_i(t), c_{i+1}(t), c_{i+2}(t))$. Under the assumption that all the n data bit streams consist of i.i.d. Bernoulli random variables with equal probabilities of being 0 or 1, the stochastic process $\hat{c}(t)$ is also a time-homogeneous Markov chain. Solving such a Markov chain with 32 states yields the coding rate 0.8634 for a wire in S. Thus, when n is large, the coding rate per wire is $(1 + 1 + 0.8634)/3 \approx 0.9544$ for the parallel bit-stuffing algorithm.

A. PROOF OF LEMMA 2.1

As the relation that there are no "$010 \to 101$" and "$101 \to 010$" transitions between three adjacent wires is symmetric, it is clear from the definition of the adjacency matrix $\mathbf{A}_n$ associated with the forbidden overlap channel that it is a symmetric matrix for all $n \geq 1$.

For $n = 1$, the two possible states are 0 and 1, and it is clear that

$$\mathbf{A}_1 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}. \tag{33}$$

It follows from (33) and $\mathbf{A}_0 = \mathbf{B}_0 = 1$ that (8) holds for $n = 1$.

For $n = 2$, the four possible states are $(0, 0), (0, 1), (1, 0)$, and $(1, 1)$, and it is easy to see that

$$\mathbf{A}_2 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{B}_1 \\ \mathbf{B}_1^T & \mathbf{A}_1 \end{bmatrix}, \text{ where } \mathbf{B}_1 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}. \tag{34}$$

It follows from (34) and $\mathbf{C}_0 = 1$ that (8) holds for $n = 2$.

For $n = 3$, the eight possible states are $(0, 0, 0), (0, 0, 1), (0, 1, 0), (0, 1, 1), (1, 0, 0), (1, 0, 1), (1, 1, 0)$, $(1, 1, 1)$, and it is clear that

$$\mathbf{A}_3 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{A}_2 & \mathbf{B}_2 \\ \mathbf{B}_2^T & \mathbf{A}_2 \end{bmatrix}, \tag{35}$$

where

$$\mathbf{B}_2 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{B}_1 \\ \mathbf{C}_1 & \mathbf{A}_1 \end{bmatrix}, \tag{36}$$

in which

$$\mathbf{C}_1 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{A}_0 & \mathbf{O}_0 \\ \mathbf{B}_0^T & \mathbf{A}_0 \end{bmatrix}. \tag{37}$$

It follows from (35)–(37) that (8) holds for $n = 3$.

Assume as the induction hypothesis that (8) holds for some $n - 1 \geq 3$ and $n - 2 \geq 2$, i.e.,

$$\mathbf{A}_{n-1} = \begin{bmatrix} \mathbf{A}_{n-2} & \mathbf{B}_{n-2} \\ \mathbf{B}_{n-2}^T & \mathbf{A}_{n-2} \end{bmatrix} \tag{38}$$

and

$$\mathbf{A}_{n-2} = \begin{bmatrix} \mathbf{A}_{n-3} & \mathbf{B}_{n-3} \\ \mathbf{B}_{n-3}^T & \mathbf{A}_{n-3} \end{bmatrix}. \tag{39}$$

To ease the presentation in the rest of the proof, we denote

$$\begin{aligned} \mathbf{c}^{(n-1)} &= (c_{n-1}, c_{n-2}, \ldots, c_1), \\ \mathbf{c}^{(n-2)} &= (c_{n-2}, c_{n-3}, \ldots, c_1), \\ \mathbf{c}^{(n-3)} &= (c_{n-3}, c_{n-4}, \ldots, c_1). \end{aligned}$$

Also, we denote

$$\begin{aligned} c_n\mathbf{c}^{(n-1)} &= (c_n, c_{n-1}, \ldots, c_1), \\ c_n c_{n-1}\mathbf{c}^{(n-2)} &= (c_n, c_{n-1}, \ldots, c_1), \\ c_n c_{n-1} c_{n-2}\mathbf{c}^{(n-3)} &= (c_n, c_{n-1}, c_{n-2}, \ldots, c_1). \end{aligned}$$

For $\mathbf{c}^{(n-3)} = (c_{n-3}, c_{n-4}, \ldots, c_1) \in \{0,1\}^{n-3}$ and $\mathbf{c}'^{(n-3)} = (c'_{n-3}, c'_{n-4}, \ldots, c'_1) \in \{0,1\}^{n-3}$, it is easy to see from the definition of the adjacency matrix $\mathbf{A}_n$ and (38)–(39) that

$$
\begin{aligned}
&(\mathbf{A}_n)_{000\mathbf{c}^{(n-3)},000\mathbf{c}'^{(n-3)}} = (\mathbf{A}_n)_{000\mathbf{c}^{(n-3)},010\mathbf{c}'^{(n-3)}} = (\mathbf{A}_n)_{000\mathbf{c}^{(n-3)},100\mathbf{c}'^{(n-3)}} \\
&= (\mathbf{A}_n)_{000\mathbf{c}^{(n-3)},110\mathbf{c}'^{(n-3)}} = (\mathbf{A}_n)_{001\mathbf{c}^{(n-3)},001\mathbf{c}'^{(n-3)}} = (\mathbf{A}_n)_{001\mathbf{c}^{(n-3)},011\mathbf{c}'^{(n-3)}} \\
&= (\mathbf{A}_n)_{001\mathbf{c}^{(n-3)},101\mathbf{c}'^{(n-3)}} = (\mathbf{A}_n)_{001\mathbf{c}^{(n-3)},111\mathbf{c}'^{(n-3)}} = (\mathbf{A}_n)_{010\mathbf{c}^{(n-3)},000\mathbf{c}'^{(n-3)}} \\
&= (\mathbf{A}_n)_{010\mathbf{c}^{(n-3)},010\mathbf{c}'^{(n-3)}} = (\mathbf{A}_n)_{010\mathbf{c}^{(n-3)},100\mathbf{c}'^{(n-3)}} = (\mathbf{A}_n)_{010\mathbf{c}^{(n-3)},110\mathbf{c}'^{(n-3)}} \\
&= (\mathbf{A}_n)_{011\mathbf{c}^{(n-3)},001\mathbf{c}'^{(n-3)}} = (\mathbf{A}_n)_{011\mathbf{c}^{(n-3)},011\mathbf{c}'^{(n-3)}} = (\mathbf{A}_n)_{011\mathbf{c}^{(n-3)},101\mathbf{c}'^{(n-3)}} \\
&= (\mathbf{A}_n)_{011\mathbf{c}^{(n-3)},111\mathbf{c}'^{(n-3)}} = (\mathbf{A}_n)_{100\mathbf{c}^{(n-3)},000\mathbf{c}'^{(n-3)}} = (\mathbf{A}_n)_{100\mathbf{c}^{(n-3)},010\mathbf{c}'^{(n-3)}} \\
&= (\mathbf{A}_n)_{100\mathbf{c}^{(n-3)},100\mathbf{c}'^{(n-3)}} = (\mathbf{A}_n)_{100\mathbf{c}^{(n-3)},110\mathbf{c}'^{(n-3)}} = (\mathbf{A}_n)_{101\mathbf{c}^{(n-3)},001\mathbf{c}'^{(n-3)}} \\
&= (\mathbf{A}_n)_{101\mathbf{c}^{(n-3)},011\mathbf{c}'^{(n-3)}} = (\mathbf{A}_n)_{101\mathbf{c}^{(n-3)},101\mathbf{c}'^{(n-3)}} = (\mathbf{A}_n)_{101\mathbf{c}^{(n-3)},111\mathbf{c}'^{(n-3)}} \\
&= (\mathbf{A}_n)_{110\mathbf{c}^{(n-3)},000\mathbf{c}'^{(n-3)}} = (\mathbf{A}_n)_{110\mathbf{c}^{(n-3)},010\mathbf{c}'^{(n-3)}} = (\mathbf{A}_n)_{110\mathbf{c}^{(n-3)},100\mathbf{c}'^{(n-3)}} \\
&= (\mathbf{A}_n)_{110\mathbf{c}^{(n-3)},110\mathbf{c}'^{(n-3)}} = (\mathbf{A}_n)_{111\mathbf{c}^{(n-3)},001\mathbf{c}'^{(n-3)}} = (\mathbf{A}_n)_{111\mathbf{c}^{(n-3)},011\mathbf{c}'^{(n-3)}} \\
&= (\mathbf{A}_n)_{111\mathbf{c}^{(n-3)},101\mathbf{c}'^{(n-3)}} = (\mathbf{A}_n)_{111\mathbf{c}^{(n-3)},111\mathbf{c}'^{(n-3)}} \\
&= (\mathbf{A}_{n-3})_{\mathbf{c}^{(n-3)},\mathbf{c}'^{(n-3)}}, && (40) \\
&(\mathbf{A}_n)_{000\mathbf{c}^{(n-3)},001\mathbf{c}'^{(n-3)}} = (\mathbf{A}_n)_{000\mathbf{c}^{(n-3)},011\mathbf{c}'^{(n-3)}} = (\mathbf{A}_n)_{000\mathbf{c}^{(n-3)},101\mathbf{c}'^{(n-3)}} \\
&= (\mathbf{A}_n)_{000\mathbf{c}^{(n-3)},111\mathbf{c}'^{(n-3)}} = (\mathbf{A}_n)_{010\mathbf{c}^{(n-3)},011\mathbf{c}'^{(n-3)}} = (\mathbf{A}_n)_{010\mathbf{c}^{(n-3)},111\mathbf{c}'^{(n-3)}} \\
&= (\mathbf{A}_n)_{100\mathbf{c}^{(n-3)},001\mathbf{c}'^{(n-3)}} = (\mathbf{A}_n)_{100\mathbf{c}^{(n-3)},011\mathbf{c}'^{(n-3)}} = (\mathbf{A}_n)_{100\mathbf{c}^{(n-3)},101\mathbf{c}'^{(n-3)}} \\
&= (\mathbf{A}_n)_{100\mathbf{c}^{(n-3)},111\mathbf{c}'^{(n-3)}} = (\mathbf{A}_n)_{110\mathbf{c}^{(n-3)},011\mathbf{c}'^{(n-3)}} = (\mathbf{A}_n)_{110\mathbf{c}^{(n-3)},111\mathbf{c}'^{(n-3)}} \\
&= (\mathbf{A}_{n-2})_{0\mathbf{c}^{(n-3)},1\mathbf{c}'^{(n-3)}} = (\mathbf{B}_{n-3})_{\mathbf{c}^{(n-3)},\mathbf{c}'^{(n-3)}}, && (41) \\
&(\mathbf{A}_n)_{001\mathbf{c}^{(n-3)},000\mathbf{c}'^{(n-3)}} = (\mathbf{A}_n)_{001\mathbf{c}^{(n-3)},100\mathbf{c}'^{(n-3)}} = (\mathbf{A}_n)_{011\mathbf{c}^{(n-3)},000\mathbf{c}'^{(n-3)}} \\
&= (\mathbf{A}_n)_{011\mathbf{c}^{(n-3)},010\mathbf{c}'^{(n-3)}} = (\mathbf{A}_n)_{011\mathbf{c}^{(n-3)},100\mathbf{c}'^{(n-3)}} = (\mathbf{A}_n)_{011\mathbf{c}^{(n-3)},110\mathbf{c}'^{(n-3)}} \\
&= (\mathbf{A}_n)_{101\mathbf{c}^{(n-3)},000\mathbf{c}'^{(n-3)}} = (\mathbf{A}_n)_{101\mathbf{c}^{(n-3)},100\mathbf{c}'^{(n-3)}} = (\mathbf{A}_n)_{111\mathbf{c}^{(n-3)},000\mathbf{c}'^{(n-3)}} \\
&= (\mathbf{A}_n)_{111\mathbf{c}^{(n-3)},010\mathbf{c}'^{(n-3)}} = (\mathbf{A}_n)_{111\mathbf{c}^{(n-3)},100\mathbf{c}'^{(n-3)}} = (\mathbf{A}_n)_{111\mathbf{c}^{(n-3)},110\mathbf{c}'^{(n-3)}} \\
&= (\mathbf{A}_{n-2})_{1\mathbf{c}^{(n-3)},0\mathbf{c}'^{(n-3)}} = (\mathbf{B}^T_{n-3})_{\mathbf{c}^{(n-3)},\mathbf{c}'^{(n-3)}}, && (42) \\
&(\mathbf{A}_n)_{001\mathbf{c}^{(n-3)},010\mathbf{c}'^{(n-3)}} = (\mathbf{A}_n)_{001\mathbf{c}^{(n-3)},110\mathbf{c}'^{(n-3)}} = (\mathbf{A}_n)_{101\mathbf{c}^{(n-3)},110\mathbf{c}'^{(n-3)}} \\
&= (\mathbf{A}_{n-1})_{01\mathbf{c}^{(n-3)},10\mathbf{c}'^{(n-3)}} = (\mathbf{B}_{n-2})_{1\mathbf{c}^{(n-3)},0\mathbf{c}'^{(n-3)}} = (\mathbf{C}_{n-3})_{\mathbf{c}^{(n-3)},\mathbf{c}'^{(n-3)}}, && (43) \\
&(\mathbf{A}_n)_{010\mathbf{c}^{(n-3)},001\mathbf{c}'^{(n-3)}} = (\mathbf{A}_n)_{110\mathbf{c}^{(n-3)},001\mathbf{c}'^{(n-3)}} = (\mathbf{A}_n)_{110\mathbf{c}^{(n-3)},101\mathbf{c}'^{(n-3)}} \\
&= (\mathbf{A}_{n-1})_{10\mathbf{c}^{(n-3)},01\mathbf{c}'^{(n-3)}} = (\mathbf{B}^T_{n-2})_{0\mathbf{c}^{(n-3)},1\mathbf{c}'^{(n-3)}} = (\mathbf{C}^T_{n-3})_{\mathbf{c}^{(n-3)},\mathbf{c}'^{(n-3)}}, && (44) \\
&(\mathbf{A}_n)_{010\mathbf{c}^{(n-3)},101\mathbf{c}'^{(n-3)}} = (\mathbf{A}_n)_{101\mathbf{c}^{(n-3)},010\mathbf{c}'^{(n-3)}} = 0. && (45)
\end{aligned}
$$

Therefore, we have from (40)–(45), (39), $\mathbf{A}^T_{n-3} = \mathbf{A}_{n-3}$, (38), and $\mathbf{A}^T_{n-2} = \mathbf{A}_{n-2}$ that

$$
\mathbf{A}_n = \begin{bmatrix}
\mathbf{A}_{n-3} & \mathbf{B}_{n-3} & \mathbf{A}_{n-3} & \mathbf{B}_{n-3} & \mathbf{A}_{n-3} & \mathbf{B}_{n-3} & \mathbf{A}_{n-3} & \mathbf{B}_{n-3} \\
\mathbf{B}^T_{n-3} & \mathbf{A}_{n-3} & \mathbf{C}_{n-3} & \mathbf{A}_{n-3} & \mathbf{B}^T_{n-3} & \mathbf{A}_{n-3} & \mathbf{C}_{n-3} & \mathbf{A}_{n-3} \\
\mathbf{A}_{n-3} & \mathbf{C}^T_{n-3} & \mathbf{A}_{n-3} & \mathbf{B}_{n-3} & \mathbf{A}_{n-3} & \mathbf{O}_{n-3} & \mathbf{A}_{n-3} & \mathbf{B}_{n-3} \\
\mathbf{B}^T_{n-3} & \mathbf{A}_{n-3} & \mathbf{B}^T_{n-3} & \mathbf{A}_{n-3} & \mathbf{B}^T_{n-3} & \mathbf{A}_{n-3} & \mathbf{B}^T_{n-3} & \mathbf{A}_{n-3} \\
\mathbf{A}_{n-3} & \mathbf{B}_{n-3} & \mathbf{A}_{n-3} & \mathbf{B}_{n-3} & \mathbf{A}_{n-3} & \mathbf{B}_{n-3} & \mathbf{A}_{n-3} & \mathbf{B}_{n-3} \\
\mathbf{B}^T_{n-3} & \mathbf{A}_{n-3} & \mathbf{O}_{n-3} & \mathbf{A}_{n-3} & \mathbf{B}^T_{n-3} & \mathbf{A}_{n-3} & \mathbf{C}_{n-3} & \mathbf{A}_{n-3} \\
\mathbf{A}_{n-3} & \mathbf{C}^T_{n-3} & \mathbf{A}_{n-3} & \mathbf{B}_{n-3} & \mathbf{A}_{n-3} & \mathbf{C}^T_{n-3} & \mathbf{A}_{n-3} & \mathbf{B}_{n-3} \\
\mathbf{B}^T_{n-3} & \mathbf{A}_{n-3} & \mathbf{B}^T_{n-3} & \mathbf{A}_{n-3} & \mathbf{B}^T_{n-3} & \mathbf{A}_{n-3} & \mathbf{B}^T_{n-3} & \mathbf{A}_{n-3}
\end{bmatrix}
$$

$$= \begin{bmatrix} \mathbf{A}_{n-2} & \mathbf{B}_{n-2} & \mathbf{A}_{n-2} & \mathbf{B}_{n-2} \\ \mathbf{B}_{n-2}^T & \mathbf{A}_{n-2} & \mathbf{C}_{n-2} & \mathbf{A}_{n-2} \\ \mathbf{A}_{n-2} & \mathbf{C}_{n-2}^T & \mathbf{A}_{n-2} & \mathbf{B}_{n-2} \\ \mathbf{B}_{n-2}^T & \mathbf{A}_{n-2} & \mathbf{B}_{n-2}^T & \mathbf{A}_{n-2} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{n-1} & \mathbf{B}_{n-1} \\ \mathbf{B}_{n-1}^T & \mathbf{A}_{n-1} \end{bmatrix},$$

where

$$\mathbf{B}_{n-1} = \begin{bmatrix} \mathbf{A}_{n-2} & \mathbf{B}_{n-2} \\ \mathbf{C}_{n-2} & \mathbf{A}_{n-2} \end{bmatrix}, \text{ in which } \mathbf{C}_{n-2} = \begin{bmatrix} \mathbf{A}_{n-3} & \mathbf{O}_{n-3} \\ \mathbf{B}_{n-3}^T & \mathbf{A}_{n-3} \end{bmatrix}.$$

The induction is completed.

B. THE TRANSITION PROBABILITY MATRIX FOR THE MARKOV CHAIN IN (S1)

Let $D_{n-2}(t)$ (resp., $S_{n-2}(t)$) be the event that $c_{n-2}(t)$ is a data (resp., stuffed) bit. Then we have

$$\lim_{t\to\infty} P(D_{n-2}(t)) = r_{n-2}, \tag{46}$$

$$\lim_{t\to\infty} P(S_{n-2}(t)) = \lim_{t\to\infty}(1 - P(D_{n-2}(t))) = 1 - r_{n-2}. \tag{47}$$

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

$$\lim_{t\to\infty} P(c_{n-2}(t) = 0) = \lim_{t\to\infty} P(c_{n-2}(t) = 1) = \frac{1}{2}, \tag{48}$$

$$\lim_{t\to\infty} P(c_{n-2}(t-1) = c_{n-2}(t) = 0) = \lim_{t\to\infty} P(c_{n-2}(t-1) = c_{n-2}(t) = 1). \tag{49}$$

Note from the law of total probability that

$$\begin{aligned} &P(c_{n-2}(t) = c_{n-2}(t-1)) \\ &= P(D_{n-2}(t))P(c_{n-2}(t) = c_{n-2}(t-1)|D_{n-2}(t)) \\ &\quad + P(S_{n-2}(t))P(c_{n-2}(t) = c_{n-2}(t-1)|S_{n-2}(t)). \end{aligned} \tag{50}$$

Given that $c_{n-2}(t)$ is a stuffed bit, we know from the bit-stuffing rule that $c_{n-2}(t) = c_{n-2}(t-1)$, and so we have

$$P(c_{n-2}(t) = c_{n-2}(t-1)|S_{n-2}(t)) = 1. \tag{51}$$

Given that $c_{n-2}(t)$ is a data bit, $c_{n-2}(t)$ is a Bernoulli random variable with equal probabilities of being 0 or 1, and is conditionally independent of $c_{n-2}(t-1)$, and thus we have

$$\begin{aligned} &P(c_{n-2}(t) = c_{n-2}(t-1)|D_{n-2}(t)) \\ &= \sum_{c_{n-2}\in\{0,1\}} P(c_{n-2}(t-1) = c_{n-2}|D_{n-2}(t)) \\ &\qquad \times P(c_{n-2}(t) = c_{n-2}|D_{n-2}(t), c_{n-2}(t-1) = c_{n-2}) \\ &= \sum_{c_{n-2}\in\{0,1\}} \frac{1}{2} P(c_{n-2}(t-1) = c_{n-2}) = \frac{1}{2}. \end{aligned} \tag{52}$$

As such, it follows from (50)–(52) and (46)–(47) that

$$\begin{aligned} &\lim_{t\to\infty} P(c_{n-2}(t) = c_{n-2}(t-1)) \\ &= \frac{1}{2}\cdot r_{n-2} + 1\cdot(1 - r_{n-2}) = 1 - \frac{r_{n-2}}{2}. \end{aligned} \tag{53}$$

Therefore, we have from (48), (49), and (53) that

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

By symmetry, we have $(\mathbf{P}'_n)_{1,1} = 1-\frac{r_{n-2}}{2}$. Finally, $(\mathbf{P}'_n)_{0,1} = 1-(\mathbf{P}'_n)_{0,0} = \frac{r_{n-2}}{2}$ and $(\mathbf{P}'_n)_{1,0} = 1-(\mathbf{P}'_n)_{1,1} = \frac{r_{n-2}}{2}$.

C. THE TRANSITION PROBABILITY MATRIX FOR THE MARKOV CHAIN IN (S2)

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

$$\lim_{t\to\infty} P(D_{n-1}(t-1)) = \lim_{t\to\infty} P(D_{n-1}(t)) = r_{n-1}, \tag{54}$$

$$\lim_{t\to\infty} P(S_{n-1}(t-1)) = \lim_{t\to\infty} P(S_{n-1}(t)) = \lim_{t\to\infty} (1-P(D_{n-1}(t))) = 1-r_{n-1}. \tag{55}$$

Analogous to the arguments in Appendix B, we have

$$\lim_{t\to\infty} P(c_{n-1}(t-1)=c_{n-2}(t-1)) = 1-\frac{r_{n-1}}{2}. \tag{56}$$

Also,

$$\lim_{t\to\infty} P(c_{n-1}(t-1)=0|c_{n-2}(t-1)=0) = 1-\frac{r_{n-1}}{2}. \tag{57}$$

$$\lim_{t\to\infty} P(c_{n-1}(t-1)=1|c_{n-2}(t-1)=1) = 1-\frac{r_{n-1}}{2}. \tag{58}$$

$$\lim_{t\to\infty} P(c_{n-1}(t-1)=1|c_{n-2}(t-1)=0) = \frac{r_{n-1}}{2}. \tag{59}$$

$$\lim_{t\to\infty} P(c_{n-1}(t-1)=0|c_{n-2}(t-1)=1) = \frac{r_{n-1}}{2}. \tag{60}$$

From the conditional independence property in (P4), we see that

$$\begin{aligned}&P(c_{n-1}(t-1)=c_{n-2}(t-1)=c_{n-2}(t))\\ &= \sum_{c_{n-2}\in\{0,1\}} P(c_{n-2}(t-1)=c_{n-2})P(c_{n-1}(t-1)=c_{n-2}, c_{n-2}(t)=c_{n-2}|c_{n-2}(t-1)=c_{n-2})\\ &= \sum_{c_{n-2}\in\{0,1\}} P(c_{n-2}(t-1)=c_{n-2})P(c_{n-1}(t-1)=c_{n-2}|c_{n-2}(t-1)=c_{n-2})\\ &\qquad \times P(c_{n-2}(t)=c_{n-2}|c_{n-2}(t-1)=c_{n-2}).\end{aligned} \tag{61}$$

As such, we have from (61), (57)–(58), and (28) that

$$\lim_{t\to\infty} P(c_{n-1}(t-1)=c_{n-2}(t-1)=c_{n-2}(t)) = \left(1-\frac{r_{n-1}}{2}\right)\left(1-\frac{r_{n-2}}{2}\right). \tag{62}$$

Similarly, we have

$$\lim_{t\to\infty} P(c_{n-1}(t-1)=c_{n-2}(t-1)=\bar{c}_{n-2}(t)) = \left(1-\frac{r_{n-1}}{2}\right)\frac{r_{n-2}}{2}, \tag{63}$$

$$\lim_{t\to\infty} P(c_{n-1}(t-1)=\bar{c}_{n-2}(t-1)=c_{n-2}(t)) = \frac{r_{n-1}r_{n-2}}{4}, \tag{64}$$
$$\lim_{t\to\infty} P(c_{n-1}(t-1)=\bar{c}_{n-2}(t-1)=\bar{c}_{n-2}(t)) = \frac{r_{n-1}}{2}\left(1-\frac{r_{n-2}}{2}\right). \tag{65}$$

Note that

$$\begin{aligned} &P(c_{n-1}(t-1)=c_{n-2}(t-1)=c_{n-1}(t)=c_{n-2}(t)) \\ &= P(D_{n-1}(t))P(c_{n-1}(t-1)=c_{n-2}(t-1)=c_{n-1}(t)=c_{n-2}(t)|D_{n-1}(t)) \\ &\quad +P(S_{n-1}(t))P(c_{n-1}(t-1)=c_{n-2}(t-1)=c_{n-1}(t)=c_{n-2}(t)|S_{n-1}(t)). \end{aligned} \tag{66}$$

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 $c_{n-1}(t-1)$, $c_{n-2}(t-1)$, and $c_{n-2}(t)$, and thus we have

$$\begin{aligned} &P(c_{n-1}(t-1)=c_{n-2}(t-1)=c_{n-1}(t)=c_{n-2}(t)|D_{n-1}(t)) \\ &= \sum_{c'_{n-1}\in\{0,1\}} P(c_{n-1}(t-1)=c_{n-2}(t-1)=c_{n-2}(t)=c'_{n-1}|D_{n-1}(t)) \\ &\qquad \times P(c_{n-1}(t)=c'_{n-1}|D_{n-1}(t), c_{n-1}(t-1)=c_{n-2}(t-1)=c_{n-2}(t)=c'_{n-1}) \\ &= \frac{1}{2}P(c_{n-1}(t-1)=c_{n-2}(t-1)=c_{n-2}(t)|D_{n-1}(t)). \end{aligned} \tag{67}$$

Given that $c_{n-1}(t)$ is a stuffed bit, we know from the bit-stuffing rule of our bit-stuffing algorithm that $c_{n-1}(t)=c_{n-1}(t-1)=c_{n-2}(t)=\bar{c}_{n-2}(t-1)$, and thus we have

$$P(c_{n-1}(t-1)=c_{n-2}(t-1)=c_{n-1}(t)=c_{n-2}(t)|S_{n-1}(t))=0. \tag{68}$$

It follows from (66)–(68), the bit-stuffing rule of our bit-stuffing algorithm, and (62) that

$$\begin{aligned} &\lim_{t\to\infty} P(c_{n-1}(t-1)=c_{n-2}(t-1)=c_{n-1}(t)=c_{n-2}(t)) \\ &= \lim_{t\to\infty} \frac{1}{2}P(c_{n-1}(t-1)=c_{n-2}(t-1)=c_{n-2}(t), D_{n-1}(t)) \\ &= \lim_{t\to\infty} \frac{1}{2}P(c_{n-1}(t-1)=c_{n-2}(t-1)=c_{n-2}(t)) \\ &= \frac{1}{2}\left(1-\frac{r_{n-1}}{2}\right)\left(1-\frac{r_{n-2}}{2}\right). \end{aligned} \tag{69}$$

By similar arguments, we can obtain

$$\begin{aligned} &\lim_{t\to\infty} P(c_{n-1}(t-1)=c_{n-2}(t-1)=\bar{c}_{n-1}(t)=c_{n-2}(t)) \\ &= \lim_{t\to\infty} P(c_{n-1}(t-1)=c_{n-2}(t-1)=c_{n-1}(t)=c_{n-2}(t)) \\ &= \frac{1}{2}\left(1-\frac{r_{n-1}}{2}\right)\left(1-\frac{r_{n-2}}{2}\right), \end{aligned} \tag{70}$$
$$\begin{aligned} &\lim_{t\to\infty} P(c_{n-1}(t-1)=c_{n-2}(t-1)=\bar{c}_{n-1}(t)=\bar{c}_{n-2}(t)) \\ &= \lim_{t\to\infty} P(c_{n-1}(t-1)=c_{n-2}(t-1)=c_{n-1}(t)=\bar{c}_{n-2}(t)) \\ &= \left(1-\frac{r_{n-1}}{2}\right)\frac{r_{n-2}}{4}, \end{aligned} \tag{71}$$
$$\begin{aligned} &\lim_{t\to\infty} P(c_{n-1}(t-1)=\bar{c}_{n-2}(t-1)=\bar{c}_{n-1}(t)=\bar{c}_{n-2}(t)) \\ &= \lim_{t\to\infty} P(c_{n-1}(t-1)=\bar{c}_{n-2}(t-1)=c_{n-1}(t)=\bar{c}_{n-2}(t)) \end{aligned}$$

$$
\begin{aligned}
&= \frac{r_{n-1}}{4}\left(1-\frac{r_{n-2}}{2}\right), \qquad (72)\\
&\lim_{t\to\infty} P(c_{n-1}(t-1)=\bar{c}_{n-2}(t-1)=\bar{c}_{n-1}(t)=c_{n-2}(t))\\
&= \lim_{t\to\infty}\frac{1}{2}P(c_{n-1}(t-1)=\bar{c}_{n-2}(t-1)=c_{n-2}(t),D_{n-1}(t)),\\
&\lim_{t\to\infty} P(c_{n-1}(t-1)=\bar{c}_{n-2}(t-1)=c_{n-1}(t)=c_{n-2}(t))\\
&= \lim_{t\to\infty}\frac{1}{2}P(c_{n-1}(t-1)=\bar{c}_{n-2}(t-1)=c_{n-2}(t),D_{n-1}(t))+(1-r_{n-1}).
\end{aligned}
$$

Since we have

$$
\begin{aligned}
&P(c_{n-1}(t-1)=c_{n-2}(t-1)=\bar{c}_{n-1}(t)=c_{n-2}(t))\\
&+P(c_{n-1}(t-1)=c_{n-2}(t-1)=c_{n-1}(t)=c_{n-2}(t))\\
&+P(c_{n-1}(t-1)=c_{n-2}(t-1)=\bar{c}_{n-1}(t)=\bar{c}_{n-2}(t))\\
&+P(c_{n-1}(t-1)=c_{n-2}(t-1)=c_{n-1}(t)=\bar{c}_{n-2}(t))\\
&+P(c_{n-1}(t-1)=\bar{c}_{n-2}(t-1)=\bar{c}_{n-1}(t)=\bar{c}_{n-2}(t))\\
&+P(c_{n-1}(t-1)=\bar{c}_{n-2}(t-1)=c_{n-1}(t)=\bar{c}_{n-2}(t))\\
&+P(c_{n-1}(t-1)=\bar{c}_{n-2}(t-1)=\bar{c}_{n-1}(t)=c_{n-2}(t))\\
&+P(c_{n-1}(t-1)=\bar{c}_{n-2}(t-1)=c_{n-1}(t)=c_{n-2}(t))=1,
\end{aligned}
$$

it follows that

$$
\lim_{t\to\infty} P(c_{n-1}(t-1)=\bar{c}_{n-2}(t-1)=c_{n-2}(t),D_{n-1}(t))=\frac{r_{n-1}r_{n-2}}{4}-(1-r_{n-1}).
$$

As such, we have

$$
\lim_{t\to\infty} P(c_{n-1}(t-1)=\bar{c}_{n-2}(t-1)=\bar{c}_{n-1}(t)=c_{n-2}(t))=\frac{r_{n-1}r_{n-2}}{8}-\frac{1}{2}(1-r_{n-1}), \qquad (73)
$$

$$
\lim_{t\to\infty} P(c_{n-1}(t-1)=\bar{c}_{n-2}(t-1)=c_{n-1}(t)=c_{n-2}(t))=\frac{r_{n-1}r_{n-2}}{8}-\frac{1}{2}(1-r_{n-1}). \qquad (74)
$$

Finally, it is easy to see that (29) follows from the symmetry of the bit-stuffing algorithm, (70)–(74), and (56).

D. THE TRANSITION PROBABILITY MATRIX FOR THE MARKOV CHAIN IN (S3)

From the chain rule and the conditional independence property in (P5), we immediately see that

$$
\begin{aligned}
&(\mathbf{P}'''_n)_{c_nc_{n-1}c_{n-2},c'_nc'_{n-1}c'_{n-2}}\\
=&\lim_{t\to\infty} P(c_n(t)=c'_n,c_{n-1}(t)=c'_{n-1},c_{n-2}(t)=c'_{n-2}\\
&\qquad |c_n(t-1)=c_n,c_{n-1}(t-1)=c_{n-1},c_{n-2}(t-1)=c_{n-2})\\
=&\lim_{t\to\infty} P(c_{n-1}(t)=c'_{n-1},c_{n-2}(t)=c'_{n-2}\\
&\qquad |c_n(t-1)=c_n,c_{n-1}(t-1)=c_{n-1},c_{n-2}(t-1)=c_{n-2})\\
&\quad \times P(c_n(t)=c'_n|c_{n-1}(t)=c'_{n-1},c_{n-2}(t)=c'_{n-2},\\
&\qquad c_n(t-1)=c_n,c_{n-1}(t-1)=c_{n-1},c_{n-2}(t-1)=c_{n-2})\\
=&\lim_{t\to\infty} P(c_{n-1}(t)=c'_{n-1},c_{n-2}(t)=c'_{n-2}|c_{n-1}(t-1)=c_{n-1},c_{n-2}(t-1)=c_{n-2})
\end{aligned}
$$

$$\begin{aligned}&\times q(c_n, c_{n-1}, c_{n-2}, c'_n, c'_{n-1}, c'_{n-2})\\ =&(\mathbf{P}''_n)_{c_{n-1}c_{n-2},c'_{n-1}c'_{n-2}} q(c_n, c_{n-1}, c_{n-2}, c'_n, c'_{n-1}, c'_{n-2}).\end{aligned} \tag{75}$$

It is easy to see that (30) follows from (75), (29), and Table VII.

E. COMPUTING THE CODING RATE R_N IN (S5)

From (30), (31), and Table VII, we see that

$$\begin{aligned} r_n =& - \sum_{c_n,c_{n-1},c_{n-2}\in\{0,1\}} \pi'''_{n,c_nc_{n-1}c_{n-2}} \\ &\times \sum_{c'_n,c'_{n-1},c'_{n-2}\in\{0,1\}} (\mathbf{P}'''_n)_{c_nc_{n-1}c_{n-2},c'_nc'_{n-1}c'_{n-2}} \log_2 q(c_n, c_{n-1}, c_{n-2}, c'_n, c'_{n-1}, c'_{n-2}) \\ =& (\pi'''_{n,000} + \pi'''_{n,001} + \pi'''_{n,011} + \pi'''_{n,100} + \pi'''_{n,110} + \pi'''_{n,111}) \cdot 1 \\ &+ (\pi'''_{n,010} + \pi'''_{n,101}) \cdot \left[1 - \left(\frac{r_{n-2}}{4} - \frac{1-r_{n-1}}{r_{n-1}}\right)\right] \\ =& \frac{r_{n-1}r_{n-2}\left(1 - \frac{r_{n-1}}{2}\right) + 4(3r_{n-1}-1) + 2r_{n-1}(1-r_{n-1})}{r_{n-1}r_{n-2} + 4(3r_{n-1}-1)}. \end{aligned}$$

REFERENCES

C. Alexopoulos and A. Seila, “Implementing the batch means method in simulation experiments,” *Proceedings of the 28th conference on Winter simulation. IEEE Computer Society, 1996.*

C.-S. Chang, J. Cheng, T.-K. Huang, X.-C. Huang, D.-S. Lee, and C.-Y. Chen, ”Bit-stuffing algorithms for crosstalk avoidance in high speed switching,” *IEEE Transactions on Computers*, vol. 22, no. 9, pp. 2030–2033, December 2015.

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. 64, no. 12, pp. 3404–3416, September 2014.

J. Cheng, C.-S. Chang, T.-H. Chao, D.-S. Lee, and C.-M. Lien, ”On constructions of optical queues with a limited number of recirculations,” in *Proceedings of IEEE INFOCOM 2008*.

C.-C. Chou, C.-S. Chang, D.-S. Lee and J. Cheng, “A necessary and sufficient condition for the construction of 2-to-1 optical FIFO multiplexers by a single crossbar switch and fiber delay lines,” *IEEE Transactions on Information Theory*, vol. 52, pp. 4519–4531, October 2006.

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

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.

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.

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

International Technology Roadmap for Semiconductors, 2003, Semiconductor Industry Association. [Online]. Available: http://www.itrs.net/Links/2003ITRS/Home2003.htm

International Technology Roadmap for Semiconductors, 2005, Semiconductor Industry Association. [Online]. Available: http://www.itrs.net/Links/2005ITRS/ExecSum2005.pdf

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.

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.

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

M. Mutyam, “Fibonacci codes for crosstalk avoidance,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, pp. 1899-1903, 2012.

R. Nelson. *Probability, stochastic processes, and queueing theory: the mathematics of computer performance modeling*. Springer Science and Business Media, 1995.

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

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.

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

S. R. Sridhara, "Communication inspired design of on-chip buses," Ph.D. Dissertation, University of Illinois, Urbana, IL, USA, 2006.

M. Stan and W. Burleson, "Limited-weight codes for low power I/O," in *Proc. IEEE/ACM Int. Workshop Low Power Design*, 1994. pp. 209–214.

B. Victor, "Bus encoding to prevent crosstalk delay," M. S. Thesis, University of California, Berkeley, CA, USA, 2001.

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.

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.

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*, pp. 548–558, 2011.

X. Wu, Z. Yan, and Y. Xie, "Two-dimensional crosstalk avoidance codes," in *Proceedings IEEE Workshop on Signal Processing Systems (SiPS'08)*, Washington, D. C., USA, October 8–10, 2008, pp. 106–111.