An RLL code design that maximises channel utilisation

Milan Simic

School of Aerospace, Mechanical, and Manufacturing Engineering,
RMIT University,
P.O Box 71, Bundoora East Campus,
Cnr. Plenty Road, Mc Kimmies Road,
Bundoora VIC 3083, Melbourne, Australia
Email: milan@rmit.edu.au

Abstract: Comprehensive (*d,k*) sequences study is presented, complemented with the design of a new, efficient, Run-Length Limited (RLL) code. The new code belongs to group of constrained coding schemas with a coding rate of $R = 2/5$ and with the minimum runlength between two successive transitions equal to 4. Presented RLL (4, ∞) code uses channel capacity highly efficiently, with 98.7% and consequently it achieves a high-density rate of $DR = 2.0$. It is implying that two bits can be recorded, or transmitted with one transition. Coding techniques based on the presented constraints and the selected coding rate have better efficiency than many other currently used codes for high density optical recording and transmission.

Keywords: run-length limited coding; digital communication; (*d,k*) sequences; magnetic recording; optical recording; near-field optical recording.

Reference to this paper should be made as follows: Simic, M. (2013) 'An RLL code design that maximises channel utilisation', *Int. J. Reasoning-based Intelligent Systems*, Vol. 5, No. 1, pp.63–72.

Biographical notes: Milan Simic is a Senior Lecturer and Program Director at the School of Aerospace, Mechanical, and Manufacturing Engineering, at RMIT University in Melbourne, Australia. He received his PhD in Electronics Engineering from the University of Niš, Serbia. He has an extensive track record of research across many settings, including the University of Niš, University of Pristina and RMIT University. His research interests include RLL coding, communications, computer networking, mechatronics, robotics, autonomous systems, simultaneous localisation and mapping, intelligent transport systems, remote data acquisition and control, sensor networks, hybrid and electric automotive systems, and renewable energy production.

This paper is a revised and expanded version of a paper entitled 'New RLL code for high density optical recording' presented at the 'Jubilee 10th International Conference on Telecommunications in Modern Satellite, Cable and Broadcasting Services (TELSIKS'2011)', Niš, Serbia, 5–8 October 2011.

1 Introduction

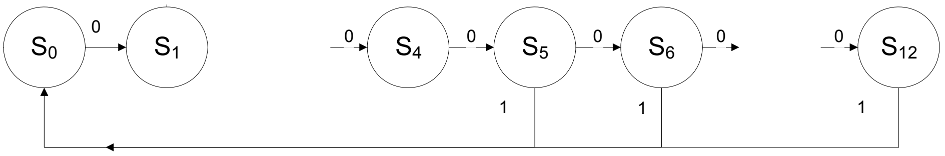
1.1 Communication channel modelling

Coding, in general, is a change of data representation from one alphabet into another, in order to fulfil a certain task in data processing or communications/recording. Line coding is translation of arbitrary data streams into suitable streams of symbols that can be accepted by the constrained channels. All physical channels, like electrical/electronic types, using electrical voltage, or current, as signal elements, or magnetic, optical, magneto-optical and other, using various types of physical quantities have their limitations.

Various RLL coding techniques, applied for line coding in digital data recording and communications, are designed to best suit baseband physical channel constraints. Those constraints are expressed in terms of spectrum limitations, needs for minimisation of Inter Symbol Interference (ISI) and efficiency in data transmission, or recording. Although, in every communication system, other coding techniques are employed to deal with errors, some error propagation limiting capability of the RLL coding is extremely helpful. This is particularly important when dealing with the symbol streams, i.e. codewords that have variable length, or state dependent RLL coding is applied.

Suitable symbol streams, used in RLL coding, known as (*d,k*) sequences, express discrete noiseless channels of the type defined by Shannon (1948), in which restrictions can be represented by sequential Finite State Machines (FSM), as shown in Figure 1.

Figure 1 Channel model, i.e. FSM model of $(d,k) = (5,12)$ constrained sequences



In the presented FSM model of a (d,k) constrained channel

- Parameter $d > 0$ reflects ISI;
- Second parameter $k < \infty$ is responsible for self-clocking;
- Where, usually code rules require that:

  $k \le M$, for ($M = 1,2,3,4,5,6,7,\ldots$Mmax),

  to ensure synchronisation at the receivers side and
- d and k together define signal spectrum.

In general, physical constraints of the channel are reflecting spectral shaping, self-timing (clock imbedded) and limitation on inter-symbol interference. A well-designed line code should have the following characteristics:

- No DC component
- Self-clocking is supported
- Some error detection capabilities especially error propagation limitation
- Physical channel required spectrum shaping is achievable.

To summarise, signal sequences are selected to match the characteristics of the channel, which is one of the main advantages of digital communications.

Binary stream, (d,k) constrained is basically a Non Return to Zero (NRZ) data representation. All RLL coding techniques apply pre-coding to that stream by use of NRZ Inverted (NRZI) scheme. If NRZ data are given as:

$$\{X_i\};\ i = 0 - \infty;\ X_i \in \{0,1\} \tag{1}$$

then

$$\{Y_i\};\ i = 0 - \infty;\ Y_i \in \{0,1\} \tag{2}$$

represents NRZI data after pre-coding.

This, NRZI application is expressed by a logical equation as:

$$Y_i = Y_{i-1} \oplus X_i \tag{3}$$

which derives as

$$Y_i = X_0 \oplus X_1 \oplus \ldots \oplus X_{i-1} \oplus X_i \tag{4}$$

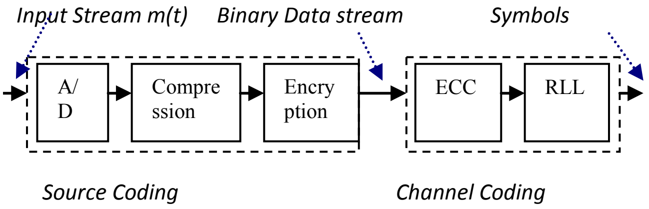
Since the symbol $\oplus$ represents module 2 addition, or *Exclusive OR* logic function, *XOR*, this whole transformation is also known as integration by module 2. As a result of this integration process, generated sequences given by equation (2) possess the property that the maximum and the minimum runlengths, i.e. number of consecutive like symbols, are limited between $(d + 1)$ and $(k + 1)$ symbol intervals. This can be clearly seen from the example shown in Table 1.

Table 1 An example of $(d,k) = (1,3)$ constrained sequences shown as X_i, before, and after NRZI precoding, as Y_i

X_i	0	1	0	1	0	0	1	0	0	0	1
Y_i	0	1	1	0	0	0	1	1	1	1	0

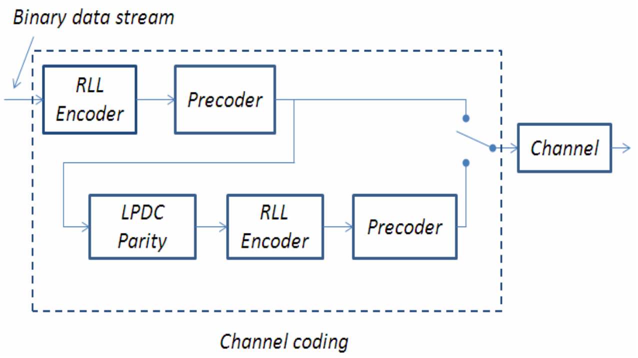
Blahut (1990) has given a comprehensive picture of digital transmission of information, put in the context and explained the role that each type of encoding techniques plays in a communication system. Standard concatenation scheme which Blahut employs is shown in Figure 2. It has been widely used in communications, optical and magnetic recording for many years.

Figure 2 Standard source and channel coding model



As shown in Figure 2, Error-Correcting Coding (ECC) is applied before line coding, i.e. RLL coding. RLL decoding is usually performed by the use of a look-ahead register. Coding techniques sequence arrangement, as shown in Figure 2, degrades Bit Error Rate (BER). In order to better control error propagation, another type of sequencing could be applied. In that changed concatenation scenario, ECC and RLL encoders' positions, in the channel coding part of the communication systems, are rearranged as shown in Figure 3. This is known as Low Density Parity Check (LDPC) coded Bliss scheme for RLL coding. LDPC generator will output non-constrained codewords. That is the reason to apply second RLL encoder in parity line and it is the same as the first RLL encoder.

Figure 3 Channel coding arrangement with LDPC-coded Bliss scheme (see online version for colours)



1.2 *Some familiar constrained coding schemes*

The Manchester coding scheme, used in digital communications and in magnetic recording, can be seen as an $RLL(0,1)$ coding, with coding rate $R = 1/2$ and the density rate of $DR = 0.5$. This code is a simple scheme which has good self-clocking characteristic; however the channel capacity is not used efficiently.

Another popular technique, Miller code, with the same coding rate of $R = 1/2$ and density $DR = 1.0$, is two times more efficient than Manchester and can be seen as $RLL(1,3)$ coding.

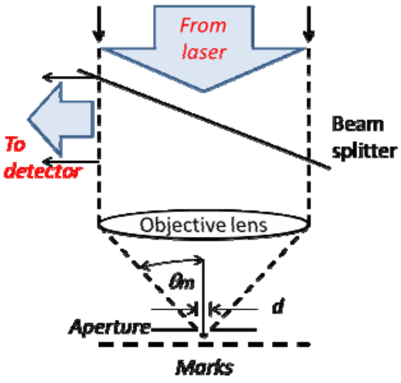
It was shown (Simic and Petrovic, 1989; Simic and Petrovic, 1991) that an additional step forward can be achieved with the coding techniques based on $RLL(5,k)$ constraints, where $k = (12 - \infty)$. Coding rate is $R = 1/3$ and the density rate achieved is $DR = 2.0$. Recently, a selected RLL codes' performance investigation results were published as an outcome of the joint Sony and Philips research project in the area of high-density optical recording (Haibin et al., 2007). Performance of RLL codes with different d constraints were analysed, for LDPC-coded Bliss scheme and for the standard concatenation scheme. RLL coding schemes with $d = 0$, 1, 2 and 5 were investigated for the application in the near-field optical recording.

1.3 *Research in optical channels*

Constrained, run-length limited coding is applied across a variety of physical channels for digital communication and for the recording. This is a quick look at the applications in a relatively new technology area: Blue-ray storage systems. By using a Solid Immersion Lens (SIL) that flies over the disc, near-field optical recording achieves high-numerical aperture (NA) (Verschuren et al., 2006). The distance between the lens and the disc is just few tens of nanometres. Numerical aperture is a key factor of feasible storage capacity.

In addition to that, new discovered techniques could be applied to any other baseband channel. Finally, results in near-field optical recording and holographic data storage are also applicable in other research areas (Park et al., 2011). The benefits from the emerging technologies are new memories small physical size, high-position accuracy and low-production costs. An example of optical physical channel is shown in Figure 4.

Figure 4 A near-field optical recording system with aperture (see online version for colours)



Apart from the efficient data storage, applications of the new optical technology include near-field optics-based microscopy, new imaging systems of the small size, plasmonic nanolithography using an SIL, nanoscale aperture, bio-applications and other.

2 Constrained channel capacity

2.1 *Coding with constrained (d,k) sequences*

Most coding techniques employed in a communication system can be conducted as block coding, where unconstrained input data sequences of length m are mapped to the constrained output sequences, of the length n. In the case of unconstrained coding, and when both input and the output sides of an encoder use binary alphabet for data representation and the codewords, it is obvious that the following expression must be fulfilled:

$$2^m \le 2^n \tag{5}$$

Similarly, when constrained coding is applied the number of constrained sequences available, $N_{dk}(n)$, must be equal, or larger than the number of input, unconstrained sequences. If a binary system, with the alphabet $B = (0,1)$, is used for the unconstrained alphabet at the input, and for the constrained output sequences, i.e. codewords generation, then the same basic coding requirement is expressed as:

$$2^m \le N_{dk}(n) \tag{6}$$

Although constrained sequences comply with channel requirements they have less information capacity, $C(d,k)$. In order to compensate for this loss, clock rate has to be increased as given:

$$CLR = n / m \tag{7}$$

Increasing the clock rate shortens the bit window.

Thanks to the limitation introduced as minimum number of new bit windows between successive transitions, constrained coding archives longer periods between two transitions. That is used as a measure of code efficiency, or density rate (DR) attained.

In addition to block coding, Variable Length (VL) coding can also be conducted. In both cases, coding has to be synchronous, which means that the *bit per symbol ratio*, coding rate R is constant over each data word mapping, and it is given as:

$$R = m / n = 1/CLR \tag{8}$$

Coding efficiency, or density ratio of an RLL code, is the ratio of the minimum signal transitions timing and data bits timing. It is expressed as following:

$$DR = \frac{T_{out\_\min}}{T_{in}} = R(d+1) \tag{9}$$

In the above expression symbol T_{in} represents input data strings bit window, while T_{out_min} corresponds to the minimum time interval between output signal transitions.

Referring to equation (6), the number of codewords available for the given (*d,k*) constraints has to be identified. Recursion relations, derived to define this number, are given by Tang and Bahl (1970). It can be shown that the characteristic equation of (*d*,*k*) sequences is given as:

$$Z^{k+2} - Z^{k+1} - Z^{k-d+1} + 1 = 0 \tag{10}$$

Characteristic equation incorporates particular cases when $d = 0$, and when $k = \infty$. Information capacity of the constrained sequences is set as:

$$C\left(d,k\right) = \log_2 \lambda \tag{11}$$

where λ is the largest positive root of equation (10). The maximum capacity can be obtained for large values of parameter *k*, as:

$$C_{\max} = \lim_{k\to\infty} C\left(d,k\right) = \log_2 \lambda_d \tag{12}$$

Table 2 shows the capacity of *d* constrained sequences, *C*(*d*,∞) = *C*(*d*) and the corresponding maximum achievable density ratio, i.e. coding efficiency, for the values of *d* = (0, 1,...,10). Table 3 presents the selected channels' capacity and the coding efficiency that could be achieved, with the coding techniques for the various values of the parameter *d* and the finite values of the second constraint, *k*. It can be seen from Tables 2 and 3, that with the values of first constraint, $d < 4$ a significant increase of the density rate of up to 2 and above is not possible. Finally, from Table 3, it could be observed what could be the minimum possible values of parameter *k* that would still give us efficiency above 2.0. For $k = 4$ it is 16 and for the $k = 5$ it is 12. The first versions of *RLL*(5,16) coding technique, with data rate *R* = *1*/3 and the density rate *DR* = 2.0, was presented by Simic and Petrovic (1989) and the second one with better error propagation limitation was published in the work of Simic and Petrovic (1991). Much later Haibin et al. (2007) reported on considering the use of this class of codes, with *R* = 1/3 and *d* = 5 for the applications in near-field optical recording, for the new generations of Blu-ray (BD) discs.

Table 2 Capacity and coding efficiency of (*d*,∞) constrained sequences

d	*Capacity*	*Efficiency [bits per transition]*
0	1.000	1.000
1	0.694	1.338
2	0.551	1.654
3	0.464	1.859
4	0.405	2.028
5	0.361	2.171
6	0.328	2.297
7	0.301	2.408
8	0.278	2.508
9	0.260	2.600
10	0.244	2.684

Table 3 Capacity and coding efficiency as functions of *d* and *k* constraints

D	*k*	*Capacity C(d,k)*	*Efficiency [bits per transition]*
0	1	0.694	0.694
0	3	0.947	0.947
1	3	0.551	1.102
1	7	0.679	1.358
2	7	0.517	1.551
3	7	0.406	1.629
3	11	0.452	1.808
3	17	0.463	1.852
4	9	0.362	1.810
4	12	0.389	1.946
4	15	0.399	1.995
4	16	0.400	2.004
5	11	0.328	1.969
5	12	0.336	2.021
5	13	0.343	2.059
5	14	0.345	2.075
5	15	0.351	2.107
5	16	0.353	2.122
5	17	0.357	2.136
6	13	0.301	2.107
6	20	0.324	2.268
7	15	0.279	2.232
7	23	0.298	2.284
7	17	0.260	2.340
8	26	0.276	2.484

Just recently a new efficient coding technique was presented by Simic (2011). With the coding rate of *R* = 2/5 and the constraint *d* = 4 it achieves high-density rate of *DR* = 2.0. This paper presents further contribution to that coding research. The new coding scheme is designed and a more comprehensive approach and presentation of the encoding and decoding solutions are given.

2.2 Various channel models and encoding schemas

Many researchers were already looking for the coding solutions on various (*d*,*k*) channels shown in Table 3. Other RLL schemas were designed with the coding rate of *R* = 4/11, and (*d*,*k*) = (4,20) achieving density rate of *DR* = 1.818 (Kim and Lee, 1995). There are also RLL codes proposed with (*d*,*k*) = (4,13), (*d*,*k*) = (3,11) and more, all with *DR* below 2.0.

It is much harder to define coding schemas when the coding rate is close to the capacity of the given (*d*,*k*) channel. An interesting study on successful allocation of codewords to the encoder states to maximise the code rate is given in Cai and Immink's (2006). The study shows that the number of encoder states, for the certain RLL codes, can always be associated with generalised Fibonacci numbers.

Research was conducted for the design of the capacity-approaching (d,k) codes with $d = 1$ constraint. It would be interesting to generalise this approach for the higher values of channel constraints.

Starting from the beginning, there is already a large number of line coding techniques with parameter $d = 0$ constraint. The most popular and widely used are unipolar, or bipolar, NRZ and NRZI, with many other variations of techniques based on those. Finally, as already mentioned, Manchester code, which is $RLL(0,1)$, is one of the mostly used. For example, IEEE 802.3 standards for LAN and MAN employ Manchester coding.

One of the interesting approaches is the maximum RLL coding. Codes from this group are basically $RLL(0,k)$ techniques which eliminate predefined, unwanted sequences from the output stream of symbols (Van Wijngaarden and Immink, 2010).

Further to this, there are many techniques based on $d = 1$ constraint and coding rate of $R = 1/2$. Again, one of the most popular, Miller code was already mentioned here. It is just one of the large number of codes designed on $(d,k) = (1,k)$ constrains. Some other are: Modified Frequency Modulation (MFM) or $RLL(1,3)$, Modified MFM (M^2MFM) or $RLL(1,4)$, Jordan code and other. Another $d = 1$ technique $RLL(1,7)$ has coding rate of $R = 2/3$ and achieves better density rate of $DR = 1.33$.

Among the techniques with $d = 2$, a well-known technique, $RLL(2,7)$ coding, is used for magnetic storage. With the coding rate of $R = 1/2$ it achieves respective density rate of $DR = 1.5$. Eight-to-Fourteen Modulation (EFM) is a channel code used by compact discs (CD) that is basically $RLL(2,10)$, with the coding rate of $R = 8/17$ and density rate of $24/17 = 1.41$.

One of the streams in the further development of run-length coding is two dimensional (2D) and three dimensional (3D) constrained coding. An investigation in application of $RLL(0,2)$ and $RLL(3,\infty)$ combined is presented in the Sharov and Roth's (2010) study. Encoder – decoder solution is given for a large number of (2D) RLL constrained codes in the work of Tal et al. (2009). Sliding block decoding is applied in most of the linear, i.e. one-dimensional (1D), RLL encoding schemas. In the procedure shown, a (2D) constraint is converted into (1D) case and processed. Multi-dimensional constrained coding becomes interesting for optical applications and holographic storage.

One more encoding schema, from the group of run-length codes with $d = 4$, is $RLL(4,18)$ code with the coding rate of $R = 1/3$ and the density rate of $DR = 1.67$. It was presented and performances were investigated for optical storage applications (Yang et al., 2004).

This was a short review of the coding solutions. The new code design is following. When selecting a new coding rate the following rule must be obeyed:

$$R = m/n < C(d,\infty) \tag{13}$$

By examining Table 2, for $d = 4$, it was perceived that the coding with the coding rate of $R = 2/5$ is possible and that the channel capacity, in that case, would be used extremely efficiently. The efficiency of RLL codes, in terms of channel information capacity utilisation (Immink, 2001), is defined as following:

$$\eta = R/C(d, k) \tag{14}$$

For the selected coding rate of R=2/5 it is equal to 98.7%, since the following

$$R = 2/5 = 0.4 < C(4,\infty) = 0.405 \tag{15}$$

Chosen coding rate will provide significant density rate of

$$DR = R(d+1) = 2.0 \tag{16}$$

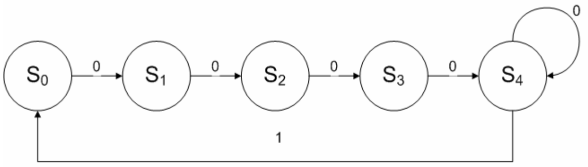
A comprehensive approach in constrained codes design is presented next and a new efficient code for multipurpose applications in data recording and communications is developed.

3 New RLL coding schemes solutions

Constrained codes considered here are from the class of trellis codes for data translation. A continuous stream of unconstrained data symbols enters the encoder, and another stream of (d,k) constrained codeword signals leave the encoder. The input data stream is of arbitrary length and is so long that the length plays no role in the design of the RLL code encoder and decoder. Input is viewed as an infinite stream of data symbols. The encoder output is consequently an infinite stream of codeword symbols.

There are various methods for constrained codes design. A systematic approach, used here, is based on Adler, Coppersmith, Hassner (ACH) technique (Adler et al., 1983). In the process of a constrained channel code design, it is possible to derive an encoder state transition table, in methodical manner, starting from an FSM model of the channel constraints. A model of $(4,\infty)$ channel is shown in Figure 5 and it will used in the design process.

Figure 5 FSM model of $(4, \infty)$ constrained channel



The original ACH method is based on a complex state splitting algorithm. Blahut (1990) introduced more mathematical background of the procedure and simplified the method. When the efficiency of coding is close to its maximum capacity for the given channel it becomes more challenging to define translation rules. Finite value of k constraint, for any $k = \infty$ code, can be achieved by substitutions applied when too many zeros would occur in the channel stream. This approach is used in the case of $RLL(1,10)$ coding presented in Coene et al.'s (2006) study. Another way to deal with infinity is by applying reverse concatenation coding, for storage systems, in which the information is encoded first by a constraint code and then by ECC as shown in Figure 3. Unconstrained parts of the RLL coder output stream are filled in with parity check bits of ECC code and that action breaks the unconstrained stream. Two families of RLL constraints were investigated, *RLL(d,*∞) and *RLL*(*d*,2*d*+2) and the results presented in the work of Louidor (2010).

Finally, the encoder design procedure begins with a constrained channel description given by a state transition matrix ***B***. Matrix ***B*** is associated with FSM model as shown in Figure 5, and it is a square 5-*by*-5 matrix, in this case, as given below.

$$B=\begin{bmatrix}0&1&0&0&0\\0&0&1&0&0\\0&0&0&1&0\\0&0&0&0&1\\1&0&0&0&1\end{bmatrix} \tag{17}$$

In this matrix representation, matrix entry $b_{ij}=1$ indicates that there is a transition between two associated FSM states: from $\boldsymbol{S}_i$ to $\boldsymbol{S}_j$. Those transitions are the paths of length one, i.e. one symbol element long. Since two data bits are represented with five signal elements, according to the chosen coding rate, $R=2/5$, another transition matrix is needed whose entries will indicate existence of the paths through the FSM of the length 5.

Following that, new matrix $\boldsymbol{B}_5$ is derived from ***B***, or could be generated directly from the FSM channel model. Matrix $\boldsymbol{B}_5$ contains the number of paths of the length 5 from the state $\boldsymbol{S}_i$ to the state $\boldsymbol{S}_j$, where *i*, *j* € {0, 1, 2, 3, 4}. As already known, from the work of Blahut (1990), there is a useful property of the transition matrix ***B*** which states that

$$\boldsymbol{B}_n=\boldsymbol{B}^n \tag{18}$$

This means that, in order to generate this matrix, the paths of length 5 could be traced through the FSM model of the channel, or transition matrix ***B*** of single paths could be multiplied to the power of *n*, or five, in this case. Accordingly, matrix $\boldsymbol{B}_5$ is given as:

$$B_5=B^5=\begin{bmatrix}1&0&0&0&1\\1&1&0&0&1\\1&1&1&0&1\\1&1&1&1&1\\1&1&1&1&2\end{bmatrix} \tag{19}$$

It could be observed, from the expression for matrix $\boldsymbol{B}_5$ that there are two paths from the state $\boldsymbol{S}_4$ to itself, while all other transitions are single, or do not exist. This will be important later for encoder design. The basic coding prerequisite, as given in equation (6) can now be expressed in matrix form as:

$$2^{\mathrm{m}}\mathbf{v}\le\mathbf{B}^{\mathrm{n}}\mathbf{v} \tag{20}$$

Following this, the next step in the procedure is to find an integer-valued vector ***v*** of length 5 that satisfies matrix inequality equation (20). This vector is then used to define the number of new states for each channel state, in the state splitting process to be performed next. The splitting process is necessary for definition of a more specific encoder model that should be derived from the initial FSM channel model as mentioned before.

4 Transition from FSM to encoder

In this particular case, for (4,∞) modelling, components of the vector ***v***, are given as:

$$v(p)\ (p=0\text{ to }4) \tag{21}$$

Encoder matrix ***E*** is a square matrix of the order *q* where:

$$q=\sum v(p)\text{ for }p=0\text{ to }4 \tag{22}$$

By solving matrix inequality equation (20) for the FSM model from Figure 5 with $m=2$, $n=5$, the vector ***v*** could be found. A solution is:

$$v=\begin{bmatrix}2\\3\\4\\5\\7\end{bmatrix} \tag{23}$$

This solution can easily be verified by direct substitution.

Vector component ***v***(*i*) defines the number of Encoder states, denoted by subscripts and superscripts as well, i.e. $\boldsymbol{S}_i^j$. They are generated by splitting the corresponding FSM state $\boldsymbol{S}_i$. According to that, the association between channel states and encoder states that can be set up is as following:

- FSM State S_0 should be split into ***v***(0) = 2 encoder states: $S_0{}^1$ and $S_0{}^2$
- Another FSM state S_1 should be split into ***v***(1) = 3 states, and so on.

All other states should be split accordingly generating a total of 21 new, encoder states. Generally speaking each FSM state S_i will be split into a set with the number of *v*(*i*) states, S_i^1 where $l=1$ to *v*(*i*). In summary, starting from the set of FSM states,

$$S_{FSM}=\{S_0,S_1,S_2,S_3,S_4\} \tag{24}$$

the set of encoders states can be generated as following

$$\begin{aligned}S_{Encoder}=\{&S_0^1,S_0^2,S_1^1,S_1^2,S_1^3,S_2^1,S_2^2,S_2^3,S_2^4,S_3^1,S_3^2,S_3^3,S_3^4,S_3^5,\\&S_4^1,S_4^2,S_4^3,S_4^4,S_4^5,S_4^6,S_4^7\}\end{aligned} \tag{25}$$

Since two data bits are replaced with five signal elements, four pathways should leave each encoder state. Furthermore, it is important that input order of each state should be at least equal to a minimum of one. Hence, each encoder state has at least one input path and a minimum of four output paths, as per equation (5).

The notation $S_i^{1,2,\ldots,v(i)}$ refers to the set of substates originating from the FSM state S_i. Therefore, that state should be split into *v*(*i*) new states.

Another way to explain splitting process would be as following: if $\boldsymbol{B}_5$ entry $b_{ij} = 1$ then all S_i^1 to S_j^p state transitions exist, where $i,j = 0, \ldots, (n-1)$ while $l = 1, \ldots, v(i)$ and $p = 1, \ldots, v(j)$. In that case, the appropriate element of the encoder matrix is 1. In all other cases elements of encoder matrix are equal to zero.

There might be more transitions than required in the extended matrix, due to the enlarged number of states and hence transitions. To manage this, the next stage in encoder design is to delete unnecessary transitions. This can be completed by changing some entries from 1 to 0 and so define the more specific encoder state transition matrix $\boldsymbol{E}$.

The procedure could start by splitting the first state S_0 into new encoder states. According to the value of the first element in vector $\boldsymbol{v}$, which is 2, there should be two new states which are shown in the set $S_{Encoder}$ above. Transitions should be allocated to each new state in the new state transition matrix, i.e. now encoder matrix. In order to do that, it is needed to look at the successors of the splitting state, in the $\boldsymbol{B}_5$ transition matrix. They are S_0 and S_4. They both have substates denoted by superscripts as following:

$$S_0^{1,2} \rightarrow S_0^{1,2},\ S_4^{1,2,3,4,,5,6,7} \tag{26}$$

The next step is to partition the set of successor states into the subsets with minimum four elements, presented by the superscripts. This action will generate the following subsets:

$$\left\{S_0^{1,2}, S_4^{1,2,3,4,5,6,7}\right\} = \left\{S_0^{1,2}, S_4^{1,2}\right\} U \left\{S_4^{3,4,5,6,7}\right\} \tag{27}$$

Finally, each of the new states (now becoming encoder states) has to be associated with a subset of new successor states as destinations for the transitions. It is shown as:

$$S_0^1 \rightarrow \left\{S_0^{1,2}, S_4^{1,2}\right\} \tag{28}$$

$$S_0^2 \rightarrow \left\{S_4^{3,4,5,6,7}\right\} \tag{29}$$

The last subset, given in equation (29), has more than four superscripts, i.e. states, corresponding to transitions. One transition has to be discarded, which could be any of those five transitions. Algorithm has to be performed until all states are split.

The full encoder state transition matrix is illustrated in Figure 6. In this matrix expression, if the transition from two encoder states exists, corresponding matrix entry has value of 1. Symbol x appears if the corresponding transition exists, as the result of the splitting process, but it has been discarded since there were more than four transitions in the row, essentially from the associated encoder state.

As previously discussed, codewords correspond to the paths of length 5, through the FSM channel model. The set of available codewords, C, is given below and it consists of only six elements.

$$C = \{00001, 00010, 00100, 01000, 10000, 00000\} \tag{30}$$

The set of unconstrained input data bits, D, is given as:

$$D = \{00, 01, 10, 11\} \tag{31}$$

New code design can be finalised by specifying another state transition matrix, this time with codewords and also with association of codewords to the data. This matrix is shown in Figure 7.

One realisation of the new code could be achieved by simply associating data words, as presented in the set D, sequentially to the codewords in each row of the transition matrix. This association can also be seen in Table 4, which can be used as an encoder look up table, or as a truth table to define logic functions of the encoder. There are ten logic functions of seven input variables. For example codeword functions are given as following:

$$C_i = f_1(x_4, x_3, x_2, x_1, x_0, d_1, d_0)\ i = (0-4) \tag{32}$$

Figure 6 State transition matrix for new RLL code generated by splitting FSM states

$$\begin{array}{c|ccccccccccccccccccccc}
 & S_0^1 & S_0^2 & S_1^1 & S_1^2 & S_1^3 & S_2^1 & S_2^2 & S_2^3 & S_2^4 & S_3^1 & S_3^2 & S_3^3 & S_3^4 & S_3^5 & S_4^1 & S_4^2 & S_4^3 & S_4^4 & S_4^5 & S_4^6 & S_4^7 \\
\hline
S_0^1 & 1 & 1 & & & & & & & & & & & & & 1 & 1 & & & & & \\
S_0^2 & & & & & & & & & & & & & & & & & 1 & 1 & 1 & 1 & x \\
S_1^1 & 1 & 1 & 1 & 1 & & & & & & & & & & & & & & & & & \\
S_1^2 & & & & & 1 & & & & & & & & & & 1 & 1 & 1 & & & & \\
S_1^3 & & & & & & & & & & & & & & & & & & 1 & 1 & 1 & 1 \\
S_2^1 & 1 & 1 & 1 & 1 & & & & & & & & & & & & & & & & & \\
S_2^2 & & & & & 1 & 1 & 1 & 1 & & & & & & & & & & & & & \\
S_2^3 & & & & & & & & & 1 & & & & & & 1 & 1 & 1 & & & & \\
S_2^4 & & & & & & & & & & & & & & & & & & 1 & 1 & 1 & 1 \\
S_3^1 & 1 & 1 & 1 & 1 & & & & & & & & & & & & & & & & & \\
B_E = S_3^2 & & & & & 1 & 1 & 1 & 1 & & & & & & & & & & & & & \\
S_3^3 & & & & & & & & & 1 & 1 & 1 & 1 & & & & & & & & & \\
S_3^4 & & & & & & & & & & & & & 1 & 1 & 1 & 1 & & & & & \\
S_3^5 & & & & & & & & & & & & & & & & & 1 & 1 & 1 & 1 & x \\
S_4^1 & 1 & 1 & 1 & 1 & & & & & & & & & & & & & & & & & \\
S_4^2 & & & & & 1 & 1 & 1 & 1 & & & & & & & & & & & & & \\
S_4^3 & & & & & & & & & 1 & 1 & 1 & 1 & & & & & & & & & \\
S_4^4 & & & & & & & & & & & & & 1 & 1 & 1 & 1 & & & & & \\
S_4^5 & & & & & & & & & & & & & & & & & 1 & 1 & 1 & 1 & \\
S_4^6 & & & & & & & & & & & & & & & 1 & 1 & 1 & & & & 1 \\
S_4^7 & & & & & & & & & & & & & & & & & & 1 & 1 & 1 & 1
\end{array}$$

Figure 7 State transition matrix for new RLL code with the codewords assigned

$$B_E = \begin{array}{c|ccccccccccccccccccccc}
 & s_0^1 & s_0^2 & s_1^1 & s_1^2 & s_1^3 & s_2^1 & s_2^2 & s_2^3 & s_2^4 & s_3^1 & s_3^2 & s_3^3 & s_3^4 & s_3^5 & s_4^1 & s_4^2 & s_4^3 & s_4^4 & s_4^5 & s_4^6 & s_4^7 \\
\hline
s_0^1 & 00001 & 00001 &&&&&&&&&&&& & 00000 & 00000 &&&&& \\
s_0^2 &&&&&&&&&&&&&&&&& & 00000 & 00000 & 00000 & 00000 & \\
s_1^1 & 00001 & 00001 & 00010 & 00010 &&&&&&&&&&&&&&&&& \\
s_1^2 &&&& & 00010 &&&&&&&&& & 00000 & 00000 & 00000 &&&& \\
s_1^3 &&&&&&&&&&&&&&&&& & 00000 & 00000 & 00000 & 00000 \\
s_2^1 & 00001 & 00001 & 00010 & 00010 &&&&&&&&&&&&&&&&& \\
s_2^2 &&&& & 00010 & 00100 & 00100 & 00100 &&&&&&&&&&&&& \\
s_2^3 &&&&&&&& & 00100 &&&&& & 00000 & 00000 & 00000 &&&& \\
s_2^4 &&&&&&&&&&&&&&&&& & 00000 & 00000 & 00000 & 00000 \\
s_3^1 & 00001 & 00001 & 00010 & 00010 &&&&&&&&&&&&&&&&& \\
s_3^2 &&&& & 00010 & 00100 & 00100 & 00100 &&&&&&&&&&&&& \\
s_3^3 &&&&&&&& & 00100 & 01000 & 01000 & 01000 &&&&&&&&& \\
s_3^4 &&&&&&&&&&&& & 01000 & 01000 & 00000 & 00000 &&&&& \\
s_3^5 &&&&&&&&&&&&&&&& & 00000 & 00000 & 00000 & 00000 & \\
s_4^1 & 00001 & 00001 & 00010 & 00010 &&&&&&&&&&&&&&&&& \\
s_4^2 &&&& & 00010 & 00100 & 00100 & 00100 &&&&&&&&&&&&& \\
s_4^3 &&&&&&&& & 00100 & 01000 & 01000 & 01000 &&&&&&&&& \\
s_4^4 &&&&&&&&&&&& & 01000 & 01000 & 10000 & 10000 &&&&& \\
s_4^5 &&&&&&&&&&&&&&&& & 10000 & 10000 & 10000 & 10000 & \\
s_4^6 &&&&&&&&&&&&&& & 00000 & 00000 & 00000 &&& & 10000 \\
s_4^7 &&&&&&&&&&&&&&&&& & 00000 & 00000 & 00000 & 00000
\end{array}$$

Table 4 fully corresponds to Figure 7.

Table 4 A segment from the encoder truth table

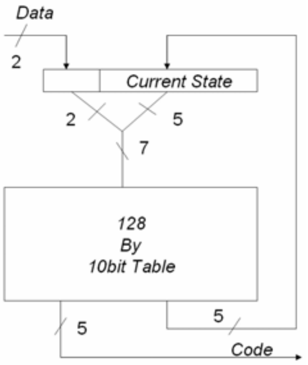
Input			*Output*		
Initial State	*State in binary* $x_4x_3x_2x_1x_0$	*Data* $d_1\ d_0$	*Codeword* $C_4C_3C_2C_1C_0$	*New State in binary*	*New State*
S_0^1	00000	00	00001	00000	S_0^1
S_0^1	00000	01	00001	00001	S_0^2
S_0^1	00000	10	00000	01110	S_4^1
S_0^1	00000	11	00000	01111	S_4^2
S_0^2	00001	00	00000	10000	S_0^1
S_0^2	00001	01	00000	10001	S_0^2
…					
S_4^7	10100	10	00000	10011	S_4^6
S_4^7	10100	11	00000	10100	S_4^7

This encoder truth table has 21*4 entries. Since there are seven input bits, other

$$128 - 84 = 44$$

entries are illegal. This leads into some error detection capabilities. Furthermore, it might be good to investigate if some of the particular associations of data bits to codewords can contribute to the better error propagation capabilities of this RLL $d = 4$ coding, as it was already shown for *RLL* $d = 5$ codes (Simic and Petrovic, 1991). The encoder block diagram is illustrated in Figure 8.

Figure 8 New RLL(4,∞) code table look-up encoder



As mentioned before, *RLL* (*d*,*k*) codes belong to ACH class of coding techniques. Importantly, one of the advantages of ACH codes is that errors in the channel can cause only a finite number of errors at the decoder output.

5 Decoder design

It was demonstrated how an expanded set of FSM states becomes internal states set for an RLL encoder. The same new states are important for the decoding process and refer to the decoder as well. Looking from the original FSM channel model, each subset of states sharing the same subscript corresponds to the original state, but decoder should also reconstruct the sequence of internal states, of the encoder, for the purpose of data decoding. The main problem is that the same codewords could represent different data. Therefore, information about the future encoder states is vital for the decoding process. Few scenarios were analysed and are presented here.

For example, codeword received at the discrete time instance $t = n$ was $C_n = 00001$. It is shown in the extract of the decoder table (Table 5). The codeword will cause channel FSM model transition to state S_0. However, this information is insufficient to make the decision regarding the data bit associated with the codeword. There are two possible substates that encoder could have ended up when generating codeword 00001; they are S_0^1 and S_0^2. The first one would correspond to data bits $d_n = 00$ and the second one to $d_n = 01$. This can be seen form Table 4, as shown in the first two rows.

Table 5 A segment from the decoder table

C_n	C_{n+1}	C_{n+2}	C_{n+3}	d_n	d_{n+1}
00001	00001			00	
00001	00000	00001		00	10
		00010	00001	00	10
		00100			
		01000			
		10000			
		00000			

In order to assign superscript, decoder needs to look at the following codeword. There are just two possible, legal codewords that could appear now: 00001 and 00000. This can be seen from the encoder state transition matrix given in Figure 7. If the following codeword, C_{n+1} is again 00001, which would point to superscript 1, this would further indicate that the encoder ended up in the S_0^1 state and that it was caused by the data word transmitted being 00.

The second possible codeword 00000 could not support decoder to make a decision about the superscript, at this stage, since it could be generated from any of the two S_0 states as it can be seen from the encoder table (Table 4). The decoder will need to continue analysing more incoming codewords using the same algorithm to obtain the result.

After receiving the sequence of 00000 the channel FSM model will go to the state S_4. If it was sent from S_0^1 substate the new substates would be either S_4^1 or S_4^2. If it originated for the different substate, S_0^2, the subsequent encoder states could be any of the following: S_4^3, S_4^4, S_4^5, S_4^6. The decoder proceeds with the algorithm again.

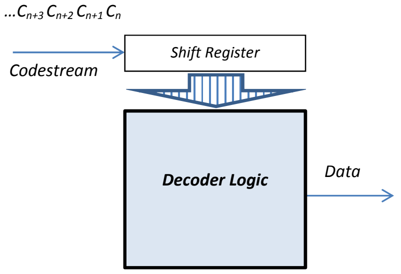
The only two possible codewords from the S_4^1 are 00001 and 00010. The first would end the ambiguity since it is only possible from the first of the considered states now. The second codeword appearance is possible from the two S_4 states, S_4^1 and S_4^2. This means that decoder has to proceed with the algorithm once more.

Other allowed codewords, from the rest of the subset of S_4 states considered here, are given in Table 5. This was a short presentation of the decoding process based on the look ahead detection.

Fine tuning of the association between two sets, the set of codewords, C, and the set of data, D, as given above, could shorten and simplify the decoding process. Code optimisation should be subjected to further research which may, also, lead to improved error propagation limitations (Simic and Petrovic, 1991).

According to the encoder truth table, (Table 4), there are 84 legal entries in the table, while the number of total entries is 128. There are six combinations that could appear for a single codeword; for two codewords it will make 36 and for the three codewords there are 6^3 combinations (216 combinations). Subsequently, it is expected that the decoder which uses feed-forward shift register, with the current codeword and the following 3, at the average, would be able to uniquely decode channel sequence received. This could be the subject of further research and fine modifications in the association of data to the state transitions and codewords assignment. An error in the received codestream could effect at most four data combinations, i.e. eight data bits. This is concluded according to the overall characteristic of the ACH codes. Generally, it states that the channel errors can cause just a finite number of errors in the decoder output. The decoder block diagram is shown in Figure 9.

Figure 9 Look ahead decoder block diagram for new RLL code (see online version for colours)



6 Conclusion

A short review of constrained coding is presented. New coding ideas were investigating, leading to new efficient coding techniques based on the channel *(d,k)* = (4, ∞) modelling with the coding rate $R = 2/5$, which is close to the maximum channel information capacity. Current findings suggest that the proposed code, or codes, achieve the best possible use of the given channel capacity. However, further comprehensive performance investigation is needed to validate possible applications. Codes can be used for data recording and transmission on various physical channels. Since the majority of communication systems already have well-established standards, the new and emerging technologies, like near-field optical recording for the new generations of Blue-ray Discs, could easily benefit from this research.

References

Adler, R., Coppersmith, D. and Hassner, M. (1983) 'Algorithms for sliding block codes –an application of symbolic dynamics to information theory', *IEEE Transactions on Information Theory*, Vol. 29, pp.5–22.

Blahut, R.E. (1990) *Digital Transmission of Information*, Addison-Wesley Publication Co., New York, USA.

Cai, K. and Immink, K.S. (2006) 'On the number of encoder states of a type of RLL codes', *IEEE Transactions on Information Theory*, Vol. 52, pp.3313–3319.

Coene, W., Hekstra, A., Yin, B., Yamagishi, H., Noda, M., Nakaoki, A. and Horigome, T. (2006) 'A new d = 1, k = 10 soft-decodable RLL code with r = 2 MTR-constraint and a 2-to-3 PCWA mapping for DC-control', *Optical Data Storage Topical Meeting*, pp.168–170.

Haibin, Z., Hekstra, A.P., Coene, W.M.J. and Bin, Y. (2007) 'Performance investigation of soft-decodable runlength-limited codes with different minimum runlength constraints in high-density optical recording', *IEEE Transactions on Magnetics*, Vol. 43, pp.3525–3534.

Immink, K.S. (2001) 'Efficient Dc-free RLL codes for optical recording', *Proceedings of IEEE International Symposium on Information Theory*, Vol. 51, pp.326–331.

Kim, M-G. and Lee, J.H. (1995) '(4, 20) runlength limited modulation code for high density storage system', *Electronics Letters*, Vol. 31, pp.539–541.

Louidor, E. (2010) 'The tradeoff function for a class of RLL (d, k) constraints', *IEEE International Symposium on Information Theory Proceedings (ISIT 2010)*, 13–18 June, pp.1223–1227.

Park, N.C., Park, Y.P., Park, K.S. and Yang, H. (2011) 'Applications of next generation optical data storage technologies', *IEEE Transactions on Magnetics*, Vol. 47, pp.669–678.

Shannon, C.E. (1948) 'A mathematical theory of communication', *The Bell System Technical Journal*, Vol. 27, No. 3, pp.379–423.

Sharov, A. and Roth, R.M. (2010) 'Two-dimensional constrained coding based on tiling', *IEEE Transactions on Information Theory*, Vol. 56, pp.1800–1807.

Simic, M.N. (2011) 'New RLL code for high density optical recording', *10th International Conference on Telecommunication in Modern Satellite Cable and Broadcasting Services (TELSIKS, 2011)*, 5–8 October, Serbia, Niš, pp.160–163.

Simic, M. and Petrovic, R. (1989) 'New RLL code for digital data storage', *Electronics Letters*, Vol. 25, pp.951–953.

Simic, M. and Petrovic, R. (1991) 'New EPL RLL(5,16) code', *Electronics Letters*, Vol. 27, pp.2100–2102.

Tal, I., Etzion, T. and Roth, R.M. (2009) 'On row-by-row coding for 2-D constraints', *IEEE Transactions on Information Theory*, Vol. 55, pp.3565–3576.

Tang, D.T. and Bahl, L.R. (1970) 'Block codes for a class of constrained noiseless channels', *Information and Control*, Vol. 17, pp.436–461.

Van Wijngaarden, A.J. and Immink, K.S. (2010) 'Construction of maximum run-length limited codes using sequence replacement techniques', *IEEE Journal on Selected Areas in Communications*, Vol. 28, pp.200–207.

Verschuren, C.A., van den Eerenbeemd, J.M.A., Zijp, F., Lee, J.I. and Bruls, D.M. (2006) 'Near-field recording with a solid immersion lens on polymer cover-layer protected discs', *Japanese Journal of Applied Physics Part 1 – Regular Papers Brief Communications & Review Papers*, Vol. 45, pp.1325–1331.

Yang, X., Gong, M., Huang, L., Yan, P. and Zhang, H. (2004) 'Performance of run-length limited (4, 18) code for optical storage systems', *Optical and Quantum Electronics*, Vol. 36, pp.1079–1088.