CHAPTER 3.

CONSERVATIVE CODE FOR MULTIPLE ACCESS CHANNELS

3.1. General

The family of conservative codes is a new coding scheme for preserving the time integrity of serial communication over a multiple access channel. These codes are characterized by a constant number of transitions in each codeword and a known delimiting transition (rising or falling edge) at the end of each codeword ([Ofek87a], [Ofek87b]).

The conservative code enables a receiver and its serial–to–parallel converter to operate without a training period (preamble), i.e., no phase–locked loop is used for clock recovery. Instead, very accurate delay lines are used at the receiving side for decoding and for serial–to–parallel conversion. At very high bandwidth, delay lines can be accurately realized by using transmission lines or wave guides.

The main objective of this code is to preserve timing information, as opposed to codes which preserve data integrity. In fact, these two coding objectives can be combined by a two–level transformation, as shown in Figure 3.1. At the transmitter side, the error–detecting/correcting transformation is first applied, and then, the conservative code is used for adding the timing information. At the receiver side, the conservative code is used for the decoding and parallel–to–serial conversion, and then, the error–detecting/correcting function is applied. These two levels of encoding/decoding correspond to the first two levels of the communication protocol: the conservative

transformation is part of the physical level interface, while the error–detecting/correcting codes are part of the link level protocol.

In most existing designs the receiver's clock is explicitly recovered from the serial codewords with a phase–locked loop. This clock is then used for decoding and serial–to–parallel conversion. Hence, the major requirements of the encoding/decoding scheme are to ensure a sufficient number of transitions. Common examples for this kind of code are Manchester, biphase, 4B/5B, and others (see [WiFr83], [Sore84], [Seve80], [Mang83], [Lacr84] and [Joly84]). None of the existing coding schemes have a fixed ratio of total number of transitions to the total number of bits. Therefore, it is more difficult to decode and to repartition the serial information into bytes and words without a phase–locked loop.

Two additional constraints are imposed on the conservative encoding scheme: (i) balancing each codeword, i.e., making the number of zeros and ones equal, and (ii) limiting the maximum run–lengths at the high and low levels, i.e., the maximum level duration is limited. These two restrictions limit the DC shift of the receiver, and thereby

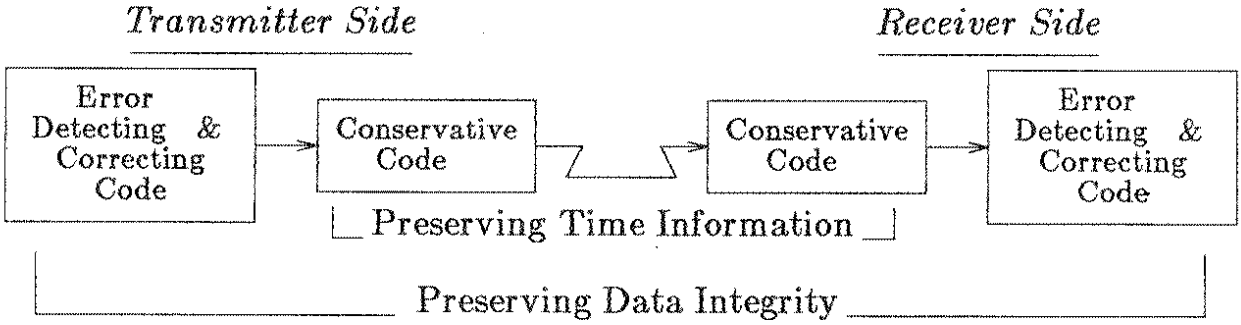

Figure 3.1: The Encoding/Decoding Protocols Levels

increase the decoding reliability and decrease the receiver complexity. It is shown that both constraints can be imposed simultaneously without a significant degradation in encoding efficiency.

The family of conservative codes is a feasible alternative to other known codes. Applying the additional constraints of balancing and limited run–length helps to simplify and improve the encoder and decoder design. The balancing constraint reduces the number of variables of the encoding/decoding Boolean functions from n to $n/2$. The maximum run–length constraint can reduce the time delay needed for realizing the decoding and encoding of Boolean functions.

3.2. Description of the Conservative Code

The family of conservative codes is defined on a block of n bits (let n be even) $a_1,a_2,.....,a_n$, with exactly $n/2$ transitions in every codeword. If a_i can be either -1 or $+1$, then the encoding is defined by

$$\sum_{i=1}^{i=n-1} a_i a_{i+1} = +1$$

and

$$a_n a_{n+1} = -1, \quad or \quad a_n = 1 \quad \text{and} \quad a_{n+1} = a_0 = -1$$

The a_{n+1} is the first bit (a_0) of the next codeword. In the more general case, for b transitions per codeword, $b \leq n$, and b is even,

$$\sum_{i=1}^{i=n-1} a_i a_{i+1} = n - 2b + 1$$

and the known delimiter always occurs at the end of the n^{th} bit–cell of each codeword. If b is odd, then every other codeword is complemented bit–by–bit, and the above equa-

tion holds. The transitions are used at the receiver side to count how many bits have arrived and to clock them in parallel into an array of shift registers, with the codeword's delimiting transitions.

The conservative encoding scheme maps l–bit data words onto n–bit codewords. For n even, b maximizes the different possible codewords when $b = \frac{n}{2}$. Given n and l, the number of possible different conservative codewords is computed as follows: each codeword c can be mapped uniquely to c' such that if there is a transition at the end of the bit cell, then this bit is mapped to one, and if there is no transition at the end of the bit cell this bit is mapped to zero. So, the mapped codeword c' has exactly $b-1$ ones which are arbitrarily placed in c', and the b^{th} one is always after the n^{th} bit cell. Thus, the number of different codewords is $\binom{n-1}{b-1}$, and the values of l, n and b satisfy the following inequality:

$$2^l < \frac{(n-1)!}{(n-b)!(b-1)!}$$

The efficiency of the encoding, μ, is the ratio of l to n.

If n is exactly divisible by four, then the codeword's bit counter need use only the rising edges or the falling edges. In this case, the counter counts to $\frac{n}{4}$ in order to determine the delimiting transition of a codeword. If n is not exactly divisible by four, the delimiting transition must be determined with a counter that uses both edges, which is more complex and can reduce the digital electronic bandwidth. In the following discussion it is assumed that $n = 4k$.

3.2.1. The 1–2 PWM code

The 1–2 pulse–width–modulation (PWM) code is a special case of the conservative scheme, with exactly one transition for every bit. In this scheme, zero is encoded as a level for one clock period, and one is encoded as a level for two clock periods, as shown in Figure 3.2. The decoding circuitry is very simple, as shown in Figure 3.3. One shift register is used for recording the odd bits, and the other shift register for recording the even bits. The outputs of the two registers are combined into one parallel word. Since the transmission length depends on the actual data, and not only on the number of bits in each packet, the baud rate is not constant. The efficiency of this encoding averages 67%, and in worst case is only 50%. If the transmission is slotted (TDMA), then one must assume the worst case efficiency of 50%.

3.2.2. The 8B/12B conservative code

Another example is encoding 8 bits of data into a 12–bit codeword (8B/12B conservative scheme). Every codeword has (i) 6 transitions, (ii) the 12^{th} bit is always one, and (iii) the first bit is always zero. Hence, every codeword has a falling edge as its delimiter. The number of bits is increased by 50%, and the encoding efficiency is 67%.

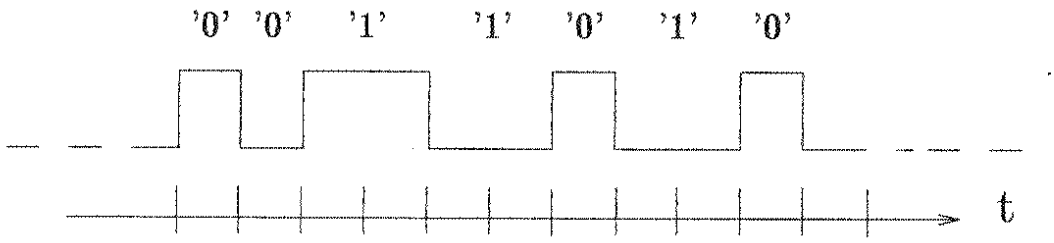

Figure 3.2: 1–2 PWM Encoding Scheme

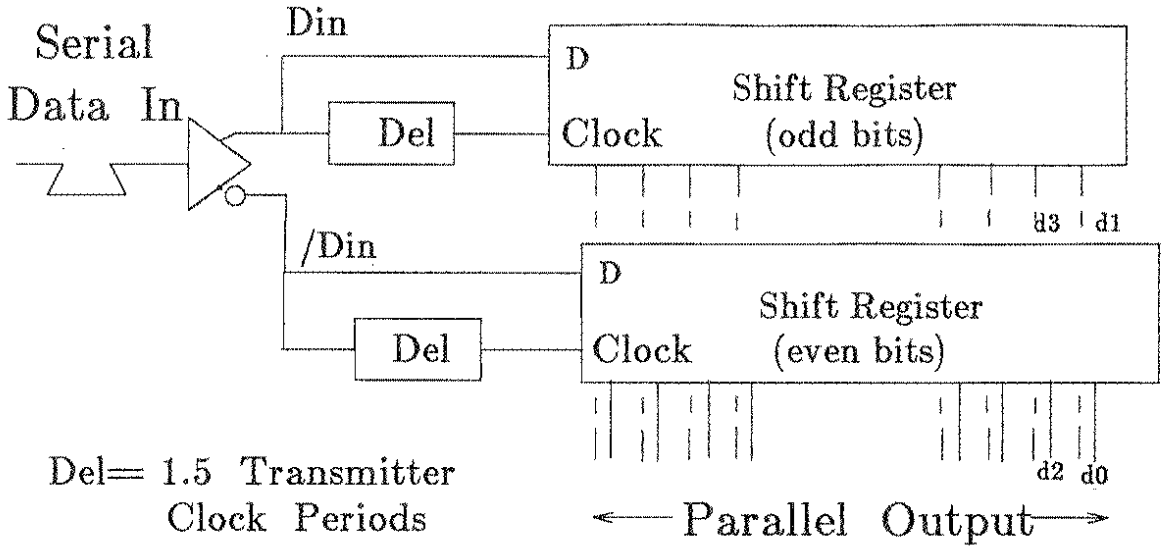

Figure 3.3: The 1–2 PWM Decoder

Eight bits of data need 256 different 12–bit codewords. The total number of different possible codewords is $\binom{11}{5} = \frac{11!}{5!6!} = 462$, which is greater than 256.

Higher–order encoding is possible, e.g., 12–bit data into a 16–bit code (12B/16B conservative scheme), or 16–bit data into a 20–bit code (16B/20B conservative scheme), with respective efficiencies of 75% and 80%.

3.3. Analysis of the Conservative Code

For practical purposes, two additional constraints are imposed on the encoding. First, limited run–length, i.e., the maximum duration of the serial signal at a high or low level, is limited to m bits. Second, the codeword is balanced, such that each codeword has the same number of low and high bits.

The main use of the conservative encoding scheme is in very high bandwidth fiber optic networks. Since fiber optic receivers are usually ac–coupled, using capacitors, an excessive dc level shift can cause errors in the interpretation of serial data. Also, the clock information encoded into the serial data must be accurately interpreted. The above constraints limit the dc shift and, therefore, increase the decoding reliability and decrease the optical receiver complexity. It will be shown that both constraints can be simultaneously imposed without a significant efficiency degradation.

Balanced block encoding schemes have been proposed by [WiFr83], [Knut86] and [Leis84]. Some other almost balanced codes are described by [Joly84], [Mang83], [Lacr84]. But none of these attempts to conserve the number of transitions in each codeword. Limited run–length coding is treated in [Fran70], [Fran82] and [HoOs75].

3.3.1. Analysis

The analysis proceeds in two steps: first, the total number of different conservative codewords of length n under the above constraints is computed, then the efficiency is computed.

Figure 3.4 shows the model for computing the number of different codewords, as a square wave with $2k$–transitions ($b=2k$), i.e., k bits of high level and k bits of low level. All the high and low levels can be extended by a total of $2k$ bit periods. If e_i is the amount each high or low level is extended, then

$$e_1 + e_2 + ... + e_{2k} = 2k,$$

with $e_i \in 0, 1, 2, \cdots, 2k$.

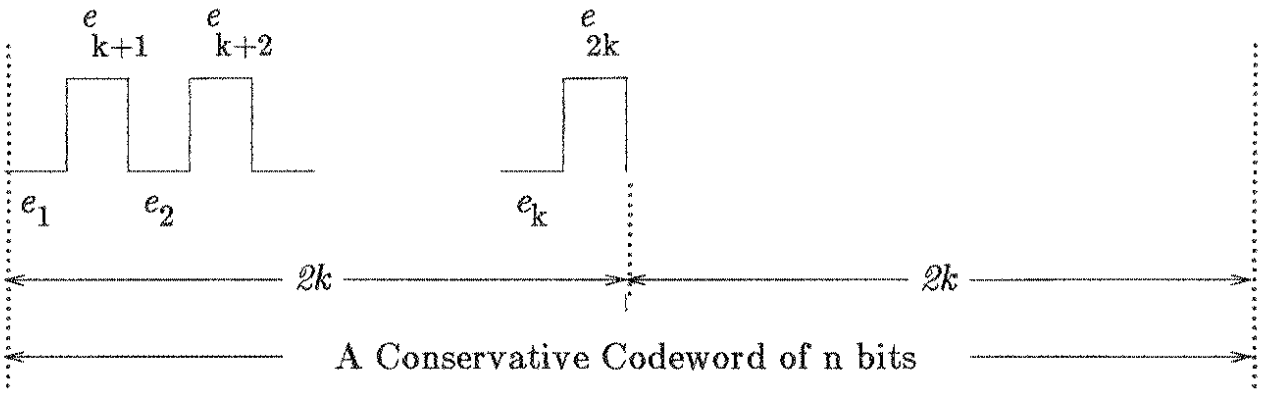

Figure 3.4: The Analysis Model

Thus, with no additional constraints, the total number of different conservative codewords is equal to the number of different integer solutions of the above equation:

$$C(n) = \binom{4k-1}{2k-1} = \binom{n-1}{\frac{n}{2}-1} = \frac{(n-1)!}{(2k-1)!2k!}$$

3.3.2. Conservative/balanced encoding

Under the balancing constraints, the high levels and the low levels are independently extended by k bit periods independently. Thus, the number of different low–level duration arrangements is equal to the number of different integer solutions of the following equation:

$$e_1+e_2+e_3+\ldots+e_k=k$$

with $e_i \in 0, 1, 2, \cdots, k$, which is:

$$CB'(n) = \binom{2k-1}{k-1}$$

Since the high and low levels can be arranged independently, the total number of possible arrangements of a conservative/balanced codeword of length n is

$$CB(n) = (CB'(n))^2 = \binom{2k-1}{k-1}^2 = \left[\frac{(2k-1)!}{(k-1)!k!}\right]^2$$

Figure 3.5 describes the encoding efficiency $\mu(n)$ as a function of n, for the two cases of conservative and conservative/balanced codes.

$$\mu_C(n) = 100\frac{\lfloor \log_2 C(n) \rfloor}{n}$$

$$\mu_{CB}(n) = 100\frac{\lfloor \log_2 CB(n) \rfloor}{n}$$

It is apparent that the efficiency of the conservative/balanced encoding is about 80% for codewords larger than 32 bits. Since the high– and low–level encodings are independent, it is possible to realize the encoder by using two look–up tables of $n/2$ address lines ($2^{n/2}$ entries); while the realization of a conservative encoding with a look–up table requires n address lines (2^n entries). Therefore, if the maximum table size which is reasonable to construct has 16 address lines, the efficiency of the conservative/balanced code is 78%, and is better than the 75% efficiency of the conservative encoding.

3.3.3. Encoding with a limited run–length

This section analyzes the efficiency of the conservative code with limited run–length of m at a high or a low level. The problem can be expressed as follows: given the above square–wave form with $2k$ transitions, all the high and low levels are extended by a total of $2k$ bit periods not exceeding $m-1$ at each continuous level. If e_i is the amount each high or low level is extended ($e_i \in \{0, 1, 2, .. , m-1\}$, $m-1 \leq 2k$), then the number of different arrangements is equal to the number of different integer solutions of the following equation:

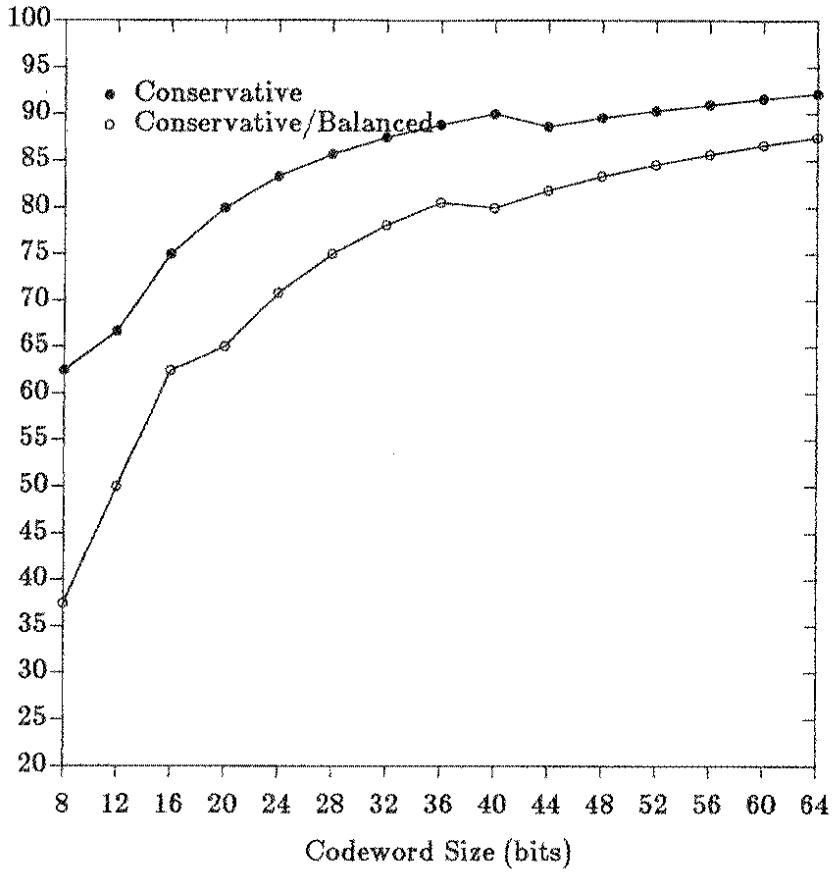

Figure 3.5: Conservative and Conservative/Balanced

$$e_1+e_2+e_3+...+e_{2k} = 2k$$

which is the coefficient of x^{2k} in the polynomial expansion of the following generating function:

$$h(x)=(1+x+x^2+x^3+..+x^{m-1})^{2k}$$

The coefficient is expressed as the function *CR(n,m)*:

$$CR(n,m) = \sum_{i=0}^{i \le 2k/m} (-1)^i \binom{2k}{i} \binom{n-1-mi}{2k-mi}$$

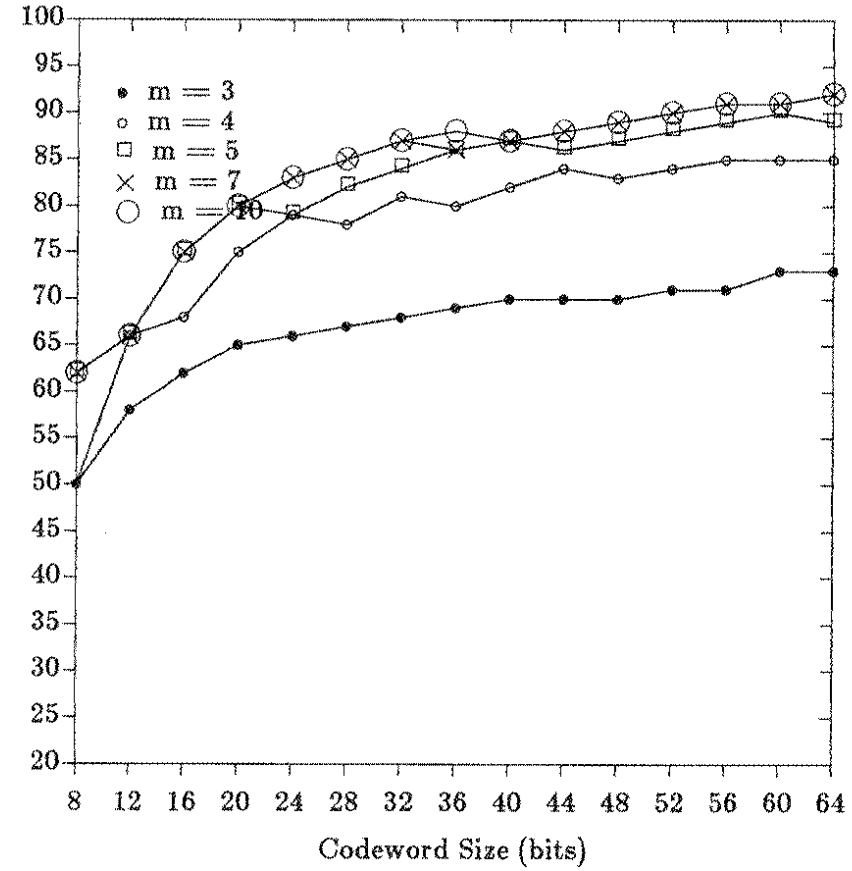

Figure 3.6: Encoding with a Limited Run–length m

Figure 3.6 shows the encoding efficiency $\mu(n)$ as a function of n, for conservative codes with a limited run–length of m.

$$\mu_{CR}(n) = 100 \frac{\left\lfloor \log_2 CR(n,m) \right\rfloor}{n}$$

From Figure 3.6 it is apparent that the encoding efficiency is little improved for a run–length greater than 4.

3.3.4. Balanced encoding with limited run–length

In this section both constraints are imposed on the conservative code. Formally, the code has the following constraints: (i) $n/2$ transitions, (ii) $n/2$ ones, (iii) a falling edge as its delimiter, and (iv) maximum run–length at the high or low level of m.

This counting problem can be solved independently for the high and low levels. The number of different arrangements for the high (or low) level is the coefficient of x^k in the polynomial expansion of the following generating function *($m-1 \leq k$)*:

$$h(x)=(1+x+x^2+x^3+..+x^{m-1})^k$$

Raising this coefficient to the power 2 will produce the total number of possible arrangements, which is expressed in the function $CBR(n,m)$:

$$CBR(n,m) = \left[\sum_{i=0}^{i \leq k/m} (-1)^i \binom{k}{i} \binom{2k-1-mi}{k-mi} \right]^2$$

Figure 3.7 describes the encoding efficiency $\mu(n)$ as the function of n for conservative/balanced codes with limited run–length of m.

$$\mu_{CBR}(n)=100\frac{\left\lfloor \log_2 CBR(n,m) \right\rfloor}{n}$$

Again, the encoding efficiency is little improved for a run–length greater than 4, and that applying both constraints changes the encoding efficiency very little.

3.3.5. Collision detection of the conservative/balanced code

The star topology of the net guarantees that if a collision has occurred, it can be seen by all the net's nodes for **any packet length**. The following design enables the nodes to detect and diagnose collisions, by encoding the data with the

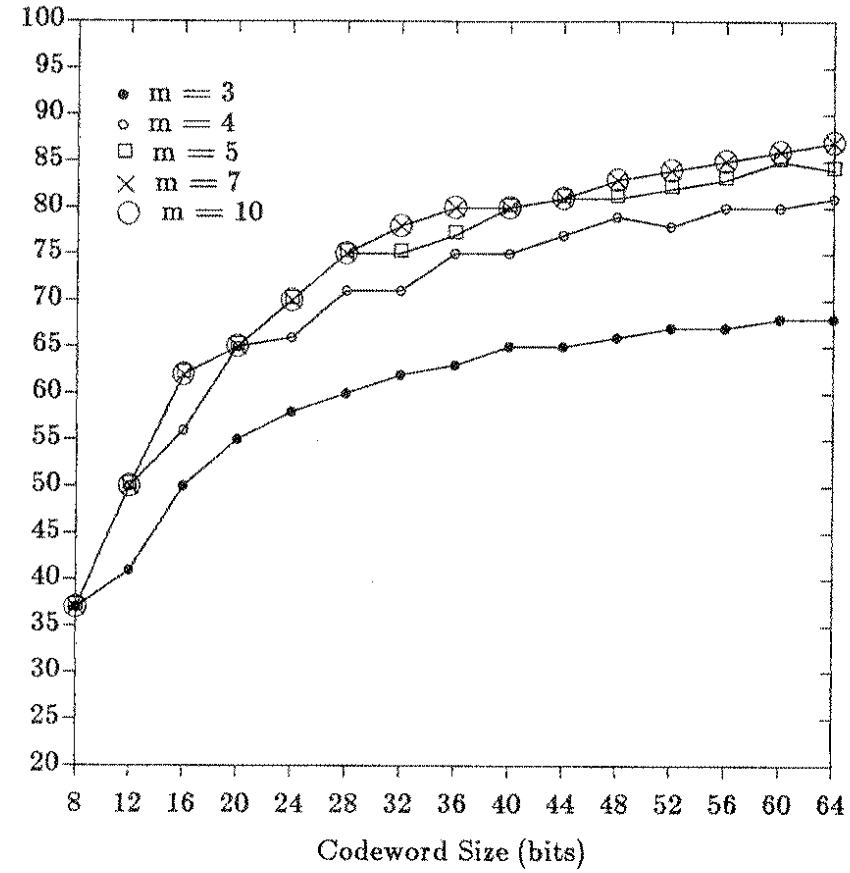

Figure 3.7: Conservative/Balanced with Run–length m

conservative/balanced code. Thus, each codeword has a fixed number of transitions, and an equal number of zeros and ones. After an optical collision the high–level duration is longer and the low–level duration is shorter. Hence, the codeword is not balanced anymore. A simple combinational network can detect the nonbalanced codeword, as shown in Figure 3.8, and then indicate a collision or an error. The error–

detection circuit counts the number of ones in a codeword of n bits. If the number of ones is larger than $\frac{n}{2}$, then it is probable that a collision has occurred. If the number of ones is less than $\frac{n}{2}$, then the failure is probably due to other causes. If the number of ones is exactly $\frac{n}{2}$, then it is a legal codeword. This simple error–detection mechanism is very effective, and uses the inherent properties of the conservative/balanced encoding scheme.

3.4. Generating the Look–up Table for the Code

In this section simple methods for generating the look–up table for encoding and decoding under the various constraints are presented.

3.4.1. Conservative code with no constraints

The transformations of the l–bit data word into an n–bit codeword and back can be done by using one look–up table for the encoder and one for the decoder. These tables are realized by two combinational or sequential networks, i.e., ROM or finite state machine.

Let U be the data word, $U = (u_1, u_2, \cdots, u_l)$, and C be the codeword, $C = (c_1, c_2, \cdots, c_n)$. Let b be the number of transitions in every codeword.

First, all possible binary data words of length l are listed in their binary order. The size of this list is 2^l. As data words we select a subset of the above list, since not all possible may be used, it is assumed that the size of data word subset is $L_{DATA} \leq 2^l$.

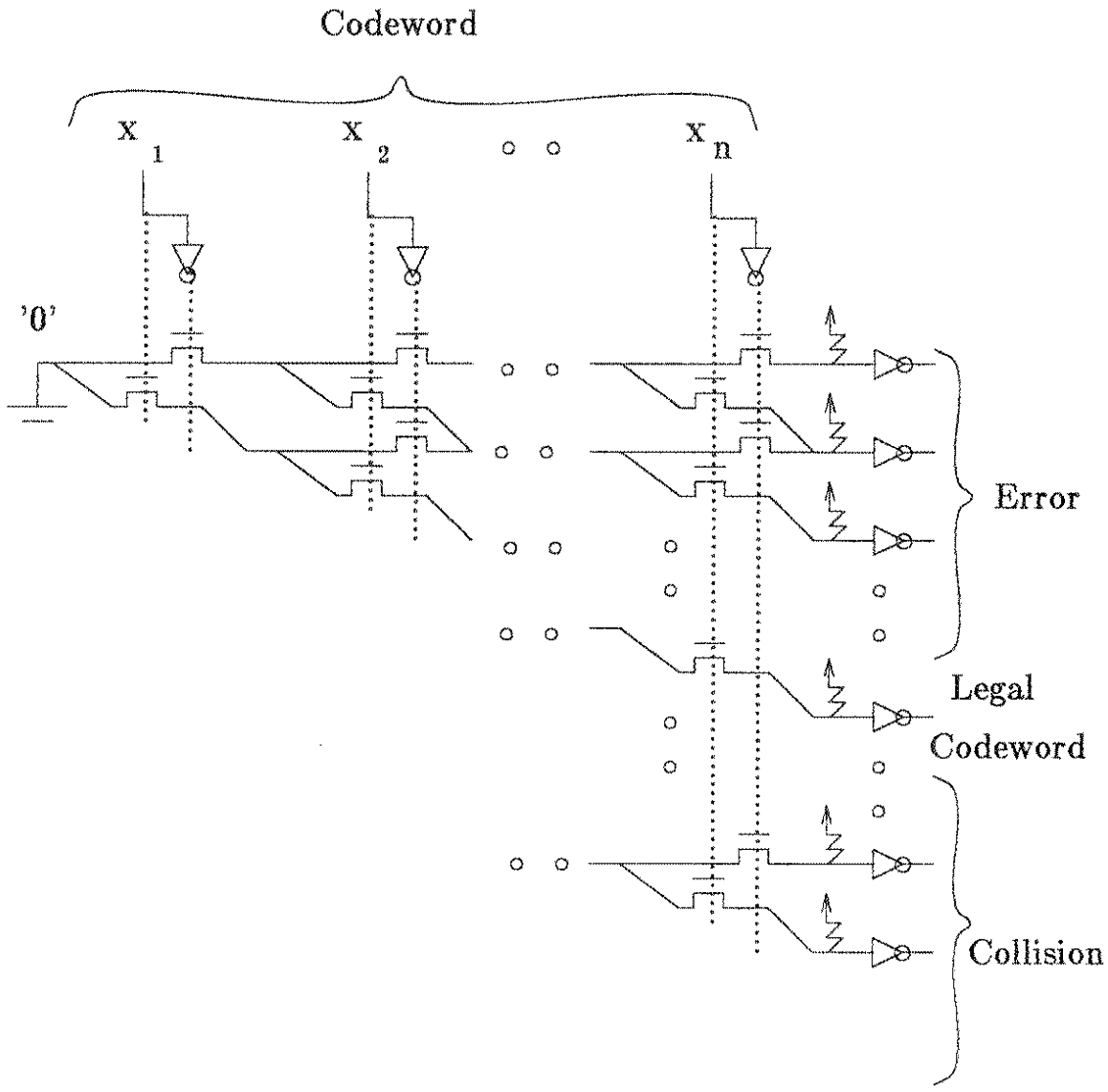

Figure 3.8: Collision and Error Detection for Conservative/Balanced Code

The list of all possible conservative codewords is generated in the following steps: Step 1 – a line of length n is divided into n segments. There are $n-1$ possible positions for $b-1$ transitions, i.e., a transition at the end of the first line segment, or 2^{nd} segment, and so on. The last transition that can be selected is at the end of the $n-1^{th}$ line segment. The b^{th} transition is the delimiter, and is always at the end of the n^{th}

line segment. Thus, $b-1$ transitions are selected from $n-1$ positions in the codeword. Let $\{1, 2, 3, \ldots, n-1\}$ be the set of different transition positions in the codeword. Then all possible subsets of length $b-1$ are listed in a **lexicographical order**. There are $\binom{n-1}{b-1}$ different subsets.

Step 2 – from the above list of subsets, the list of all different codewords is generated. Using the following procedure each subset $\{s_j, 1 \leq j \leq b-1\}$ is mapped uniquely into a conservative codeword.

Note that if, the number of transitions (b) is even, then there are two equivalent cases: (i) the first bit is always zero, and (ii) the first bit is always one. If b is odd then the first bit alternates between zero and one. The following procedure generates codewords such that the first bit is always zero. If the first bit would be one, then a complement list of conservative codewords would be obtained.

$v = 0$

$c_1 = 0$

DO i =2 to n

IF $i-1 \in \{s_1, \ldots, s_{b-1}\}$,

THEN $v = v+1 \bmod 2$

$c_i = v$

END

Note that if b is odd, then, while transmitting a sequence of codewords, all even codewords should be inverted (or complemented) bit–by–bit. Thus, the delimiting transition would be preserved. At the receiving side, after the serial–to–parallel conversion, these codewords should be inverted back to the original codeword.

In general, a subset of codewords is selected from this list, with a size of $L_{CON} \leq \binom{n-1}{b-1}$. Clearly, for one–to–one transformations from the subset of data words to the subset of codewords, $L_{DATA} \leq L_{CON} = p$.

The encoding table is a one–to–one mapping from the data subset to the codeword subset; the decoding table is the reverse. All the unused codewords in the codeword list may be used for an error signal or for control.

Note that the one–to–one mapping is arbitrary, and there are $p!$ different possible mappings.

In the following example l=4, n=7, and b=4, and a possible one–to–one mapping is

Data Word	Codeword
0000	0101111
0001	0100111
0010	0100011
0011	0100001
0100	0110111
0101	0110011
0110	0110001
0111	0111011
1000	0111001
1001	0111101
1010	0010111
1011	0010011
1100	0010001
1101	0011011
1110	0011001
1111	0011101
Unused	0001011
Unused	0001001
Unused	0001101
Unused	0000101

The unused codewords may be used for error detection or communication control.

3.4.2. Conservative balanced code

Generating the mapping table for the conservative balanced code is done as described before, with an additional step that eliminates the nonbalanced codewords from the codeword list, as described in the following procedure:

$r = 0$, $s = 0$

DO $i = 1$ to n

IF $c_i = 1$,

THEN $r = r + 1$

ELSE $s = s + 1$

END

IF $r - s \neq$ {one of predetermined set of values},

THEN take this codeword off the codeword list

Note that the above set of values may be zero for codewords of even length, or (1, –1) for codewords of odd length.

In the above example with $l = 4$, $n = 7$, $b = 4$, and the possible differences between the high and low level is +1 or –1, the mapping is as follows:

Data Word	Codeword
0001	0100111
0010	0100011
0101	0110011
0110	0110001
1000	0111001
1010	0010111
1011	0010011
1101	0011011
1110	0011001
1111	0011101

0000	0001011
0011	0001101

Note that the following data words are not mapped to any codeword, and therefore cannot be used: 0100, 0111, 1001, 1100.

3.4.3. Limited run–length

Generating the mapping table for the conservative code with limited run–length of m, is done as described before, with an additional step which eliminates the codewords with run–length greater than m from the codeword list, as described in the following procedure:

$r = 1$

$v = 0$

DO i=1 to $n-1$

IF $c_i = c_{i+1}$,

THEN $r = r+1$

ELSE $r = 1$

IF $r > m$,

THEN $v = 1$

END

IF $v = 1$,

THEN take this codeword off the codeword list

In the above example with l=4, n=7, b=4, and $m = 3$, the mapping is as follows:

Data Word	Codeword
0001	0100111
0010	0100011
0011	0100001
0100	0110111
0101	0110011
0110	0110001
0111	0111011
1000	0111001
1010	0010111
1011	0010011
1100	0010001
1101	0011011
1110	0011001
1111	0011101
0000	0001011
1001	0001001
Unused	0001101
Unused	0000101

3.4.4. Limited run–length with balancing

Generating the mapping table for the balanced conservative code with limited run–length of m is done as described before, with an additional step which eliminates the codewords with run–length greater than m, and unbalanced codewords from the conservative codewords list, as described in the following procedure:

$r = 0$, $s = 0$

DO $i = 1$ to n

IF $c_i = 1$,

THEN $r = r + 1$

ELSE $s = s + 1$

END

IF $r - s \neq$ {one of predetermine set of values},

THEN take this codeword off the codeword list

$r = 1$

$v = 0$

DO i=1 to $n-1$

IF $c_i = c_{i+1}$,

THEN $r = r+1$

ELSE $r = 1$

IF $r > m$,

THEN $v = 1$

END

IF $v = 1$,

THEN take this codeword off the codeword list

In the above example with l=4, n=7, b=4, and $m = 3$, the mapping is as follows:

Data Word	Codeword
0001	0100111
0010	0100011
0101	0110011
0110	0110001
1000	0111001
1010	0010111
1011	0010011
1101	0011011
1110	0011001
1111	0011101
0000	0001011
0011	0001101

Note that the following data words are not mapped to any codeword, and therefore cannot be used: 0100, 0111, 1001, 1100.