ode's error correcting properties are unchanged but off-
time correlations are minimized. This code-form supplies
at least a part of the synchronization power which is
supplied by the pilot-tone of the two-channel scheme.
In almost all situations of interest, it supplies all that is
necessary.

One may generalize from the results of Section 2 to
say that the power is optimally divided when the proba-
bility of a word error given correct synchronization
is equal to the probability of a synchronization error.
Consequently, a one-channel system will be called “optimally
self-synchronizable” if its synchronization error probability
after reception of \( \beta^2/k \) words is at most equal to the
synchronized word error probability. That this condition
is satisfied for most situations of interest may be inferred
from Table 1, extracted from Tables 6.2 and 6.3 of Ref.(10).
The table is for orthogonal codes with word error probability \( 10^{-9} \). \((\beta^2/k)_{\text{min}}\) is the minimum value of
that parameter for which the synchronization error probability \( \leq 10^{-9} \). \( E \{(\beta^2/k)_{\text{min}}\} \) is the expected value
assuming all code vectors are equally likely, whereas
max \((\beta^2/k)_{\text{min}}\) is an absolute upper-bound.

Table 1. Minimum values of \((\beta^2/k)\) for which a comma-
free code is optimally self-synchronizable

<table>
<thead>
<tr>
<th>( W = 2^k )</th>
<th>( E {(\beta^2/k)_{\text{min}}} )</th>
<th>( \max {(\beta^2/k)_{\text{min}}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>10</td>
<td>—</td>
</tr>
<tr>
<td>16</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>32</td>
<td>5</td>
<td>15</td>
</tr>
<tr>
<td>64</td>
<td>3</td>
<td>13</td>
</tr>
<tr>
<td>128</td>
<td>( \sim 1 )</td>
<td>4</td>
</tr>
</tbody>
</table>

The timing variations which exist at the word-
synchronization level are also always the low-frequency
error, i.e. the “skipping”, of phase-locked loop operating
at the RF carrier or subcarrier level. The rate of this
skipping is usually quite low (Ref. 6), and hence \( \beta^2/k \) is
apt to be several orders of magnitude larger than the
constraint values of Table 1. Knowing this, it is difficult
to envision a design situation in which the synchroniz-
ability of the comma-free code would not be adequate.

6. Discussion

In most channels, the constraint upon both the comma-
free code and the pilot-tone system is the number of
words which the receiving equipment is able to use to
determine the sync position. Since the comma-free code
provides its own sync after receiving only a small num-
ber of words, while a pilot tone synchronizable in the
same time would require a fairly large fraction of the
available power for sync, the comma-free code would
seem to be preferable in all cases.

However, the self-synchronization property of the
comma-free block codes can only be utilized through
sophisticated receiver processing. If transmitter power is
cheap, and complex receiving equipment expensive, as in
a ground-to-vehicle space telemetry application, a two-
channel system is preferable; but if transmitter power is
severely limited, and the sophisticated receiver processing
no problem, as in a vehicle-to-ground telemetry applica-
tion, the one-channel self-synchronizing system is far
more preferable to one using pilot-tone synchronization.

The conclusion is that for a Voyager-class telemetry
system, self-synchronizing codes provide the best syn-
chronization method.

The analysis has assumed that bit timing is known.
Imperfect bit timing causes an identical decrease in the
effective signal strength at both the message detector
and the maximum-likelihood word-timing detector. If “g”
denotes the ratio of the effective signal power to the
true signal power (given the degree of bit-timing uncer-
tainty), a first-order correction for bit timing can be
obtained by substituting “\( gP \)” for the message signal
power and “\( g(1-a)P \)” for the maximum-likelihood-
detected pilot-signal power in the foregoing analyses. In
most cases, this would require only slight modification
of the results.

D. A Serial Orthogonal Decoder

R. R. Green

In this article a new approach to the decoding problem
for certain block coded communication systems is pre-
presented. A simple and efficient decoder is presented.

Assume that a code word is selected from one of the
2^n code words in the dictionary \( H_n \), where \( H_n \) is defined by

\[ H_n = H_{n-1} \otimes H_1 \]
and

\[ H_1 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \]

(\(\otimes\) is the symbol for the Kroneck Product). This word is then transmitted over a channel which adds white Gaussian noise and is then available as a received signal \(x(t)\). If we let \(\tau\) be the time required to transmit each symbol of the code word, then the time required to transmit the whole word is \(T = 2^\tau\). It has been shown (Ref. 11) that to do optimal decoding we want to find \(k\) such that

\[ 1 \leq k \leq 2^n \text{ and } c_k = \max_{j=1, \ldots, 2^n} \{c_j\} \]

where

\[ c_i = \int_0^1 x(t)h_i(t)dt \]

and where \(h_i(t)\) is one of the \(2^n\) possible code words (or one of the \(2^n\) rows of \(H_n\)).

Since \(h_i(t) = \pm 1\) for all \(t\) we have

\[ c_i = \sum_{j=1}^{2^n} x_j h_{ij} \]

where

\[ x_j = \int_{(j-1)\tau}^{j\tau} x(t)dt \text { and } h_{ij} \text { is the } j \text{th bit of } h_i(t). \text{ So:} \]

\[ c_k = \max_{j=1, \ldots, 2^n} \{c_j\} \]

\[ = \max_{j=1, \ldots, 2^n} \{(H_n x)_j\} \]

\[ = \max_{j=1, \ldots, 2^n} \{y_j\} \]

where

\[ y = H_n x. \]

If we assume that the components of the vector \(x\) are available sequentially as \(2^n\) \(q\) bit serial binary words, we would like to find a machine which would perform the operation \(H_n x\). However, as this operation requires \(2^{n+1}\) additions or subtractions, it is rather inefficient and difficult to mechanize.

Instead, a more efficient and more easily mechanized procedure is as follows.

**Define:**

1. \(P_1 = I_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\)

   \[ P_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ and inductively;} \]

   \[ P_{n+1} = (I_1 \otimes P_n) (P_2 \otimes I_{n-1}) \quad n \geq 1 \]

2. \(R_1 = H_1\) and inductively;

   \[ R_{n+1} = (P_2 \otimes I_{n-1}) (I_1 \otimes R_n) \quad n \geq 1 \]

Note that \(P_n\) is a \(2^n \times 2^n\) permutation matrix; therefore, \(P_n^{-1} = P_n^r\).

Similarly \(P_k\) for any \(k \geq 1\) must be a \(2^n \times 2^n\) permutation matrix, therefore

\[ (P_n^k)^{-1} = (P_n^k)^r = (P_n^k)^k \]

Also, the matrix \(R_n\) has been described by Koerner in SPS 37-17, Vol. IV, page 72.

---

**Lemma 1:**

\[ P_n = (I_k \otimes P_{n-k}) (P_{k+1} \otimes I_{n-k-1}) \quad \text{for } n - 1 \geq k \geq 0 \]

**Proof by induction:** Trivial for \(n\) arbitrary \(k = 0\). True by definition for \(k = 1\).

Assume true for all \(n \geq k' + 1\) where \(k \geq k' \geq 0\), prove for \(k + 1\) for all \(n \geq (k + 1) + 1 = k + 2\)

\[ P_n = (I_k \otimes P_{n-k}) (P_{k+1} \otimes I_{n-k-1}) \]

\[ = (I_k \otimes (I_1 \otimes P_{n-k-1}) (P_{k+1} \otimes I_{n-k-2}) \]

\[ = (I_{k+1} \otimes P_{n-k-1}) (I_k \otimes P_{k+1} \otimes I_{n-k-2}) \]

\[ = (I_{k+1} \otimes P_{n-k-1}) [I_k \otimes P_{k+1} \otimes I_{n-k-2}] \]

\[ = (I_{k+1} \otimes P_{n-k-1}) \]

\[ = (I_{k+1} \otimes P_{n-k-1}) \]
Lemma 2:

\[ P_{n+1} (I_1 \otimes P_n^T) = (P_n^T \otimes I_1) P_{n+1}, \quad n \geq 1 \]

Proof by induction: Trivial for \( n = 1 \). For \( n = 2 \) see Fig. 8. Assume true for \( n = k \), prove for \( n = k + 1 \)

\[ P_3 (I_1 \otimes P_2) = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0
\end{bmatrix} \begin{bmatrix}
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0
\end{bmatrix} = P_3 (I_1 \otimes P_2) \]

Fig. 8. Lemma 2 \(- n = 1 \)

\[ P_{k+2} (I_1 \otimes P_{k+1}^T) = (I_k \otimes P_2) \left( P_{k+1} \otimes I_1 \right) \left( I_1 \otimes P_k^T \otimes I_1 \right) \left( I_k \otimes P_2 \right) \]

\[ = (I_k \otimes P_2) \left[ P_{k+1} \left( I_1 \otimes P_k^T \otimes I_1 \right) \left( I_k \otimes P_2 \right) \right] \]

\[ = (I_k \otimes P_2) \left[ P_k \left( I_1 \otimes P_{k-1} \otimes I_1 \right) \left( I_k \otimes P_2 \right) \right] \]

\[ = (I_k \otimes P_2) \left( P_k \otimes I_2 \right) \left( I_{k-1} \otimes P_2 \otimes I_1 \right) \left( I_k \otimes P_2 \right) \]

\[ = (I_k \otimes P_2) \left( P_k \otimes I_2 \right) \left( I_{k-1} \otimes P_3 \otimes I_1 \right) \left( I_k \otimes P_2 \right) \]

\[ = (I_k \otimes P_2) \left( P_k \otimes I_2 \right) \left( I_{k-1} \otimes P_3 \otimes I_1 \right) \left( I_k \otimes P_2 \right) \]

\[ = (I_k \otimes P_2) \left( P_k \otimes I_2 \right) \left( I_{k-1} \otimes P_3 \otimes I_1 \right) \left( I_k \otimes P_2 \right) \]

\[ = (P_k \otimes I_2) \left( P_{k-1} \otimes P_2 \otimes I_1 \right) \left( I_k \otimes P_2 \right) \]

\[ = (P_k \otimes I_2) \left( P_{k-1} \otimes P_2 \otimes I_1 \right) \left( I_k \otimes P_2 \right) \]

\[ = (P_k \otimes I_2) \left( P_{k-1} \otimes P_2 \otimes I_1 \right) \left( I_k \otimes P_2 \right) \]

\[ = (P_k \otimes I_2) \left( P_{k-1} \otimes P_2 \otimes I_1 \right) \left( I_k \otimes P_2 \right) \]

\[ = (P_k \otimes I_2) \left( P_{k-1} \otimes P_2 \otimes I_1 \right) \left( I_k \otimes P_2 \right) \]

\[ = (P_k \otimes I_2) \left( P_{k-1} \otimes P_2 \otimes I_1 \right) \left( I_k \otimes P_2 \right) \]

\[ = (P_k \otimes I_2) \left( P_{k-1} \otimes P_2 \otimes I_1 \right) \left( I_k \otimes P_2 \right) \]

\[ = (P_k \otimes I_2) \left( P_{k-1} \otimes P_2 \otimes I_1 \right) \left( I_k \otimes P_2 \right) \]

\[ = (P_k \otimes I_2) \left( P_{k-1} \otimes P_2 \otimes I_1 \right) \left( I_k \otimes P_2 \right) \]

\[ = (P_k \otimes I_2) \left( P_{k-1} \otimes P_2 \otimes I_1 \right) \left( I_k \otimes P_2 \right) \]

\[ = (P_k \otimes I_2) \left( P_{k-1} \otimes P_2 \otimes I_1 \right) \left( I_k \otimes P_2 \right) \]

\[ = (P_k \otimes I_2) \left( P_{k-1} \otimes P_2 \otimes I_1 \right) \left( I_k \otimes P_2 \right) \]

\[ = (P_k \otimes I_2) \left( P_{k-1} \otimes P_2 \otimes I_1 \right) \left( I_k \otimes P_2 \right) \]

\[ = (P_k \otimes I_2) \left( P_{k-1} \otimes P_2 \otimes I_1 \right) \left( I_k \otimes P_2 \right) \]

\[ = (P_k \otimes I_2) \left( P_{k-1} \otimes P_2 \otimes I_1 \right) \left( I_k \otimes P_2 \right) \]

\[ = (P_k \otimes I_2) \left( P_{k-1} \otimes P_2 \otimes I_1 \right) \left( I_k \otimes P_2 \right) \]

\[ = (P_k \otimes I_2) \left( P_{k-1} \otimes P_2 \otimes I_1 \right) \left( I_k \otimes P_2 \right) \]

\[ = (P_k \otimes I_2) \left( P_{k-1} \otimes P_2 \otimes I_1 \right) \left( I_k \otimes P_2 \right) \]

\[ = (P_k \otimes I_2) \left( P_{k-1} \otimes P_2 \otimes I_1 \right) \left( I_k \otimes P_2 \right) \]

\[ = (P_k \otimes I_2) \left( P_{k-1} \otimes P_2 \otimes I_1 \right) \left( I_k \otimes P_2 \right) \]

\[ = (P_k \otimes I_2) \left( P_{k-1} \otimes P_2 \otimes I_1 \right) \left( I_k \otimes P_2 \right) \]

\[ = (P_k \otimes I_2) \left( P_{k-1} \otimes P_2 \otimes I_1 \right) \left( I_k \otimes P_2 \right) \]

\[ = (P_k \otimes I_2) \left( P_{k-1} \otimes P_2 \otimes I_1 \right) \left( I_k \otimes P_2 \right) \]

\[ = (P_k \otimes I_2) \left( P_{k-1} \otimes P_2 \otimes I_1 \right) \left( I_k \otimes P_2 \right) \]

\[ = (P_k \otimes I_2) \left( P_{k-1} \otimes P_2 \otimes I_1 \right) \left( I_k \otimes P_2 \right) \]

\[ = (P_k \otimes I_2) \left( P_{k-1} \otimes P_2 \otimes I_1 \right) \left( I_k \otimes P_2 \right) \]

\[ = (P_k \otimes I_2) \left( P_{k-1} \otimes P_2 \otimes I_1 \right) \left( I_k \otimes P_2 \right) \]

\[ = (P_k \otimes I_2) \left( P_{k-1} \otimes P_2 \otimes I_1 \right) \left( I_k \otimes P_2 \right) \]

\[ = (P_k \otimes I_2) \left( P_{k-1} \otimes P_2 \otimes I_1 \right) \left( I_k \otimes P_2 \right) \]

\[ = (P_k \otimes I_2) \left( P_{k-1} \otimes P_2 \otimes I_1 \right) \left( I_k \otimes P_2 \right) \]

\[ = (P_k \otimes I_2) \left( P_{k-1} \otimes P_2 \otimes I_1 \right) \left( I_k \otimes P_2 \right) \]

\[ = (P_k \otimes I_2) \left( P_{k-1} \otimes P_2 \otimes I_1 \right) \left( I_k \otimes P_2 \right) \]
Proof by induction: True by definition for $k = 1$. Assume true for $k + 1$ for $n \geq k + 2$:

$$P_{n+1}^k = P_n^k P_n = (I_1 \otimes P_{n-1}^k) (P_{k+1}^R \otimes I_n) (I_{k+1} \otimes P_{n-k-1}) (P_{k+2} \otimes I_n)$$

$$= (I_1 \otimes P_{n-1}^k) (I_{k+1} \otimes P_{n-k-1}) (P_{k+1}^R \otimes I_n) (P_{k+2} \otimes I_n)$$

$$= (I_1 \otimes P_{n-1}^k) (I_{k+1} \otimes P_{n-k-1}) (P_{k+1}^R \otimes I_{k+1}) (P_{k+2} \otimes I_{n-k-2})$$

Corollary: $P_n^n = I_n$.

Proof by induction: $P_1 = I_1$. Assume true for $n$ prove for $n + 1$:

$$P_{n+1} = P_{n+1}^n P_{n+1} = (I_1 \otimes P_{n-1}^n) (P_{k+1}^R \otimes I_n) P_{n+1} = I_{n+1} P_{n+1}^R P_{n+1}$$

Note that $P_n^n = I_n$ implies $P_{n-1}^n = P_n^R$.

Lemma 3: $R_n = (P_{k+1}^R \otimes I_{n-k-2}) (I_1 \otimes R_{n-k})$ for $n - 1 \geq k \geq 0$

Proof by induction: Trivial for $k = 0$, true by definition for $k = 1$. Assume true for all $n \geq k' + 1$ where $k \geq k' \geq 0$, prove for $k + 1$ for all $n \geq k + 2$:

$$R_n = (P_{k+1}^R \otimes I_{n-k-2}) (I_1 \otimes R_{n-k})$$

$$= (P_{k+1}^R \otimes I_{n-k-2}) (I_1 \otimes P_{n-k} \otimes I_n) (I_1 \otimes R_{n-k-1})$$

$$= (P_{k+1}^R \otimes I_{n-k-2}) (I_1 \otimes R_{n-k-1})$$

Lemma 4: $P_{n+1} (R_1 \otimes I_n) = (I_1 \otimes R_{n+1}) P_{n+1}$ for $n \geq 1$

Proof by induction:

for $n = 1$, $P_2 (R_1 \otimes I_1) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} = (I_1 \otimes R_1) P_2$

Assume true for $n$, prove for $n + 1$:

$$P_{n+2} (R_1 \otimes I_{n+1}) = (I_n \otimes P_2) (P_{n+1} \otimes I_1) (R_1 \otimes I_{n+1})$$

$$= (I_n \otimes P_2) (I_n \otimes R_1 \otimes I_1) (P_{n+1} \otimes I_1)$$

$$= (I_{n+2} \otimes R_1) (I_n \otimes P_2) (P_{n+1} \otimes I_1)$$

$$= (I_{n+2} \otimes R_1) P_{n+2}$$
Theorem 2: $R_n^k = (P_{k-1} \otimes I_{n-k-1}) (I_1 \otimes R^{k-1}_{n-1})$ for $n - 1 \geq k \geq 1$

Proof by induction: True by definition for $k = 1$. Assume true for $k$, prove for $k + 1$ for $n \geq k + 2$:

\[
R_{n+1}^{k+1} = R_n R_n^k = (P_{k+2}^T \otimes I_{n-k-2}) (I_{k+1} \otimes R_{n-k-1}) (P_{k-1} \otimes I_{n-k-1}) (I_1 \otimes R_{n-1}^k)
\]
\[
= (P_{k+2} \otimes I_{n-k-2}) (P_{k+1} \otimes I_{n-k-2}) (P_{k-1} \otimes I_{n-k-1}) (I_{k+1} \otimes R_{n-k-1}) (I_1 \otimes R_{n-1}^k)
\]
\[
= (P_{k+2} \otimes I_{n-k-2}) (I_1 \otimes P_{k+2}^T \otimes I_{n-k-2}) (P_{k-1} \otimes I_{n-k-1}) (I_{k+1} \otimes R_{n-k-1}) (I_1 \otimes R_{n-1}^k)
\]
\[
= (P_{k+2} \otimes I_{n-k-2}) (I_1 \otimes (P_{k+2}^T \otimes I_{n-k-2}) (I_k \otimes R_{n-k-1}) (I_1 \otimes R_{n-1}^k)
\]
\[
= (P_{k+2} \otimes I_{n-k-2}) (I_1 \otimes R_{n-1}^k) (I_1 \otimes R_{n-1}^k)
\]
\[
= (P_{k+2} \otimes I_{n-k-2}) (I_1 \otimes R_{n-1}^k)
\]

Corollary: $R_n^n = H_n$

Proof by induction: $R_1^1 = H_1$. Assume true for $n$, prove for $n + 1$:

\[
R_{n+1}^{n+1} = R_{n+1} R_{n+1}^1 = R_{n+1} P_{n+1} (I_1 \otimes R_{n}^n)
\]
\[
= P_{n+1}^T (I_n \otimes R_1) P_{n+1} (I_1 \otimes H_n)
\]
\[
= P_{n+1}^T P_{n+1} (H_1 \otimes I_n) (I_1 \otimes H_n)
\]
\[
= H_{n+1}
\]

Theorem 3: $P_n^k R_n^k = I_{n-k} \otimes H_k$ for $n \geq k \geq 1$

Proof by induction: $P_1^1 R_1^1 = I_1, H_1 = H_1$. Assume true for $P_n^k R_n^k$, prove for $P_{n+1}^k R_{n+1}^k$:

\[
P_{n+1}^k R_{n+1}^k = (I_1 \otimes P_{n+1}^k) (P_{k+1}^T \otimes I_{n-k}) (P_{k+1} \otimes I_{n-k}) (I_1 \otimes R_{n}^k)
\]
\[
= I_1 \otimes P_{n+1}^k R_{n+1}^k = I_1 \otimes I_{n-k} \otimes H_k = I_{n+1-k} \otimes H_k
\]

and prove for $P_{n+1}^k R_{n+1}^{k+1}$:

\[
P_{n+1}^{k+1} R_{n+1}^{k+1} = P_{n+1} (I_{n+1-k} \otimes H_k) R_{n+1} = P_{n+1} (I_{n+1-k} \otimes H_k) P_{n+1}^T (I_n \otimes R_1)
\]
\[
= (I_{n-k} \otimes P_{n+1}^T) (P_{n+1} \otimes I_{n-k}) (I_{n+1-k} \otimes H_k) (I_1 \otimes P_{n+1}^T) (I_n \otimes R_1)
\]
\[
= (I_{n-k} \otimes P_{n+1}^T (I_1 \otimes H_k) P_{n+1}^T) (I_n \otimes R_1)
\]
\[
= I_{n-k} \otimes P_{n+1}^T (I_1 \otimes H_k) P_{n+1}^T (I_n \otimes R_1)
\]
\[
= I_{n-k} \otimes P_{n+1}^T (P_{n+1}^T R_{n+1}^k) R_{n+1}
\]
\[
= I_{n-k} \otimes P_{n+1}^T R_{n+1}^{k+1} = I_{n-k} \otimes H_{k+1}
\]

Thus we see that if we can build a machine which consists of $n$ stages, the $i$th one of which, $1 \leq i \leq n$, performs the operation $P_n^i R_n^i P_n^{i-1} x$, cascading these $n$ stages would give the desired operation of

\[
y = P_n^n R_n^1 (P_n^{n-1})^T P_n^{n-1} R_n P_n^{n-2} \cdots P_n^1 R_n P_n^0 P_n^0 x
\]
\[
= P_n^n P_n^k x = R_n^n x = H_n x
\]
For an example of the second stage operation for a length 8 code, see Fig. 9. One possible mechanization of a machine to accomplish the job of the ith stage is shown in Fig. 10, where \( w_{i-1} \) is the output of the ith stage of a binary counter which is pulsed every word time (\( \tau \)). Thus \( w_{i-1} \) changes states every \( 2^{i-1} \) word times, and \( w_{i-1} \) is time to go true as the first component of \( P_{n-1} R_{k-1} x \) appears at the ith stage.

Since we add \( 2^n \) binary numbers of \( q \) bits each, the digital word length must be \( m \geq q + n \). Since symbols are received at the rate of \( 1/\tau \) per second, the decoder must operate at \( s = m/\tau \) bits per second. If we take both \( m \) and \( s \) to be fixed, the data rate which the decoder can handle for an orthogonal code of length \( 2^n \) is \( r = (ms)/(m.2^n) \). For example, letting \( n = q = 7, s = 10^6 \) we have \( r > 35,000 \) data bits per second.

\[
P_3^T R_3^T =
\begin{bmatrix}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

This decoder has several advantages. Since every component of \( y \) involves all of the \( 2^n \) components of \( x \), any decoder must have at least \( 2^n - 1 \) words of memory. This decoder involves

\[
\sum_{i=1}^{n} 2^{i-1} = \sum_{i=0}^{n-1} 2^i = 2^n - 1
\]

words of memory. As is shown in Theorem 3, a decoder of \( N \) stages will decode any code for which \( N \geq n \geq 1 \), and, further, if it is desired to expand the decoder to the case of \( N + 1 \), no redesign is needed. To accomplish the expansion, simply add one more decoding stage and one more flip-flop to the \( w \) counter. The final advantage is the previously mentioned one of being able to accommodate quite high data rates.

Fig. 9. Second stage for 8/3 Code

Fig. 10. ith stage of decoder

References


References (Cont'd)


