S. J. Hong
D. L. Ostapko

Codes for Self-clocking, AC-coupled Transmission: Aspects of Synthesis and Analysis

Abstract: We consider NRZI waveform codes that satisfy a given set of run-length constraints and the upper bound on the accumulated dc charge of the waveform. These constraints enable the codeword to be self-clocking, ac-coupled, and suitable for data processing tape and communication applications. Various aspects of synthesis and analysis of such codes, called (d, k, C) codes, are illustrated by means of several examples. The choice of the initial state of the encoder is shown to influence the length of the data sequence over which the encoder must look-ahead.

Introduction

In the transmission of binary data, whether on a communication link or through a magnetic recording head onto a tape, it is generally desirable to encode the data to achieve self-synchronization. A widely accepted method of obtaining this self-clocking property is to ensure that the waveforms on the channel provide a guaranteed minimum spacing between the detectable transitions. With this method it is convenient to use ac coupling of the waveform into the channel. In the case of rotating-head magnetic recording, transformer coupling of the signal becomes a necessity. The codes we describe here are aimed at such applications. The object of this paper is not to describe production of such codes *per se*, but rather to introduce some methods of synthesizing and analyzing these codes in order to achieve flexibility of choice in the design of a system.

Various waveform coding methods are used in tape applications; nonreturn to zero (NRZ), NRZ inverse (NRZI), phase modulation (PM), frequency modulation (FM), and modified FM (MFM) are some of the well known techniques. The NRZI method is widely used at the encoder-channel boundary; it accepts the binary input string and produces as output for the recording head a ONE for transition at every bit interval. Kobayashi and Tang [1] discuss the inherent potential of NRZI waveforms for some kinds of error detection. In this paper we assume the use of the NRZI waveform for the sake of conformity, even though the techniques are also applicable to the direct-waveform NRZ method.

In NRZI terms, the required maximum spacing between transitions in a system that uses self-clocking becomes the number of maximum allowable consecutive zeros, k, in the codewords. In general, it is also desirable to set a lower bound d, for the space between transitions, which is the minimum allowable string of zeros in the codewords. The waveform pulse width is then bound between the two limits $(d + 1)$ and $(k + 1)$, which are directly related to the upper and the lower cutoff frequencies of the read-write head and the supporting circuitry, respectively. The ratio d/k gives an indication of the bandwidth of these circuits and therefore should be small for design reasons. The lower bound d on the run of zeros also influences the interference between recorded transitions in saturation recording and limits the spectrum spread in frequency-shift keying [2].

Many run-length-limited codes with (d, k) constraints have been reported [2–10]. Tang [4, 5], Gabor [6], and Kautz [7] describe block-oriented, run-length-limited codes for tape applications. Tang and Bahl [10] compute the number of (d, k)-limited sequences of given block length and the asymptotic information rate of such codes. Franaszek [2, 3] uses a sequence-state approach in the construction of block- or variable-length codes. Gilder [8] reports on the successful use, in a Bell and Howell high-density tape system, of a simple scheme that forces an additional odd parity bit at small intervals of NRZ data, thereby achieving a guaranteed transition in every two such intervals.

The ac-coupling requirement further imposes a charge constraint. The waveform should have neither lengthy nor high-magnitude dc components. Expanding on Tang's notation, we consider (d, k, C) codes, where C is an upper bound on the accumulated charge of the waveform, $(f_1 f_2 \cdots f_t : f_i = 1 \text{ or } -1)$. The (d, k, C) codes have two primary constraints,

$$d \leq \text{run of zeros in code} \leq k, \text{ and} \tag{1}$$

$$\left|\sum f_i\right| \leq C. \tag{2}$$

To be able to meet the run-length constraint, each input bit cannot be mapped into just one code bit. On the average, α bits of information map into N bits of code, where $\alpha < N$. This may seem to imply redundancy and higher frequency response of the head. The actual bandwidth requirement, however, depends primarily on the run-length limits d and k as previously discussed, and hence the ratio α/N does not directly affect the system frequency requirements. Gabor [6] shows that the efficiency of the code, also, is not directly related to the difference between α and N. In general, the (d, k) or (d, k, C) constraints determine the channel capacity, K bits per N channel digits, and the ratio between α/N and K/N determines the code efficiency. The implied redundancy α/N, nevertheless, can be used for error detection, as was demonstrated by Kobayashi and Tang [1].

The MFM code achieves $(d, k) = (1, 3)$; i.e., the transition width is bound between two and four units of tape time by mapping one information bit onto a pair of code bits for the tape. This is the so-called double-window concept, the case when $\alpha = 1$ and $N = 2$. For most applications, the higher the values of α and N, the more complex the encoder and decoder. The techniques we discuss here are most effective at small values of α and N. They can be generalized to higher values but are applied there with greater computational difficulty. Therefore, we assume $\alpha = 1$ and $N = 2$ for the codes we consider. Furthermore, we assume $d = 1$, i.e., no adjacent transitions are allowed, for our examples.

Channel states and allowed transitions

Let a denote the two NRZI bits injected into the channel; r, the latest run length at the end of a; c, the total accumulated charge at the end of a; and w, the channel waveform level (± 1) at the end of a. The quadruple $[a, r, c, w]$ sufficiently describes the state of the channel at the end of each two-channel digit boundary. The (d, k, C) constraints limit the transition from a given state $[a, r, c, w]$ to other states as follows, in which $d = 1$ limits a to 00, 01, and 10:

$$[a, r, c, w] \rightarrow \begin{cases} [00, r+2, c+2w, w] \text{ only if } |c+2w| \leq C & (3) \\ \text{and } r+2 \leq k; & (4) \\ [01, 0, c, -w] \text{ only if } r+1 \leq k & (5) \\ \text{and } |c+w| \leq C; & (6) \\ [10, 1, c-2w, -w] \text{ only if } a \neq 01 & (7) \\ \text{and } |c-2w| \leq C. & (8) \end{cases}$$

After assuming that initially $c = 0$ for the channel, the incremental charge of 0 or 2 at all transitions implies that only even c's result. Also, any state with $a = 01$ has $r = 0$, and any state with $a = 10$ has $r = 1$. Conditions (5) and (6) arise because $a = 01$ increases the run length and the charge by 1 in transit. Thus, for even C, C and $C - 1$ have the same effect, and we may assume C to be odd. Consequently, condition (6) is redundant because $|c| \leq C - 1$.

From the viewpoint of the encoder, the a's are the only outputs to the channel and there is an obvious isomorphism between states [11] as

$$[a, r, c, w] \leftrightarrow [a, r, -c, -w], \tag{9}$$

which can be easily verified by inspecting all successors of $[a, r, c, w]$ and those of $[a, r, -c, -w]$. Therefore, we need consider only half of the total channel states by nominally choosing $w = 1$ and representing the states by triplets, $[a, r, c]$.

Consider transitions to $a = 10$, i.e., $[10, 1, c-2w, -w]$. The two possible transition are either from $[00, r, c, w]$ or $[10, 1, c, w]$. For the case of transition from $a = 00$, $c = c' + 2w$ and $|c'| \leq C$, hence $|c - 2w| \leq C$. For the other case, $c = c' - 2(-w)$ and $|c'| \leq C$; hence, again, $|c - 2w| \leq C$, where c' denotes the charge of the grandfather state. Therefore, the condition (8) is redundant. Using the isomorphism, then, we can simplify the allowed state transitions as follows. Notice the negative charge designation for successor states with $a = 01$ and 10 due to the convention that $w = 1$.

$$[a, r, c] \rightarrow \begin{cases} [00, r+2, c+2] \text{ only if } c+2 \leq C \\ \text{and } r+2 \leq k; & (10) \\ [01, 0, -c] \text{ only if } r+1 \leq k; & (11) \\ [10, 1, 2-c] \text{ only if } a \neq 01. & (12) \end{cases}$$

Any state that is not reachable from the initial state assumed by the encoder need not be considered at all. We now estimate the total number of states that must be considered under the $(1, k, C)$ constraints. Obviously, $[01, 0, c]$ states can have all even c's in the range $1 - C \leq c \leq C - 1$. Therefore, No.$[01, 0, c] = C$, where "No." denotes the number of possible states. For $[10, 1, c]$

Table 1 The number of states for $(1, k, C)$ constraints.

k, C	*Number*	k, C	*Number*
2, 3	7	2, 5	13
3, 3	8	3, 5	16
4, 3	9	4, 5	19
∞, 3	9	∞, 5	25

states, $c = 1 - C$ is not allowed because $c = 2 - c' = 1 - C$ implies $c' = C + 1 > C$, where c' is the charge of any predecessor state. Therefore, No.$[10, 1, c] = C - 1$. For states with $a = 00$, notice that $r = 2$ is the smallest allowed value and that $[00, 2, c]$ must come from $[01, 0, c-2]$; $[00, 3, c]$ must come from $[10, 1, c-2]$; and, for $r \leq 4$, the state $[00, r, c]$ exists if and only if $[00, r-2, c-2]$ exists. For a given $r = r_0$, then

$$\text{No.}[00, r_0, c] = \begin{cases} C - \left\lceil \frac{1}{2} r_0 \right\rceil & \text{if } 2 \leq r_0 < 2C; \\ 0 & \text{if } 2C \leq r_0 \leq k. \end{cases} \quad (13)$$

Summing for all r_0, $2 \leq r_0 \leq k$,

$$\text{No.}[00, r, c] = \begin{cases} (k-1)C - m(m+1) + 1 & \text{if } k = 2m < 2C; \\ (k-1)C - m(m+2) & \text{if } k = 2m+1 < 2C; \\ (C-1)^2 & \text{if } k \geq 2C. \end{cases} \quad (14)$$

Hence, the total number of states can be written as

$$\text{No.}[a, r, c] = \begin{cases} (k+1)C - m(m+1) & \text{if } k = 2m < 2C; \\ (k+1)C - 1 - m(m+2) & \text{if } k = 2m+1 < 2C; \\ C^2 & \text{if } k \geq 2C. \end{cases} \quad (15)$$

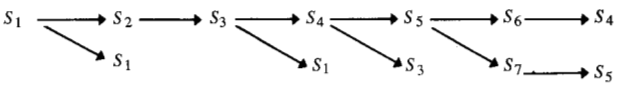
For small k and C values the total number of states given by Eq. (15) is within easily manageable range, as shown in Table 1 for some $(1, k, C)$ constraints.

Transition probability assignment

The encoder must know the current channel state and the input bit to determine which allowed successor state should be chosen. The encoder then injects the a value of the chosen successor as an NRZI waveform into the channel, which actually accomplishes the channel-state transition. The encoder must route the input data sequence, in proportion to the transition probability, to the allowed successors.

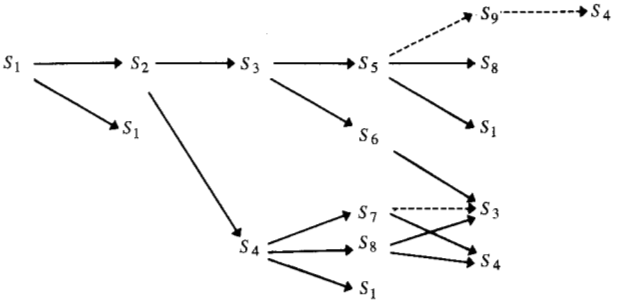
Given $(1, k, C)$ constraints, we determine the number of channel states and their transition rules. To find all states and their transitions, one has merely to begin with any known existing state, say $[a, r, c] = [01, 0, 0]$, and iteratively find successors by the rules of Eqs. (10–12). We show three examples here.

$(d, k, C) = (1, 2, 3)$:



where $S_1 = [01, 0, 0]$, $S_2 = [00, 2, 2]$, $S_3 = [10, 1, 0]$, $S_4 = [10, 1, 2]$, $S_5 = [01, 0, -2]$, $S_6 = [00, 2, 0]$, and $S_7 = [01, 0, 2]$.

$(d, k, C) = (1, 3, 3)$ and $[(1, 4, 3) : (1, 3, 3)$ does not include the dotted transitions]:



where $S_1 = [01, 0, 0]$, $S_2 = [00, 2, 2]$, $S_3 = [01, 0, -2]$, $S_4 = [10, 1, 0]$, $S_5 = [00, 2, 0]$, $S_6 = [01, 0, 2]$, $S_7 = [00, 3, 2]$, $S_8 = [10, 1, 2]$, and $S_9 = [00, 4, 2]$.

Since we are concerned with the $\alpha = 1$ and $N = 2$ case, we require transition probabilities that permit a channel capacity per state of one bit. When the channel capacity exceeds one bit (the upper bound being two bits when $d = 0$ and $k = C = \infty$), we prune away some states and/or transitions to obtain the one-bit capacity and assign transition probabilities attaining that capacity. Following Shannon [12] (see also [13] for a specific treatment of this subject), we use the connection matrix $\mathbf{A} = [a_{ij}]$, defined for n states as

$a_{ij} = 1$ if the $S_i \rightarrow S_j$ transition is allowed, and

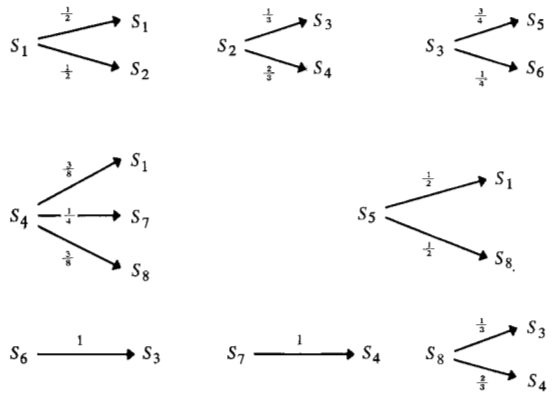
$a_{ij} = 0$ otherwise.

It has been shown that (*i*) $\mathbf{A}$ has a maximum magnitude, positive, real eigenvalue λ_0; (*ii*) the channel capacity is $\log_2 \lambda_0$ bits per state; (*iii*) the eigenvectors corresponding to λ_0 all consist of multiples of some vector $\mathbf{X} = (x_1, x_2, \cdots, x_n)$, where the x_i are all positive; and (*iv*) transition probability assignments of $p_{ij} = a_{ij} x_j / \lambda_0 x_i$ attain the channel capacity $\log_2 \lambda_0$.

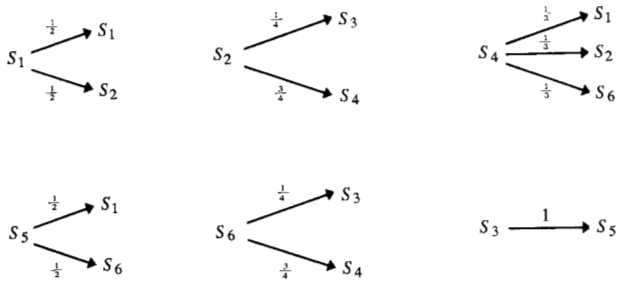
Obviously, the code does not exist if $\lambda_0 < 2$, which is the case for $(d, k, C) = (1, 2, 3)$ constraints, which results in $\lambda_0 = 1.802$. For $(d, k, C) = (1, 3, 3)$, $\lambda_0 = 2$ and the corresponding eigenvector is

$\mathbf{X} = (3, 3, 2, 4, 3, 1, 2, 3)$; also

$p_{ij} = a_{ij} x_j / 2x_i$; and



For the case $(d, k, C) = (1, 4, 3)$, we have $\lambda_0 > 2$. There are excess transitions in this case and we prune those single transition states one at a time in some trial-and-error order until $\lambda_0 = 2$ is obtained. This can be done in this case by removing the two single transition states S_6 and S_9. After the removal λ_0 becomes 2; S_2 and S_7 are isomorphic, and we may rename the states as $S_1 = [01, 0, 0]$, $S_2 = [00, (2 \text{ or } 3), 2]$, $S_3 = [01, 0, -2]$, $S_4 = [10, 1, 0]$, $S_5 = [00, 2, 0]$, and $S_6 = [10, 1, 2]$. The eigenvector corresponding to $\lambda_0 = 2$ can be calculated as $\mathbf{X} = (2, 2, 1, 3, 2, 2)$, and the transition probabilities are



Initial state consideration

The encoder starts with an assumed initial state and forces the state transitions in the channel according to the binary input sequence. Given an arbitrarily long input sequence, the decoder must be able to divide all possible input sequences in proportion to the transition probabilities of its successor states. Suppose the initial state is S_{i_0}. Then the probability of channel sequence

$$S_{i_0} \to S_{i_1} \to S_{i_2} \to \cdots \to S_{i_t}$$

for some t is

$$\frac{x_{i_1}}{\lambda_0 x_{i_0}} \frac{x_{i_2}}{\lambda_0 x_{i_1}} \cdots \frac{x_{i_t}}{\lambda_0 x_{i_{t-1}}} = \frac{1}{\lambda_0^t} \frac{x_{i_t}}{x_{i_0}}.$$

Now, if $x_{i_t}/x_{i_0} \neq e/2^h$ for some integers e and h; the number of input sequences the encoder routes to S_{i_t} must be proportional to $1/p$ of all possible binary input sequences for some prime p other than 2. However, to be able to distinguish $1/p$ of an arbitrarily long (say, infinite) binary sequence, one must look at the entire length of the input sequence, i.e., an infinite look-ahead of the input data is necessary.

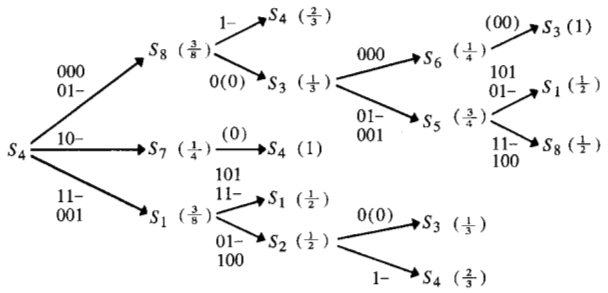


Figure 1 A $(1, 3, 3)_A$ code assignment.

Suppose the eigenvector $\mathbf{X}$ is normalized so that every component x_i is the smallest possible integer. Since all the channel states we can use must be strongly connected [11], the states can be divided into two categories according to whether the given state used as an initial state would require a finite or an infinite look-ahead encoder. That is, S_i is a finite encoder initial state if and only if the greatest odd divisor of x_i is also a divisor of all x_j, $j \neq i$. For the example $(d, k, C) = (1, 3, 3)$, discussed above, $\{S_1, S_2, S_5, S_8\}$ is the set of states that require infinite look-ahead encoders when used as the initial state, and $\{S_3, S_4, S_6, S_7\}$ is the set of states that require finite look-ahead encoders; for the $(d, k, C) = (1, 4, 3)$ case, S_4 is the only infinite look-ahead initial state.

Code construction: mapping inputs onto transitions

• *Finite look-ahead case*

If a state requiring a finite look-ahead encoder is chosen as the initial state, the code construction amounts to specifying the encoder (and decoder) function that maps the binary input data to the channel transition. For example, if S_4 of the $(1, 3, 3)$ constraint design is the initial state, the encoder could map the inputs to transitions as follows.

$$S_4 \to \begin{cases} S_8 \text{ (prob} = 3/8) & \text{iff } I = 000\cdots \text{ or } I = 01\cdots; \\ S_7 \text{ (prob} = 1/4) & \text{iff } I = 10\cdots; \\ S_1 \text{ (prob} = 3/8) & \text{iff } I = 001\cdots \text{ or } I = 11\cdots. \end{cases}$$

The whole mapping can be carried out using an encoder tree such that all transitions appear just once. One feasible assignment, which we call the $(1, 3, 3)_A$ code, is shown in Fig. 1. The leftmost input bits in the transition branches are the current input to the encoder, and the rest of the bits are future data bits. Some data bits in parentheses are predetermined by the previous transitions.

No method is known for optimal transition assignments. For practical encoder and decoder implementations, a trial process using the following guidelines is recommended.

1. Always route the input sequences in proportion to the transition probabilities.
2. Consider the decoding problem, so that it does not require too many code states for deciphering the data bits from the code sequences.
3. Try, as much as possible, to assign transitions such that the code-state to code-state $[a, r, c]$ transitions get consistent input data values. (Franaszek [3] calls this the "state-independence" criterion for his (d, k) codes.)
4. For all transitions to a given state, the prespecified future data bits should be consistent, e.g., $S_1 \rightarrow S_1$, $S_5 \rightarrow S_1$ in Fig. 1.

After defining the mapping, the states must be coded using binary-state variables. Three-state variables $P_1P_2P_3$ can be used to denote the states S_1 through S_8. With time denoted by superscripts, the $(1, 3, 3)_A$ code encoder is a combinational circuit with six inputs $(P_1^0P_2^0P_3^0I^1I^2I^3)$ and five outputs $(P_1^1P_2^1P_3^1a_1^1a_2^1)$. An optional error output E may be created using the same input variables. This signal checks the encoder circuit by detecting any illegal combination of states and inputs. The decoder requires three code states to uniquely decipher the binary data. Thus the decoder is a combinational circuit with nine inputs $(P_1^0P_2^0P_3^0a_1^1a_2^1a_1^2a_2^2a_1^3a_2^3)$ and four outputs $(P_1^1P_2^1P_3^1I^1)$. Again, an optional error function can be created using the same inputs, providing powerful channel error detection. Another example of a transition assignment for the $(1, 3, 3)$ code and an example of a $(1, 4, 3)$ code are given in the Appendix.

The $(1, 3, 3)_A$ code synthesized above requires a state calculation in both encoding and decoding. The fact that the state calculation is necessary in decoding raises a question of error propagation. One can use a Viterbi-type correction procedure en route if the probability of the most likely true code values can be decided by the engineer. Also, there is a high probability that most of the errors will resynchronize back to the correct state after a few miscalculations. The fact that the state-dependent decoding is used can also be utilized as a powerful error detection means for the transmitted message. There are many more possible codes, some of which may provide a better implementation. The objective here is simply to demonstrate the method of construction.

• *Infinite look-ahead case*

When the initial state requires an infinite look-ahead, it is necessary to provide an infinite look-ahead function that distinguishes $1/p$ of the binary sequence for some $p \neq 2$. For the $(1, 3, 3)$ example, one-third of all binary sequences must be distinguished. One method of distinguishing one-third of an infinite sequence is to classify by magnitude, i.e.,

0	00	00	00	···	00	···
$\frac{1}{3}$	01	01	01	···	01	···
$\frac{2}{3}$	10	10	10	···	10	···
1	11	11	11	···	11	···

$$\sum_{i=1}^{\infty}\left(\frac{1}{4}\right)^i=\frac{1}{3}.$$

Hence, given that the current bit is 0, the sequence belongs to $[0, \frac{1}{3})$ if and only if at least two consecutive 0's occur (including the current bit) before at least two consecutive 1's occur.

Let Z be the variable which is 0 if the current data bit is 0 and a 0 burst precedes a 1 burst, and is 1 otherwise. For example, if the

$$\text{data} = 0010110001011000101\cdots,$$

then $Z = 0111110011111100????\cdots$.

One possible assignment of transition is shown in Fig. 2 as a $(1, 3, 3)_C$ code. In Fig. 2 the data bits in parentheses denote past data bits for encoding and decoding. One observes that this code has a consistent assignment which permits state-independent decoding.

Another way of distinguishing one-third of the infinite sequences is by using either 1's or 0's parity on the data. Let P be the look-ahead-1's burst parity; i.e., $P = 1$ if and only if the current data bit is a 1 and the consecutive number of 1's including the current bit, is odd.

10

1110 $\qquad \sum_{i=1}^{\infty}\left(\frac{1}{4}\right)^i=\frac{1}{3},$

111110

etc.

A $(1, 3, 3)$ code using this P function in the assignment was originally discovered by Patel [14] and was called the ZM (zero-modulation) code. For the detailed engineering implementation and the merits of this code, see [14].

Worst-case analysis

For an engineering application, an optimal selection has to be made from a wide variety of codes that can be synthesized by the techniques discussed above. Should an infinite buffer be allowed for the sake of simpler decoding? Is state dependency of decoding a hazard or a boon for error detection in the application? What is the extent of minimization of the encoder and decoder circuits? These questions should be considered before a choice is made.

Another useful consideration in code design concerns the data characters. Often the data stream is "padded" with 0's. Also, the data may consist of long bursts of 1's. For these frequent data sequences, one may desire the waveform on the tape to be the most reliable one, i.e., a waveform with the smallest pulse width and charge drift.

The code sequences corresponding to either the zero burst (0^{∞}) or the one burst (1^{∞}) in the data can be easily obtained by tracing the code tree as in Figs. 1 and 2. The example codes $(1, 3, 3)_A$ and $(1, 3, 3)_C$ both exhibit $(10)^{\infty}$ code sequences for the one burst in the input data sequence. The $(1, 3, 3)_A$ code also produces $(10)^{\infty}$ for the zero burst of the data. However, the $(1, 3, 3)_C$ code produces a $(1000)^{\infty}$ code sequence for a zero burst data sequence.

The code sequence $(1000)^{\infty}$ is, of course, a succession of the widest pulses for $(d, k) = (1, 3)$. If a special data sequence is used for a synchronization marker, the corresponding code sequence is also desired to be the most reliable one. Again, tracing the code tree at various points with the given data sequence yields the code sequence for such analysis.

The worst-case code output may provide another criterion for code choice. We have seen that the $(1, 3, 3)_C$ code has $(1000)^{\infty}$ as its worst pattern. For the $(1, 3, 3)_A$ code, the worst-case code sequence $(1000)^{\infty}$ corresponds to the input data pattern $001(0001)^{\infty}$, starting from state S_3.

Run-length statistics

Once the transition probabilities P_{ij} are known, the stationary distribution of states can be calculated in the usual manner from the transition probability matrix. For $(1, 3, 3)$ codes, the stationary probabilities of states become

State	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8
Probability	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{9}$	$\frac{2}{9}$	$\frac{1}{12}$	$\frac{1}{36}$	$\frac{1}{18}$	$\frac{1}{8}$

The states S_1, S_3, S_4, S_6, and S_8 each contain one transition. Therefore the transition probability is

$$P_t = P(S_1) + P(S_3) + P(S_4) + P(S_6) + P(S_8) = 53/72.$$

On the average, there are P_t transitions in the waveform for each bit of input data. Since each transition marks the boundary of a pulse, the average pulse width of the waveform is just the inverse of the transition probability P_t; i.e.,

$$\text{Average pulse width} = 1/P_t = 72/53 = 1.358 \text{ bit units}$$
$$= 2.717 \text{ window units.}$$

Similarly, one can obtain the probabilities of each of the allowed run-length sequences in the channel by gathering the probabilities of transitions that produce the given run-length. For the $(1, 3, 3)$ codes, they are

Run-length	*NRZI*	*Probability*	*Percent*
2	101	25/53	47
3	1001	18/53	34
4	10001	10/53	19

The average pulse width and the run-length distribution can be important engineering criteria.

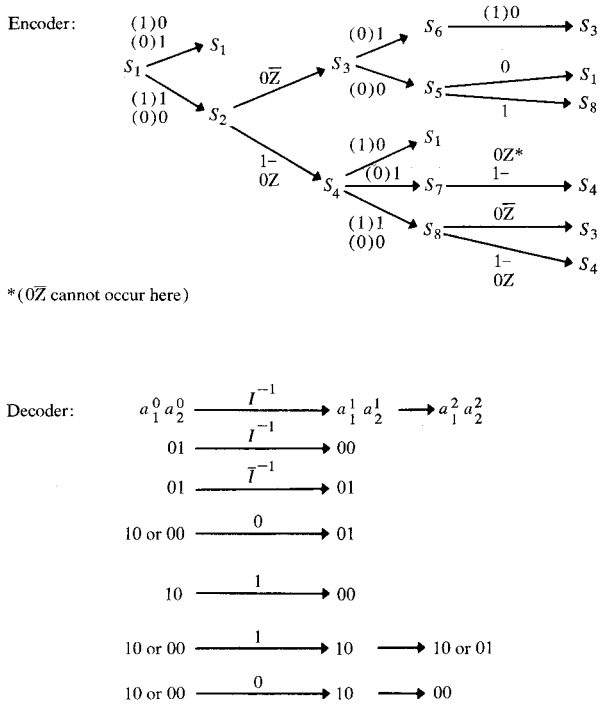


Figure 2 A $(1, 3, 3)_C$ code encoder and decoder assignment.

Summary

Various aspects of the synthesis and analysis of (d, k, C) codes have been presented. Channel states and allowed transitions for a given set of constraints have been enumerated, and it has been shown that the code synthesis depends strongly on the assumption of a particular initial channel state. A procedure for determining those initial states that lead to encoders with infinite look-ahead requirements has been given. The techniques of code construction have been illustrated by means of examples.

The procedure used in this paper may be generalized to non-NRZI cases, and α, N, and d may assume values other than those used in the examples.

Appendix A: An alternate (1, 3, 3) code: $(1, 3, 3)_B$

Sometimes a code can be constructed for simpler implementation by splitting a state into two, such that a more consistent input assignment may be carried out.

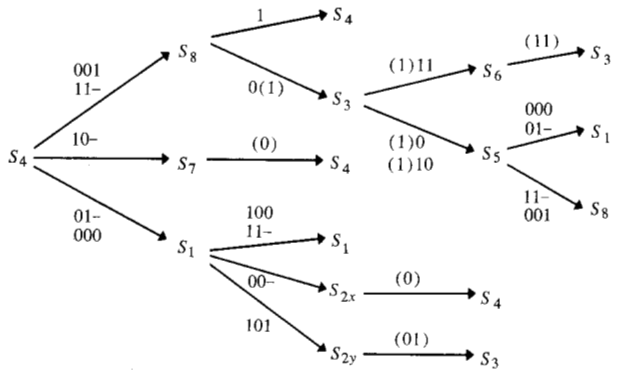
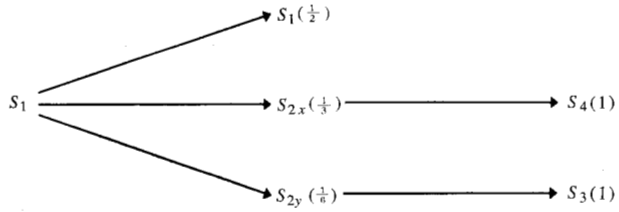


Figure A1 A split-state $(1, 3, 3)_B$ code.

In this example the state S_2 is split into two states S_{2x} and S_{2y}. The only modifications due to this split are



A highly consistent assignment of transitions is now possible, as shown in Fig. A1.

Since there are nine states, at least four state variables are necessary to code the states. The a values (a_1a_2) and two additional variables (T_1T_2) to differentiate S_{2x}, S_{2y}, S_5, and S_7 are selected. The encoder requires fewer outputs due to the high consistency of the $(1, 3, 3)_B$ code. Notice that the current input from S_{2x} and S_{2y} is always 0 and from S_3 is always 1. The encoder is a combinational circuit with seven inputs $(a_1^0a_2^0T_1^0T_2^0I^1I^2I^3)$ and four outputs $(a_1^1a_2^1T_1^1T_2^1)$. The decoder can be identical with that of the $(1, 3, 3)_A$ code because the states S_{2x} and S_{2y} need not be distinguished.

Appendix B: Assignment of transitions for (1, 4, 3) code

A (1, 4, 3) code assignment can be made as in Fig. B1, where S_1 is chosen for the initial state. It is possible in this case to assign inputs to transitions such that decoding becomes almost state-independent. Data bits in parentheses are past data bits.

Since there are only six states, we need only three binary state variables. The a values (a_1a_2) and one additional variable b to distinguish states S_1, S_4, and S_5 from the states S_2, S_3, and S_6 are selected. The encoder takes six inputs $(a_1^0a_2^0b^0I^{-1}I^0I^1)$ and generates four outputs $(a_1^1a_2^1b^1E)$, including the error function. The decoder requires seven inputs $(a_1^0a_2^0b^0a_1^1a_2^1a_1^2a_2^2)$ and generates three outputs (I^0b^1E). These combinational functions are extremely simple and the minimized functions using the error conditions as full DON'T CAREs are shown below (+ denotes OR).

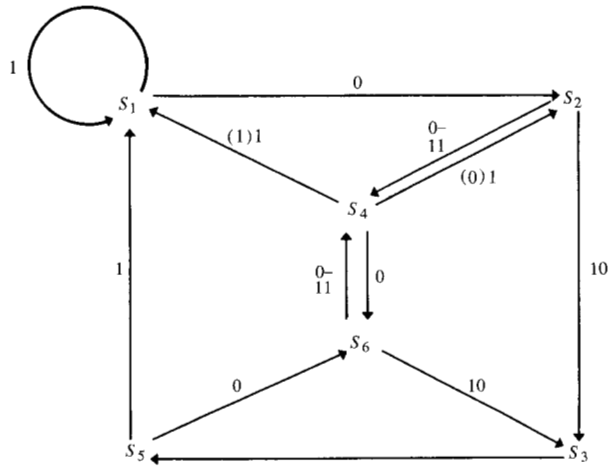


Figure B1 A (1, 4, 3) code assignment.

Encoder: $a_1^1 = \bar{a}_2^0(\bar{I}^0 + I^0I^1)$;

$$a_2^1 = \bar{a}_2^0b^0I^0\bar{I}^1 + \bar{b}^0(\bar{a}_1^0I^0 + a_1^0\bar{a}_2^0I^{-1});$$

$$b^1 = \bar{b}^0(\bar{I}^0 + a_1^0\bar{a}_2^0\bar{I}^{-1}) + \bar{a}_2^0b^0I^0;$$

E can be similarly obtained.

Decoder: $I^0 = a_2^1 + \bar{a}_2^0b^0a_1^1a_2^2$;

$$b^1 = b^0a_2^1 + \bar{b}^0\bar{a}_2^1;$$

$$E = \bar{a}_1^0\bar{a}_2^0\bar{b}^0\bar{a}_1^1\bar{a}_2^1 + a_1^0a_2^0 + a_1^1a_2^1 + a_2^0a_1^1 + a_2^0b^0a_2^1.$$

In general, more relaxed (d, k, C) constraints bring a larger choice of codes and simpler encoder-decoder circuits. All of the (1, 3, 3) codes require much more complex Boolean realizations than the above (1, 4, 3) example.

Acknowledgment

We are grateful to one of the referees for his valuable suggestions for improving the original paper.

References

1. H. Kobayashi and D. T. Tang, "Application of Partial Response Channel Coding to Magnetic Recording Systems," *IBM J. Res. Develop.* **14,** 368 (1970).
2. P. A. Franaszek, "Sequence-State Methods for Run-Length-Limited Coding," *IBM J. Res. Develop.* **14,** 376 (1970).
3. P. A. Franaszek, "Sequence-State Coding for Digital Transmission," *Bell System Tech. J.* **47,** 143 (December 1967).
4. D. T. Tang, "Run-Length Limited Codes," *IEEE International Symposium on Information Theory*, Ellenville, New York, 1969.
5. D. T. Tang, "Practical Coding Schemes with Run-Length Constraints," *Research Report RC-2022*, IBM Thomas J. Watson Research Center, Yorktown Heights, New York 10598, 1968.

6. A. Gabor, "Adaptive Coding for Self-Clocking Recording," *IEEE Trans. Electronic Computers* **EC-16,** 866 (1967).
7. W. H. Kautz, "Fibonacci Codes for Synchronization Control," *IEEE Trans. Information Theory* **IT-11,** 284 (1965).
8. J. H. Gilder, "Data Storage Improves with New Magnetic and Paper Tape Units," *Electronic Design* **24,** 140 (November 22, 1973).
9. P. A. Franaszek, "Determination of the Existence of Codes for Synchronous Transmission of Binary Data Over Discrete Constrained Channels," *Research Report RC-2468*, IBM Thomas J. Watson Research Center, Yorktown Heights, New York, 10598, 1968.
10. D. T. Tang and L. R. Bahl, "Block Codes for a Class of Constrained Noiseless Channels," *Information and Control* **17,** 436 (1970).
11. Z. Kohavi, *Switching and Finite Automata Theory*, McGraw-Hill Book Co., Inc., New York, 1970.
12. C. E. Shannon, "A Mathematical Theory of Communication," *Bell System Tech. J.* **27,** 379 and 623 (1948).
13. R. B. Ash, *Information Theory*, John Wiley & Sons, Inc., New York, 1965, pp. 209–210 and 327–328.
14. A. M. Patel, "Zero-Modulation Encoding in Magnetic Recording," *IBM J. Res. Develop.* **19,** 366 (1975, this issue).

Received March 8, 1974; revised February 20, 1975

The authors are located at the IBM System Products Division laboratories, Poughkeepsie, New York 12602. S. J. H. is on leave at the University of Illinois.