EFMPlus: THE CODING FORMAT OF THE MULTIMEDIA COMPACT DISC

Kees A. Schouhamer Immink
Philips Research Laboratories,
5656 AA Eindhoven, The Netherlands

Abstract–**We report on an alternative to Eight-to-Fourteen Modulation (EFM), called EFMPlus, which has been adopted as coding format of the MultiMedia Compact Disc proposal. The rate of the new code is 8/16, which means that a 6-7 % higher information density can be obtained. EFMPlus is the spitting image of EFM (same minimum and maximum runlength, clock content etc). Computer simulations have shown that the low-frequency content of the new code is only slightly larger than its conventional EFM counterpart.**

I Introduction

The Compact Disc (CD), introduced more than a decade ago, has become a successful medium for the distribution and storage of digital information. It is anticipated that its storage capacity, 650 MByte, will be insufficient for future graphics-intensive computer applications and high-quality digital video programs. An extension of the Compact Disc family, the MultiMedia Compact Disc (MM-CD), is a proposal for a new optical recording medium with a storage capacity five times higher than the conventional Compact Disc. The major part of the capacity increase is achieved by the use of optics, shorter laser wavelength and larger numerical aperture, that reduces the spot diameter by a factor 1.5. The track formed by the recorded pits and lands as well as the track pitch can be reduced by the same factor. The storage capacity is further increased by a complete redesign of the logical format of the disc including a more powerful error correction (CIRCPlus) and recording code (EFMPlus). The new format accords with the major application of the MultiMedia CD as a multi-Media recorder. As a result, the new format requires 30 % less overhead.

In the Compact Disc system, the Eight-to-Fourteen Modulation (EFM) code is used for transforming the digital audio bit stream into a sequence of binary symbols, called *channel bits,* which are suitable for storage on the disc [1]. Not only is the EFM coding scheme useful for the Compact Disc for which it has been designed, but it has been extensively employed in a large variety of digital audio players and home-storage products such as CD-ROM, CD-I, and MiniDisc.

The aim of our investigations was the design of a new 'EFM-like' code having a better rate than its predecessor EFM. The most important design issue is that critical parameters such as lf-content and timing should definitely not be worsened. We consider these parameters as very critical as the code's lf-content constitute an added noise signal, which has a detrimental effect on the reliability of the servos systems and the timing recovery. Obviously, the servo system is the Achilles' heel of the player as error correction is totally useless if track or clock loss occurs.

Table 1: Main parameters of MultiMedia CD.

read-out wavelength	635 nm
reference NA	0.52
disc diameter	12 cm / 8cm
disc thickness	1.2 mm
layers	single or dual layer
reference scanning speed	4 m/s
reference channel bit rate	26.6 Mbit/s
min. pit (or land) length	0.451 μm
track pitch	0.84 μm
recording code	EFMPlus
sector size	2048 bytes
error correction	CIRCPlus
max. user bit rate @ ref speed	11.2 Mbit/s

We have developed a new code, called *EFMPlus*, having properties similar to those of EFM. The strategy of EFMPlus is much more refined and more powerful than the original EFM. As a result, the rate of EFMPlus, i.e. the quotient of the numbers of bits entering and leaving the encoder, is 8:16 and it is therefore capable of recording 6 % more user information than is possible with EFM whose rate is 8:17. We start with a brief system description of the MultiMedia CD. Thereafter, we will outline the system requirements of the EFM and EFMplus codes followed by a performance comparison of the alternative codes that have been investigated.

II System description

The main parameters of the MultiMedia CD are listed in Table 1. Note that mechanical specifications such as disc thickness, outer diameter, and center hole diameter of the MM-CD are equal to the those of CD, allowing full backward compatibility. The user capacity of the disc is 3.7 GByte per layer. Blocks of 2048 user bytes are translated into 2436 bytes (2048 user + 56 sync + 24 header + 308 erco) which are are in turn translated into 2×2436 channel bits. Thus, one user bit is translated into $4872/2048 = 2.38$ channel bits. In conventional CD, an audio bit is translated into $588/192 = 3.05$ channel bits [2], and we conclude that the 'format efficiency' of

Manuscript received June 12, 1995

Table 2: Part of the EFM coding table.

Data	*Code*	*Data*	*Code*
100	01000100100010	112	10010010000010
101	00000000100010	113	00100000100010
102	01000000100010	114	01000010000010
103	00100100100010	115	00000010000010
104	01001001000010	116	00010001000010
105	10000001000010	117	00100001000010
106	10010001000010	118	01001000000010
107	10001001000010	119	00001001001000
108	01000001000010	120	10010000000010
109	00000001000010	121	10001000000010
110	00010001000010	122	01000000000010
111	00100001000010	123	00001000000010

the MultiMedia CD is improved by 29 %.

III EFM recording code

The primary purpose of recording codes is to transform the frequency spectrum of a serial data stream so that clocking can be recovered readily and AC coupling is possible. The recording code used in the Compact Disc, EFM, is a member of the family of *dc-free runlength-limited codes*. The number of sequential like symbols in a sequence is known as *runlength*. A runlength-limited sequence is a sequence of binary symbols characterized by two parameters, $T_{\min} = (d+1)$ and $T_{\max} = (k+1)$, which specify the minimum and maximum runlength, respectively, that may occur in the sequence. The parameter d controls the highest transition frequency while the parameter k ensures adequate frequency of transitions for synchronization of the read clock [2]. There are two reasons why EFM suppresses the low-frequency components. In the first place, the servo systems for track following and focusing are controlled by low-frequency signals, so that low-frequency components of the information signal could interfere with the servo-systems. The second reason is that low-frequency disturbances resulting from fingerprints on the disc can be filtered out without distorting the data signal itself.

Under EFM rules, the data bits are translated eight at a time into fourteen channel bits, with minimum runlength parameter $d = 2$ and a maximum runlength parameter $k = 10$ channel bits (this means at least 2 and at most 10 successive 'zeros' between successive 'ones'. Part of the EFM coding table is presented in Table 2, which shows the decimal representation of the 8-bit source word (left column) and its 14-bit channel representation. It should be appreciated that the codewords are described in NRZI (nonreturn-to-zero inverted) notation, which means that a 'one' represents a transition of either positive or negative polarity, and a 'zero' represents the absence of a transition. It is easily seen that at least two bits, called *merging bits*, are required to ensure that the runlength conditions continue to be satisfied when the codewords are cascaded. If the runlength is in danger of becoming too short, we choose 'zeros' for the merging bits; if it is too long we choose a 'one' for one of them. If we do this, we still retain a large measure of freedom in the choice of the merging bits. This freedom is used for minimizing the low-frequency content of the signal. In itself, *two* merging bits would be sufficient for continuing to satisfy the runlength conditions (see also later). A third merging bit is necessary, however, to give sufficient freedom for effective suppression of low-frequency content, even though it entails a loss of 6% of the information density on the disc (see also next section). The merging bits contain no information, and they are removed from the bit stream in the demodulator. Figure 1 illustrates, finally, how the merging bits are determined. Our measure of the low-frequency content is the *running digital sum* (RDS); this is the difference between the totals of pit and land lengths accumulated from the beginning of the disc. At the top, two 8-bit data words are shown. From the $d = 2$ rule, the first of the merging bits in this case must be a 'zero'. This position is marked 'x'. In the two following positions the choice is free. These are marked 'm'. The three possible choices 'xmm'='000', '010', and '001' would give rise to the patterns of pits as illustrated and

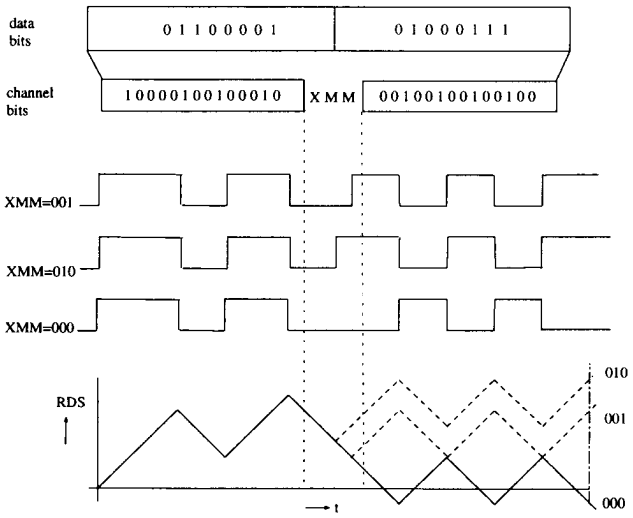


Figure 1: Strategy for minimizing the running digital sum (RDS). Eight user bits are translated into 14 channel bits. The 14 bits are merged by means of 3 merging bits in such a way that the runlength conditions continue to be satisfied. The proviso that there should be at least 2 'zeros' between 'ones' requires a 'zero' at the first merging bit position. In this case there are thus three alternatives for the merging bits: '000', '010', and '001'. The encoder chooses the alternative that gives the lowest absolute value of the RDS at the end of a new codeword, i.e. '000' in this case.

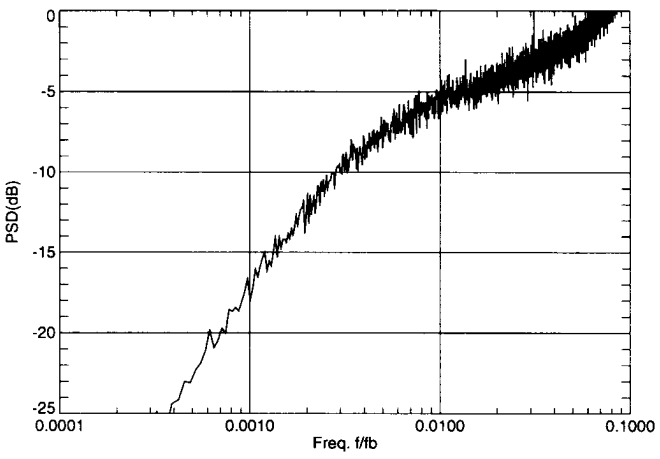


Figure 2: Spectrum of conventional EFM.

to the indicated waveform of the RDS, on the assumption that the RDS was equal to '0' at the beginning. The system now opts for the merging combination that makes the RDS at the end of the second codeword as close to zero as possible, i.e. '000' in this case. If the initial value had been -3, the merging combination '001' would have been chosen. The Power Spectral Density (PSD) function of conventional EFM has been obtained by computer simulation. Results are plotted in Figure 2. Both axes are normalized for fixed user bit rate f_b.

IV The new scheme: EFMPlus

The principle of operation of the encoder can be represented by a finite-state sequential machine with an 8-bit input, a 16-bit output, and four states which are functions of the (discrete) time. We say that the states are connected by edges, and the edges, in turn, are labeled with tags called *words*. A word in our context is a 16-bit sequence that obeys the prescribed $(d = 2, k = 10)$ constraints. Each of the four states is characterized by the type of words that enter, or leave, the given state. The states and word sets are characterized as follows.

- Words entering State 1 end with $\{0, 1\}$ trailing 'zero';
- Words entering State 2 and 3 end with $\{2, \ldots, 5\}$ trailing 'zeros';
- Words entering State 4 end with $\{6, \ldots, 9\}$ trailing 'zeros'.

The words leaving the states are chosen in such a way that the concatenation of words entering a state and those leaving that state obey the $(d = 2, k = 10)$ channel constraints. For example, words leaving State 1 start with a runlength of at least two and at most nine 'zeros'. In an analogous manner, we conclude that words leaving State 4 start with at most one 'zero'. Obviously, the sets of words leaving State 1 or 4 have no words in common. Words emerging from State 2 and 3 comply with the above runlength constraints, but they also comply with other conditions. Words leaving State 2 have been selected such that the first (msb) bit, x_1, and the thirteenth bit, x_{13}, are both equal 'zero'. In a similar fashion, words leaving State 3 are characterized by the fact that the 2-tuple $x_1 x_{13}$ does not equal '00'. The attributes of the four states defined above guarantee that any walk through the graph, stepping from state to state, produces a $(d = 2, k = 10)$-constrained sequence by reading the words tagged to the edges that connect the states. Given the above definitions, it is elementary to compute the number of edges (or words) that leave each of the four states. Let the entry t_{ij} of the 4×4 transition matrix T denote the number of words leaving state i that go to state j. Then we obtain

$$T = \begin{bmatrix} 138 & 96 & 96 & 22 \\ 145 & 90 & 90 & 27 \\ 132 & 102 & 102 & 15 \\ 164 & 113 & 113 & 25 \end{bmatrix}.$$

The number of outgoing edges, called *fan-out*, of State 1 is $138 + 96 + 96 + 22 = 352$. Similarly, the fan-out of States 2, 3, and 4 is 352, 351, and 415, respectively. Thus from each of the states at least 351 words are leaving.

We are now in the position to define an encoder graph, in short, encoder. An encoder is constructed by assigning a source word to each of the 351 edges that leave each state. Excess edges are removed. As a result, each edge in the graph has now two labels, namely a 16-bit word and a source word numbered from 0 to 350. Given the source word and the encoder state, the encoder will transmit the word tagged to the same edge as the source word at hand. It is immediate which codeword will be sent, and also which state will be next. A sequential input will define a walk in the graph, and as said, this walk generates a $(d = 2, k = 10)$-constrained sequence. As a result, the above finite-state encoder graph is a $(d = 2, k = 10)$ RLL encoder that freely accommodates 351 source words. We ignore for a while a very important item, namely whether or not, and how, an inverse operation can be found, i.e., given the received string of 16-bit words whether or not, and how, the string of source words can be uniquely decoded at the receiver's site. The encoder requires accommodation for only 256 source words. The excess, 95, words will be used for controlling the low-frequency power (see later). We will first take a closer look at the details of the encoder graph.

A Encoder operation

The encoder is defined in terms of three sets: the inputs, the outputs and the states, and two logical functions: the *output function* and the *next-state function*. The specific codeword, denoted by $\boldsymbol{x}_t$, transmitted by the encoder at

instant t is, of course, a function of the source word $\boldsymbol{b}_t$ that enters the encoder, but depends further on the particular state, s_t, of the encoder. Similarly, the "next" state at instant $t+1$ is a function of $\boldsymbol{x}_t$ and s_t. The output function $h(.)$ and the next-state function $g(.)$ can be succinctly written as

$$\boldsymbol{x}_t = h(\boldsymbol{b}_t, s_t)$$

$$s_{t+1} = g(\boldsymbol{b}_t, s_t).$$

Both the output function $h(.)$ and the next-state function $g(.)$ are described by four lists with 351 entries. A part of the output function and the next-state function is listed in Table 3. Table 3 has an entry column that describes the source (input) word i by an integer between '0' and '255'. The table also shows $h(i,s)$ the 16-bit output to a particular input i when the encoder is in one of the four states s. The words are written in NRZI notation. It is assumed that the msb bit is transmitted first. The 3rd, 5th, 7th, and 9th columns show the next state function $g(i,s)$. An example may clarify the principles of operation of the encoder. Let the encoder graph be initialized at State 1 (it will be shown in a moment, that the initial state is irrelevant for the decoding operation), and let further the source sequence be '8', '3', '4'. The response to input '8', while being in State 1, equals $h(8,1)$ = (see Table 3) '0010000010010000'. The new state becomes $g(8,1) = 3$. As a result, the response to input '3', while now being in State 3, is '0010000001001000'. In the next clock cycle, the encoder state becomes $g(3,3) = 2$. From State 2 with the input equal to '4' we find from the table that the corresponding output is $h(4,2)$ ='0010000010010000'. Note that the responses to the distinct inputs '3' and '7' while in State 1 are the same, namely '0010000001001000'. In the next section, we will describe how this ambiguity can be resolved during decoding.

B Decoder operation

From the above and inspection of the encoder table, it can be seen that each source word is represented by at least two channel representations. For example, source word '8' is represented by two codewords, namely '0010000010010000' or '1000010010000000' (see Table 3). Basically, knowledge of the encoder state at the receiver site is necessary to re-constitute the source word. This operation can be succinctly written as

$$\boldsymbol{b}_t = h^{-1}(\boldsymbol{x}_t, s_t).$$

Such state-dependent decoding is a technique that should be avoided as it usually entails a strong error propagation. In our case, the source words can be judiciously assigned in such a way to the various codewords that decoding of the channel representations can be uniquely accomplished without knowledge of the encoder state.

There is one remaining technicality that has not been discussed yet. On a few occasions, as we can observe in the table, two source words have the same channel representation(s). For example, source words '3' and '7' generate the same channel representation, '0010000001001000', when the encoder is in State 1. Evidently these words cannot be decoded by a sole observation of the 16-bit codeword. This ambiguity can be remedied by observing that the code was constructed in such a way that if the same two words do leave a given state, one of them goes to State 2 and the other to State 3. As the sets of words leaving State 2 or 3 have been chosen such that they differ in the 2-tuple formed by the symbols at positions 1 and 13, words can be uniquely decoded by observing a 16-bit codeword plus the 1st and 13th bit of the upcoming codeword. In other words,

$$\boldsymbol{b}_t = h^{-1}(\boldsymbol{x}_t, x_{t+1,1}x_{t+1,13}).$$

Under EFM rules, it suffices to observe 14 of the 17 channel bits that constitute an EFM codeword. In contrast, decoding of the new code is done by a logic array that translates (16+2) channel bits into 8 bits. Two major differences of the new code relative to EFM become apparent. Firstly, as look-ahead is employed during decoding, it is possible that two consecutive decoded bytes are in error if the channel symbol at positions 1 or 13 is in error. Secondly, a larger look-up table is required for decoding.

C Dc control

The encoder defined above can freely accommodate 351 source words. In order to make it possible to use a unique 26-bit sync word (see later), seven candidate words were barred, leaving a code size of 344. As we only need accommodation for 256 source words, the surplus 344-256=88 words can be used for minimizing the power at low frequencies, in short, the dc-control. The suppression of low-frequency components or dc-control is done in the same vein as in the EFM code, namely by controlling the running digital sum (RDS). The 88 surplus words are used as an alternative channel representation of the source words 0..87. The full encoder is described by two tables called *main* and *substitute table*, respectively (see Figure 3). The main table describes an encoder table of 256 inputs. The substitute table shows a similar table of 88 words which act as alternative representations of the source words 0,...,87 of the main table. The source words 0,...,87 can thus be represented by the designated entries of the main table or alternatively by the entries of the substitute table. A very critical part of the EFMPlus design was the process of "mating" the alternative representations. There is no systematic approach available and a simple heuristic method was used. The "mating" of the alternative representations has been done in such a way that the impact on the RDS is as large as possible and that also the polarity of the RDS contributions of the alternative representation differs. The tables were compiled in such a way that the RDS contributions of

Table 3: Part of the EFMPlus coding table.

i	$h(i,1)$, $g(i,1)$	$h(i,2)$, $g(i,2)$	$h(i,3)$, $g(i,3)$	$h(i,4)$, $g(i,4)$
0	0010000000001001, 1	0100000100100000, 2	0010000000001001, 1	0100000100100000, 2
1	0010000000010010, 1	0010000000010010, 1	1000000100100000, 3	1000000100100000, 3
2	0010000100100000, 2	0010000100100000, 2	1000000000010010, 1	1000000000010010, 1
3	0010000001001000, 2	0100010010000000, 4	0010000001001000, 2	0100010010000000, 4
4	0010000010010000, 2	0010000010010000, 2	1000000100100000, 2	1000000100100000, 2
5	0010000000100100, 2	0010000000100100, 2	1001001000000000, 4	1001001000000000, 4
6	0010000000100100, 3	0010000000100100, 3	1000100100000000, 4	1000100100000000, 4
7	0010000001001000, 3	0100000000010010, 1	0010000001001000, 3	0100000000010010, 1
8	0010000010010000, 3	0010000010010000, 3	1000010010000000, 4	1000010010000000, 4

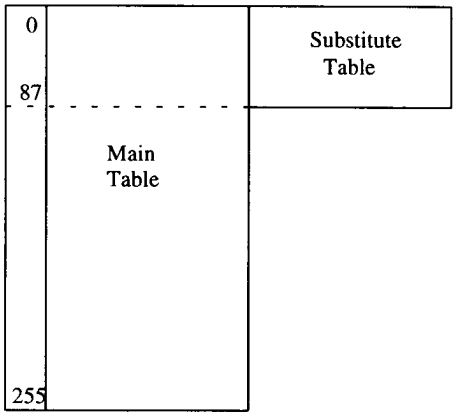


Figure 3: Block diagram of EFMPlus encoder.

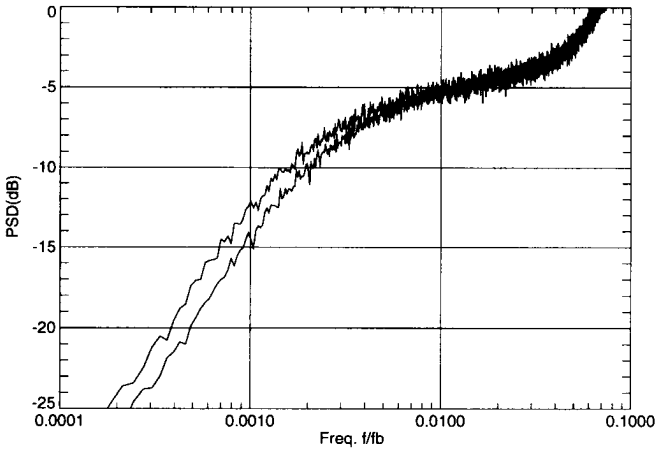


Figure 4: Spectrum of EFMPlus. The upper curve shows the spectral density without look-ahead while the lower curve shows the spectral density using two bytes look-ahead.

the first 88 entries of the main table are negative, while RDS contributions of the entries of the substitute table are predominantly positive. For source words 0,...,87 the encoder opts for that particular representation from the main table or the substitute table that minimizes the absolute value of the RDS. Note that at most only two RDS computations and comparisons per codeword have to be made, while in conventional EFM, at most four (there are at most four allowed combinations of the three merging bits) RDS computations have to be performed. Clearly, 88 out of the 256 possible source words have a means for controlling the RDS. The fact that only a part of the 256 source words has an RDS control makes the system sensitive to 'worst-case' inputs. This is not a big deal since a similar difficulty applies in EFM, which, however, seems to work satisfactorily. The power spectral density of the new code has been computed by a computer program which simulated the encoder algorithm. Results are plotted in Figure 4. A 3 dB improvement can be gained by using a look-ahead strategy, that is decisions are made on the basis of (in this case) three consecutive source bytes. The look-ahead algorithm provides a better trade-off between long and short term success.

D Sync pattern

In the MM-CD format, three distinct and unique 32-bit sync patterns are provided to mark the start of each EFMPlus frame:

S_1:	x00100000100000000000100000000001
S_2:	x00100100100000000000100000000001
S_3:	x00001000100000000000100000000001

The variable 'x' will be explained later. The essential character of this sync pattern is the occurrence of two consecutive maximum runlengths (note that a similar, but 27 bits long, sync pattern is used in conventional EFM). Just before the sync pattern starts, the new encoder is in a certain state, say s. If $s = 1$ or $s = 2$, then x is set to '0' else $x := 1$. This rule for selecting 'x' avoids the occurrence of a 'false' sync directly before the sync word. Precautions have been taken during the final compilation process of the coding tables in order to avoid

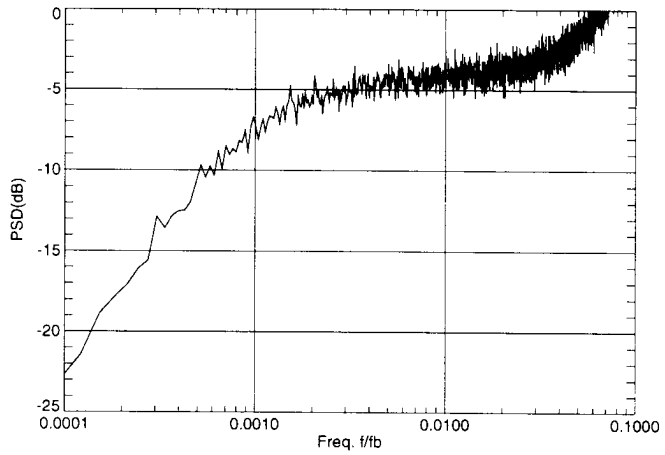


Figure 5: Spectrum of modified EFM with two merging bits.

the occurrence of the sync pattern during 'normal' transmission. After this process of discarding codewords, 344 possible source words can be accommodated. Thus we have 344-256=88 alternative representations. After emission of the sync pattern, the encoder is reset to State 1. As codewords emerging from State 1 start with at most nine 'zeros' juxtaposition of the sync pattern itself and a word generated directly following the sync pattern cannot generate a 'false' sync.

V Alternative constructions

In the next two subsections we will compare the performance of two other code constructions with that of EFMPlus. Both constructions are modifications of conventional EFM. The first one is a rate 8/16 code which uses two merging bits instead of three. The second construction offers a rate 8/15, uses only one merging bit; its merging rule is slightly more involved.

A "EFM" with two merging bits

From the previous section it is easily understood that a code with a higher rate than EFM can easily be obtained by using two merging bits instead of the usual three. The code rate would be 8/16 so that this new code would enable the storage of 6 % more information. This simple modification of EFM has a number of attractive features. Both the minimum and maximum runlength of the modified EFM would be exactly the same as those of EFM. Furthermore, decoding can be done with the same small logic circuitry. Error propagation during decoding is, as in EFM, limited to one decoded byte. On the other side of the balance is the increased low-frequency content of this alternative. The PSD function of this modification of EFM has been obtained by computer simulation. Results are plotted in Figure 5. A comparison of Figures 2 and 5 reveals that the power density at the low-frequency end has increased by 10 dB. By using a look-ahead strategy, the low-frequency content can be reduced by 5 dB.

Table 4: Basic coding table 3PM code.

Data	*Code*
000	000010
001	000100
010	010000
011	010010
100	001000
101	100000
110	100010
111	100100

B EFM alternative of rate 8/15

The merging rule of the EFM alternative of rate 8/15 is based on the so-called 3PM coding principle. The basic parameters of the 3PM (Three Position Modulation) code, which was invented by Jacoby [3] are $d = 2$, $k = 11$ and $R = 1/2$. The encoding is explained by looking at Table 4. As we may notice, the right-hand boundary position is occupied by 'zeros' in all codewords. If the juxtaposition of codewords offends the $d = 2$ condition, that is, if both the symbol at position 5 of the present codeword and the symbol at position 1 of the next codeword are 'one', the 'merging' symbol at position 6 is set to 'one' and the symbols at position 1 and 5 are set to 'zero'. The encoding logic that implements the described merging rule can be very simple. Decoding is done in a similar way. The above principle of 'merging' codewords can be applied directly to the set of 14-bit EFM words. The above merging rule requires only one merging bit and the code so designed has a rate of 8/15. As EFM words start and end with at most 8 'zeros', we simply find that the maximum runlength of this construction is $k = 17$. Decoding can be done with the EFM logic table with some extra features for recognizing the applied merging rules. By a rearrangement of the 14-bit codeword set, the maximum runlength can be reduced to 15, which is still larger than that of conventional EFM [4]. DC-control can be accomplished by observing that the merging bit can be set to 'one' if it can be uniquely observed at the decoder site that a 'normal' merging operation as described above has not been executed. It is easily seen that this is possible if the codeword ends with '100' (note that in this case the ending (lsb) bit cannot be a 'one') and the upcoming word starts with '00'. Similarly, for reasons of symmetry, if the codeword ends with '00' and the upcoming word starts with '001'. The above degree of freedom of setting a 'one' can be used to control the running digital sum of the recorded sequence [5]. Results of computer simula-

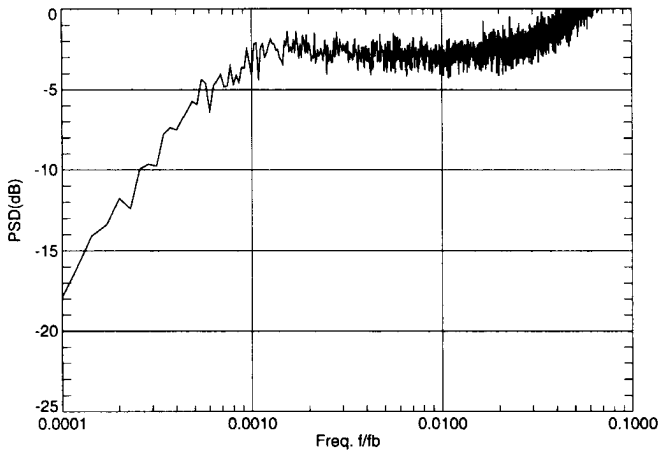


Figure 6: Spectrum of rate 8/15 'EFM-like' code.

tions are plotted in Figure 6. Though the extra 6 % of density increase that this rate 8/15 code offers, is very attractive, the increased lf-content of the code (16-18 dB above EFM level) imposes too great a risk for the servo systems and data recovery. In particular so, since typical tolerances such as defocusing and mistracking are approximately two times tighter compared with CD. A second drawback of the rate 8/15 code is that error propagation during decoding is much more severe than in the case of EFMPlus.

VI Conclusions

The MultiMedia Compact Disc is a proposal for a new optical recording medium with a storage capacity five times higher than the conventional Compact Disc. We have reported on an alternative to Eight-to-Fourteen Modulation called EFMPlus which has been adopted as the recording format in the MultiMedia Compact Disc. The strategy of EFMPlus is much more refined than the original EFM of Compact Disc. The saving in rate of EFMPlus offers a serviceable 6 % increase in capacity as compared with its predecessor EFM, without in the least compromising the reliability of the servo systems or data recovery.

Acknowledgement

The author is indebted to Shunji Yoshimura, Sony Central Research, for valuable discussions and contributions to the recording format.

References

[1] K.A.S. Immink, *Coding Techniques for Digital Recorders,* Prentice-Hall International (UK) Ltd., Englewood Cliffs, New Jersey, 1991.

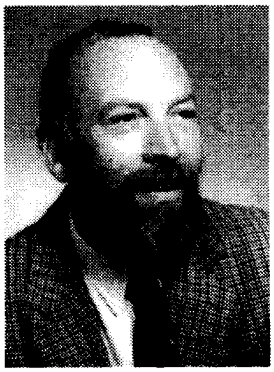
[2] J. Watkinson, *The Art of Digital Audio,* Focal Press, London, 1988.

[3] G.V. Jacoby, 'A New Look-Ahead Code for Increasing Data Density', *IEEE Trans. Magn.,* vol. MAG-13, no. 5, pp. 1202-1204, Sept. 1977. See also US Patent 4,323,931, April 1982.

[4] K.A.S. Immink, 'Constructions of Almost Block-Decodable Runlength-Limited Codes', *IEEE Trans. Inform. Theory,* vol. 41, no. 1, pp. 284-287, Jan. 1995.

[5] S. Tanaka, 'Method and Apparatus for Encoding Binary Data', US Patent 4,728,929, March 1, 1985.

Biography



Kees A. Schouhamer Immink received the M.S. and Ph.D degrees from the Eindhoven University of Technology in 1974 and 1985. Immink joined the Philips' Research Laboratories, Eindhoven, in 1968, where he is currently a Research Fellow. He is also adjunct professor at the Institute of Experimental Mathematics, Essen University, Germany. He contributed to the design and development of a variety of digital audio and video recorders such as the Compact Disc, Compact Disc Video, R-DAT, DCC, and MultiMedia CD. Immink holds 35 US patents and has written numerous papers in the field of coding techniques for optical and magnetic recorders. He is author of *Coding Techniques for Digital Recorders* and co-author of *Principles of Optical Disc Systems* and *Reed Solomon Codes: Theory and Application.* He is a Fellow of the AES, IEE, and IEEE; furthermore he received the AES Silver Medal in 1992, the IEE Sir J.J. Thomson Medal in 1993, and the SMPTE Poniatoff Gold Medal for Technical Excellence in 1994.