Frequency Modulation for Asynchronous Data Transfer

Saleem Mukhtar and Jehoshua (Shuki) Bruck

Abstract— **Consider a communication channel that consists of several subchannels transmitting simultaneously and asynchronously. As an example of this scheme, consider a board with two chips (transmitter and receiver). The subchannels represent wires connecting between the chips where differences in the lengths of the wires might result in asynchronous reception.**

The contribution of this paper is a scheme which allows pipelined asynchronous communication at very high rates even when the amount of skew is arbitrarily large and unknown apriori. Insensitivity to delay is accomplished by encoding data in the frequency of the signal, as opposed to amplitude. The theoretical questions that are answered are what rates can be accomplished. In doing so we have extended the work of Capocelli and Spickerman on generalized Fibonacci numbers. The second question that we answer is how to encode data efficiently in the frequency of the signal. For the purposes of encoding and decoding we use variable length to variable length prefix-free codes. We have provided an algorithm based on integer linear programming for constructing such codes. In essence, we have formulated a scheme which is easy to implement and allows for asynchronous data transfer at very high rates. Potential applications are in on-chip, on-board and board to board communication, enabling much higher bandwidths.

Keywords— **Delay insensitive communication, parallel asynchronous communication, skew, pipelined channel, frequency modulation, Fibonacci Numbers, Generalized Fibonacci Numbers, sum of Fibonacci Numbers, sum of Generalized Fibonacci Numbers, prefix-free codes, prefix codes, variable length to variable length prefix-free codes, variable length to variable length prefix codes.**

I. Introduction

A. Motivation and Background

Consider a communication channel that consists of several subchannels transmitting simultaneously. As an example of this scheme consider a board with two chips (transmitter and receiver), where the subchannels represent wires connecting the chips. The transmitter would like to transmit a binary vector using n-channels/wires. The propagation delay varies from wire to wire and the problem is to find an efficient communication scheme that will be delay insensitive. Clearly, this problem is very common and arises in every system that incorporates transmission of information over parallel lines. Currently there are two approaches for solving this problem in practice.

This work was partially supported by the Lee Center for Advanced Networking at the California Institute of Technology. Saleem Mukhtar is partially supported by a National Defense Science and Engineering Graduate Fellowship.

The authors can be reached via email at saleem@paradise.caltech.edu and bruck@paradise.caltech.edu. Or via normal mail at Mail Code 136-93, California Institute of Technology, Pasadena, CA 91125. USA.

A.1 Verhoeff's Scheme for Asynchronous Data Transfer

The first solution was formulated by Verhoeff [29]. He describes the forgoing physical model as a scheme in which the sender communicates with the receiver via parallel tracks by rolling marbles in them. Presence of a marble indicates a logical 1 and absence of a marble corresponds to a logical 0.

For expository purposes let us assume we have 6 tracks and the sender would like to transmit two messages. These messages are encoded using the following binary vectors – 110010 and 010111. Let us say the transmitter sends message one (110010) by rolling marbles simultaneously in track 1,2 and 5. He then transmits message two (010111) by rolling marbles simultaneously in track 2, 4, 5 and 6. But the tracks are different lengths and this can cause the following problem. The receiver will first receive a marble in track 6 (the shortest track), followed by marbles in track 1,2 and 5 and then receive the remaining marbles in tracks 2, 4 and 5. And thus she would be unable to decipher the received messages.

In order to circumvent the above described problem he proposes the following solution. The sender will transmit one message at a time. Once the receiver has received the message, she will send an acknowledgment. The transmitter can then send the next message. The relevant questions are what are the necessary and sufficient conditions for the receiver to be able to determine that she has received a complete codeword and now must send an acknowledgment? If we have n wires, at most how many different codewords can there be? And is there an efficient encoding and decoding algorithm?

He proves that the necessary and sufficient conditions are that the codewords be unordered [29]. Sperner's Lemma tells us that the largest unordered set of codewords is the set of codewords containing an equal number of ones and zeros [29], [25]. And Knuth presents an efficient encoding and decoding scheme [29], [18].

The problem with this scheme is that if we have n wires at most $O(n)$ bits can be transmitted between acknowledgment signals. Specifically pipelined utilization of the channels is not possible. As the need for bandwidth grows it becomes critical that we come up with a scheme that can do much better.

A.2 Blaum-Bruck Scheme for Asynchronous Data Transfer

The need for a scheme which permits pipelined utilization of the channels and thus data transfer at higher rates was first identified and addressed by Blaum and Bruck [4], [5], [6]. We will again use the same physical analogy to

describe their solution – namely a sender needs to transmit messages to a receiver by rolling marbles down parallel tracks.

As in the previous example, let us say that we have 6 tracks and the transmitter would like to transmit one of two messages which are encoded using the following binary vectors – 110010 and 010111. If the transmitter sends message one followed by message two, as was illustrated previously, the receiver might not be able to decipher the messages correctly since marbles might be received in a different order.

Their solution is to define the concept of skew, which is related to the number of marbles that can be received from the second message before the first message has been completely received. Given a bound on the amount of skew, they identify the sufficient and necessary conditions for skew-tolerant and skew-detecting codes and provide constructions which are optimal in a certain sense.

The problem with this scheme is that in real systems, rarely if ever, is the amount of skew known in advance. It is a function of the differences in the lengths of the tracks, i.e. the wires which will eventually be used to connect the chips. The problem then is to design a communication scheme which provides higher bandwidth by allowing pipelined utilization of the channels and which does not in any way depend on the amount of skew.

B. The New Paradigm

Both the Verhoeff scheme [29] as well as the Blaum-Bruck scheme [4], [5], [6] for asynchronous data transfer encode data in the amplitude of the signal, as is done in the case for synchronous data transfer. In both schemes, presence of a marble ($5V$) encodes a logical 1 whereas its absence ($0V$) encodes a logical 0. Consider the following example – the transmitter would like to transmit data to a receiver via three wires or subchannels. Initially the three signals in the three subchannels are synchronized with respect to one another. But since the lengths of the wires are different, by the time they reach the receiver, they will be skewed relative to one another, as is illustrated in Figure 1.1. The fact that the signals are skewed makes decoding the transmitted messages, impossible. To deal with the fact that the system is asynchronous Verhoeff has to introduce acknowledgment [29], and Blaum-Bruck [4], [5], [6] introduce skew-tolerant and skew-detecting codes.

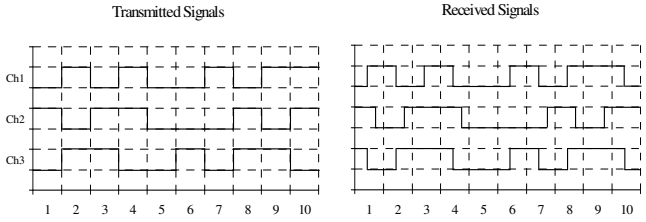


Fig. 1.1. Amplitude Modulation for Asynchronous Data Transfer

Our idea is that in the case of asynchronous data transfer, data should be encoded in the frequency of the signal as opposed to amplitude. The specific proposal is that each signal will be a sequence of spikes. Adjacent spikes will be separated by 1, 2, 3, ..., or K units of time. And information will be encoded in the time interval between adjacent spikes. Let us illustrate this with a specific example. Assume $K = 2$ and that a time interval of 1 unit between adjacent spikes encodes a 0 and a time interval of 2 units encodes a 1[1]. Figure 1.2 illustrates what happens when a transmitter sends messages to a receiver via three asynchronous subchannels. Notice that initially the signals are synchronized relative to one another. But by the time they propagate to the receiver, they are skewed relative to one another since the wires are different lengths. Although the signals are skewed relative to one another, the signals are not distorted and the time interval between adjacent spikes is preserved. The decoder can now decode the data simply by measuring the time interval between adjacent spikes. If the time interval between the first spike and the second spike is 1 unit then the first bit must be a 0, else it must have been a 1. If the time interval between the second and third spike is a 1, the second bit must have been a 0, else it must have been a 1, and so on and so forth.

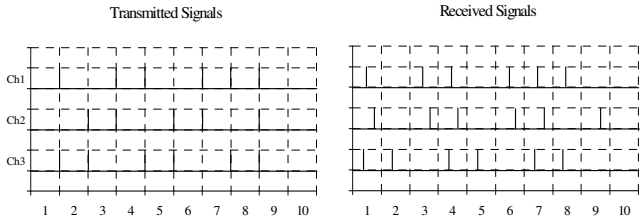


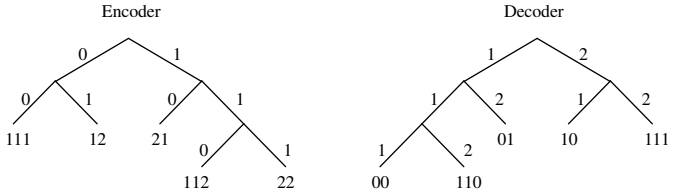
Fig. 1.2. Frequency Modulation for Asynchronous Data Transfer

As is apparent from the example above, differences in the lengths of wires or skew does not really pose a problem. And in fact transmitting information across multiple wires does not pose any additional problems than transmitting information through a single wire using frequency modulation. Hence in this paper, we will focus on this simpler case. It must be noted that if we used the scheme described above to transmit an n-bit binary vector consisting of n 1s, it would require $2n$ time units to transmit. All other n-bit binary vectors require less than $2n$ units of time to transmit. Hence, in the worst case, this scheme accomplishes an expansion (E) of 2. Next we illustrate how we use prefix-free codes to do much better. Again assume $K = 2$. Now consider the following scheme – if the binary vector to be transmitted starts with a 00, transmit 111, if it starts with 01 transmit 12, if it starts with 10, transmit a 21, if it starts with 110 transmit a 112 and if it starts with a 111, transmit 22. An inverse of this mapping can

[1]A simple generalization for $K = 3$ would be a time interval of 1 unit encodes a 0, a time interval of 2 units encodes a 10 and a time interval of 3 units encodes a 11.

be used for decoding[2]. The code is shown in Table 1.2 (D).

Example 1.1: The encoders and decoders based on the variable length to variable length prefix-free code in Table 1.2 (D) have been implemented using prefix trees below



Encoder and Decoder Implemented using Prefix Trees

Using the encoder and decoder above, we would like to transmit 01110100011011101001011000111....

The encoder parses this binary sequence as (01)(110)(10)(00)(110)(111)(01)(00)(10)(110)(00)(111)....

And encodes it as (12)(112)(21)(111)(112)(22)(12)(111)(21)(112)(111)(22)....

The decoder interprets this as (01)(110)(10)(00)(110)(111)(01)(00)(10)(110)(00)(111)....

In the code described above the source messages consists of 2 or 3 source alphabets and the codewords consist of 2 or 3 code alphabets. Since neither do all the source messages have the same number of source alphabets nor do all the codewords have the same number of code alphabets, we will refer to this as a variable length to variable length prefix-free code. It must be noted that this scheme accomplishes a worst case expansion of 1.5. Also the delay associated with the encoder is 3 units since it has to examine atmost 3 bits before it can encode part of the binary vector – in case the binary vector starts with 110 or 111. Similarly the delay associated with the decoder is 4 units since it might have to wait 4 units of time before it can decode part of the transmitted message – in case it receives a 112 or 22. The questions that need to be addressed are, given a value of K, what is the minimum achievable expansion, E^*? Given K, the desired expansion E and the delays associated with the encoder and decoder, how to design the smallest variable length to variable length prefix-free code, or equivalently the smallest encoder and decoder (in terms of number of leafs)?

[2]In this paper we assume that the vector to be transmitted is infinite. There is a minor problem if we need to transmit vectors of finite length, e.g. 011011. 01 would be encoded as a 12, 10 would be encoded as a 21, leaving a 11 which cannot be encoded. This problem is relatively easy to address. If the transmitter and receiver both know in advance how many bits are to be transmitted, the transmitter can simply insert a padding of 0s. Thus it would pad the remaining 11 with a 0 and transmit 112 which is the code for 110. The decoder gets a 12 which it interprets as a 01, then it receives a 21 which it interprets as 10, followed by 112 which it interprets as 110. Since both the receiver and transmitter agreed before hand that only 6 bits were to be transferred and the receiver has received 0110110 which is 7 bits, it simply trashes the last bit (the padding).

C. Contribution and Organization

For the theoretical parts of the paper we will make the arbitrary assumption that the coding alphabet is $\{1, 2, 3, ...K\}$. But our algorithm can be trivially extended to work for all coding alphabets. In Section II we determine the minimum achievable expansion as a function of K. In doing so we have extended the results of Spickerman and Joyner [26], [27] and Capocelli and Cull [7] who derive expressions for the nth generalized Fibonacci number (order K) in terms of the golden mean (or its analog for $K \geq 3$). We derive an expression for the sum of the first n generalized Fibonacci numbers (order K) in terms of the golden mean (or its analog for $K \geq 3$) and K. From a pragmatic point of view, the result we establish is that even for relatively small K, the minimum achievable expansion is very close to 1. We have tabulated the minimum achievable expansion for some small values of K in Table 1.1. We have already seen frequency modulation can be used for pipelined asynchronous data transfer, but amplitude modulation cannot. This means that the price we pay for encoding the data in the frequency of the signal is negligible, provided we can encode the data in a near optimal manner.

K	2	3	4	5	6	7
E^*	1.440	1.137	1.056	1.025	1.012	1.005

TABLE 1.1
Minimum Achievable Expansion for Small Values of K

For the purposes of encoding and decoding, we use variable length to variable length prefix-free codes. Block to variable length prefix-free codes have been well studied in literature [20]. Huffman [14] studied the problem of constructing an optimal block to variable length prefix-free code when the probabilities of the source alphabets are different and the transmission costs of the code alphabets are the same. The objective function he minimized was expected transmission time. Varn [28], Perl et. al. [24], Cot [8], Kapoor et. al. [15] and Golin et. al. [12] have studied the problem of constructing an optimal block to variable length prefix-free code when the probabilities of the source alphabets are the same but the transmission costs of the code alphabets are different. Again the objective function they minimized was expected transmission time. Blachman [3], Marcus [22], Karp [16], Krause [19], Cot [9], Mehlhorn [23], Altenkamp et. al. [2], Gilbert [11] and Golin et. al. [13] have studied the more complex case in which both the source alphabet probabilities as well as the transmission costs of the code alphabets are different. They also minimize expected transmission time. Our problem is most similar to the problem studied in [28], [24], [8], [15] and [12] – since we assume that the probabilities of the source alphabets are the same and the transmission cost of the code alphabets are different.

A	B	C	D
$K=2$ $R=2.00$	$K=2$ $R=2.00$	$K=2$ $R=1.66$	$K=2$ $R=1.50$
$0 \leftrightarrow 1$ $1 \leftrightarrow 2$	$00 \leftrightarrow 1111$ $01 \leftrightarrow 112$ $10 \leftrightarrow 121$ $11 \leftrightarrow 2$	$000 \leftrightarrow 11111$ $001 \leftrightarrow 1112$ $010 \leftrightarrow 1121$ $011 \leftrightarrow 1211$ $100 \leftrightarrow 2111$ $101 \leftrightarrow 122$ $110 \leftrightarrow 212$ $111 \leftrightarrow 221$	$00 \leftrightarrow 111$ $01 \leftrightarrow 12$ $10 \leftrightarrow 21$ $110 \leftrightarrow 112$ $111 \leftrightarrow 22$

TABLE 1.2
SOME BLOCK TO VARIABLE LENGTH PREFIX-FREE CODES (A,B,C) AND A VARIABLE LENGTH TO VARIABLE LENGTH PREFIX-FREE CODE (D)

The fundamental difference between our work and the work described above is that we use variable length to variable length prefix-free codes. Block to variable length prefix-free codes are special cases of variable length to variable length prefix-free codes. They have the added restriction that all source messages are of equal length – this assumption for our purposes is arbitrary, restrictive and unnecessary. In Table 1.2 (A), Table 1.2 (B) and Table 1.2 (C) we present a block to variable length prefix-free codes and in Table 1.2 (D) we present a variable length to variable length prefix-free code. It must be noted that the block to variable length prefix-free code in Table 1.2 (C) has 8 rules and accomplishes an expansion of 1.66 whereas the variable length to variable length prefix-free code in Table 1.2 (D) has 5 rules and accomplishes an expansion of 1.5[3]. In fact, the smallest block to variable length prefix-free code that accomplishes an expansion of 1.5 has a block size of 12 bits or equivalently the code has 4096 rules! A secondary difference is that we are not interested in minimizing the expected transmission time, but interested in finding the smallest code that can achieve a given worst case expansion.

The problem of encoding and decoding is also closely related to the problem of mapping arbitrary binary sequences to binary sequences which satisfy a (d,k) run-length limited constraint. A binary sequence is said to satisfy a (d,k) run-length limited constraint if and only if the number of consecutive zeros between two ones is between d and k. The problem of mapping binary sequences to binary sequences which satisfy the (d,k) run-length limited constraint arises in the context of magnetic data storage and has been studied extensively in literature. Adler, et. all [1] applied techniques from Symbolic Dynamics [21] to produce finite state encoders which could be used for coding and decoding. Alternative techniques were introduced in [17] and [10].

We will present our algorithm for constructing optimal variable length to variable length prefix-free codes in Section III. From a pragmatic point of view,the result we establish is that relatively small variable length to variable length prefix-free codes or equivalently relatively small encoders and decoders, even for small K, can achieve very low expansions. In Table 1.3 we have tabulated the performance of some encoders and decoders all of which have less than a 100 leafs as a function of K. A more detailed description of these results can be found in Section IV.

K	2	3	4	5	6	7
E	1.500	1.167	1.125	1.111	1.100	1.090

TABLE 1.3
EXPANSION THAT CAN BE ACHIEVED USING ENCODERS AND DECODERS HAVING LESS THAN A 100 LEAFS FOR SMALL VALUES OF K

The big picture is that we have formulated an efficient, high bandwidth scheme which can be used to transfer large amounts of data in a pipelined manner even if the differences in the lengths of the wires or skew is arbitrarily large and unknown a priori.

II. Rate Analysis for Frequency Modulation

Notice that a signal can be represented by a sequence of positive integers, $t_1, t_2, t_3, t_4, ...$, where t_i is the time interval between the ith and $(i+1)$th spike. Thus for all i, t_i belongs to $\{1, 2, 3, ..., K-1, K\}$. The question we seek to answer is, given K, in T units of time, how many bits can be transmitted. We will refer to this quantity as $M(T)$. We define the minimum achievable expansion $E^* = \lim_{T\to\infty} T/M(T)$. The plan is to first determine the number of sequences of length T whose elements are in $\{1, 2, 3, ..., K-1, K\}$. We will refer to this as $N_K(T)$. After that we will determine the number of sequences (excluding the null sequence) whose length is less than or equal to T and whose elements are in $\{1, 2, 3, ..., K-1, K\}$. We will refer to this as $\bar{N}_K(T)$. This will enable us to compute the number of bits, $M(T)$, that can be represented using sequences of length less than or equal to T. This will enable us to compute the minimum achievable expansion, E^*. But first, we will define the generalized Fibonacci numbers (order K) and their sum.

Definition 2.1: We define $F_K(n)$ to be the nth Fibonacci number (order K). And we define $\bar{F}_K(n)$ to be the sum of the first n Fibonacci numbers (order K).

$$F_K(n) = \begin{cases} \sum_{i=1}^{K} F_K(n-i) & n \geq K \\ 2^{n-2} & 2 \leq n \leq K-1 \\ 1 & n = 1 \\ 0 & n = 0 \end{cases}$$

$$\bar{F}_K(n) = \sum_{i=1}^{n} F_K(i) = \sum_{i=0}^{n} F_K(i)$$

[3]It is easy to prove that we cannot accomplish an expansion better than 1.66 using a block (block size 3) to variable length prefix-free code.

Next we will present a recursive formula for computing $N_K(T)$ and show that it is related to the generalized Fibonacci numbers (order K). Then we will present a known closed form formula for computing the nth generalized Fibonacci number (order K).

Theorem 2.1: $N_K(T) = \begin{cases} \sum_{i=1}^{K} N_K(T-i) & T > K \\ 2^{T-1} & 1 \le T \le K \end{cases}$

Proof: When $T \le K$, each element in the sequence will automatically be less than K. Hence, $N_K(T)$ is just the number of compositions of T, which is 2^{T-1}.

Assume $T > K$. The number of sequences of length T in which the last element is a 1, is $N_K(T-1)$, the number of sequences of length T in which the last element is a 2, is $N_K(T-2)$, and so on and so forth. Hence the total number of sequences of length T is $N_K(T-1) + ... + N_K(T-K)$. ■

Corollary 2.1.1: $N_K(T) = F_K(T+1)$

Proof: It is a consequence of the Definition 2.1 and Theorem 2.1. It can be proven formally by induction. ■

Theorem 2.2: Let $p(\lambda) = \lambda^K - \sum_{i=0}^{K-1} \lambda^i$

Let $\phi_i^{(K)}$be the distinct roots of $p(\lambda) = 0$. (Furthermore, we will assume that $\phi_1^{(K)}$is the unique positive real root of $p(\lambda) = 0$). Define $\alpha_i^{(K)}$ as follows

$$\alpha_i^{(K)} = \frac{\phi_i^{(K)} - 1}{\phi_i^{(K)}\left[(K+1)\phi_i^{(K)} - 2K\right]}$$

Then[4]

$$F_K(n) = \sum_{i=1}^{K} \alpha_i^{(K)} \left[\phi_i^{(K)}\right]^n = \left\langle \alpha_1^{(K)} \left[\phi_1^{(K)}\right]^n \right\rangle$$

Proof: This was proven by Capocelli and Cull [7]. ■

Next we will show that $\bar{N}_K(T)$ is related to the cumulative sum of generalized Fibonacci numbers (order K). Having done so, we will present a recursive as well as a closed form formula in terms of the golden mean and K, for computing the sum of the first n generalized Fibonacci numbers (order K).

Theorem 2.3: $\bar{N}_K(T) = \sum_{i=1}^{T} N_K(i) = \bar{F}_K(T+1) - 1$

Proof: It follows trivially from Corollary 2.1.1 and Definition 2.1. ■

Theorem 2.4: $\bar{F}_K(n)$ can be computed recursively as follows,

$$\bar{F}_K(n) = \begin{cases} 1 + \sum_{i=1}^{K} \bar{F}_k(n-i) & n \ge K \\ 2^{n-1} & 1 \le n \le K-1 \\ 0 & n = 0 \end{cases}$$

Proof: The proof is by induction. First we prove the base cases. Proof for $n = 0$ is obvious.

Case 1: $1 \le n \le K-1$

$\bar{F}_K(n) = \sum_{i=0}^{n} F_K(i) = F_K(0) + F_K(1) + \sum_{i=2}^{n} F_K(i)$

$= 0 + 1 + \sum_{i=2}^{n} 2^{i-2} = 1 + 2^{n-1} - 1 = 2^{n-1}$

Case 2: $n = K$

$\bar{F}_K(K) = \bar{F}_K(K-1) + F_K(K)$

$= \bar{F}_K(K-1) + \sum_{i=1}^{K} F_k(K-i)$

$= \bar{F}_K(K-1) + \sum_{i=0}^{K-1} F_K(i) = 2\bar{F}_K(K-1) = 2^{K-1}$

$1 + \sum_{i=1}^{K} \bar{F}_k(K-i) = 1 + \sum_{i=0}^{K-1} \bar{F}_k(i) = 1 + 2^{K-1} - 1 = 2^{K-1}$

Case 3: $n \ge K+1$

Assume: $\bar{F}_K(n) = 1 + \sum_{i=1}^{K} \bar{F}_k(n-i)$

$\bar{F}_K(n+1) = \bar{F}_K(n) + F_K(n+1)$

$\bar{F}_K(n+1) = 1 + \sum_{i=1}^{K} \bar{F}_k(n-i) + \sum_{i=1}^{K} F_k(n+1-i)$

$\bar{F}_K(n+1) = 1 + \sum_{i=1}^{K} \bar{F}_k(n+1-i)$ ■

Before we derive the closed form formula for $\bar{F}_k$, we need to prove several lemmas which will simplify the proof of the theorem. The first lemma involves a new idea and hence is presented here. The other lemmas are tedious and hence have been delegated to the appendix.

Lemma 2.5.1: $\sum_{i=1}^{K} \alpha_i^{(K)} \frac{1}{\phi_i^{(K)} - 1} = [K-1]^{-1}$

Proof: First let us define a new function G as follows

$$G_K(n) = \begin{cases} \sum_{i=1}^{K} G_K(n-i) & n \ge K \\ 2G_K(n-1) - 1 & 3 \le n < K \\ K & n = 2 \\ 1 & n = 1 \\ 1 & n = 0 \end{cases}$$

Next we will prove

$G_K(n) = \sum_{i=1}^{K} \gamma_i^{(K)} \left[\phi_i^{(K)}\right]^n$

where $\gamma_i^{(K)} = (K-1)\alpha_i^{(K)} \left(\phi_i^{(K)} - 1\right)^{-1}$

Since, G is a Fibonacci type recurrence having the same characteristic polynomial, it can be written as

$G_K(n) = \sum_{i=1}^{K} \gamma_i^{(K)} \left[\phi_i^{(K)}\right]^n$.

[4]Throughout this paper, the notation $\langle x \rangle$ will refer to x rounded to the nearest integer.

The only issue that needs to be addressed is the exact value of $\gamma_i^{(K)}$.

Case 1: $K = 2$ From Capocelli and Cull [7],
$R_i = \left[1, \phi_i^{(2)} - 1\right]$ and $I = [G_2(1), G_2(0)] = [1, 1]$

$$\gamma_i^{(2)} = \frac{R_i I\left(\phi_i^{(K)} - 1\right)}{\left[\phi_i^{(K)}\right]^{K-1}\left[(K+1)\,\phi_i^{(K)} - 2K\right]}$$

$$\gamma_i^{(2)} = \frac{\left[\phi_i^{(2)}\right]^2 - \left[\phi_i^{(2)}\right]}{\left[\phi_i^{(2)}\right]^{K-1}\left[(K+1)\,\phi_i^{(2)} - 2K\right]}$$

But since $p(\phi_i^{(2)}) = 0$ and $K = 2$,
$\left[\phi_i^{(2)}\right]^2 - \left[\phi_i^{(2)}\right] = 1 = \left[\phi_i^{(2)}\right]^{K-2}(K-1)$

$$\gamma_i^{(2)} = (K-1)\left(\phi_i^{(2)}\left[(K+1)\,\phi_i^{(2)} - 2K\right]\right)^{-1}$$

$$\gamma_i^{(2)} = (K-1)\alpha_i^{(2)}\left(\phi_i^{(2)} - 1\right)^{-1}$$

Case 2: $K \geq 3$ From Capocelli and Cull [7],
$R_i = [1, \left[\phi_i^{(K)}\right] - 1, ..., \left[\phi_i^{(K)}\right]^{K-1} ..... - \left[\phi_i^{(K)}\right] - 1]$
$I = [G_K(K-1), ..., G_K(1), G_K(0)]$

$$\gamma_i^{(K)} = \frac{R_i I\left(\phi_i^{(K)} - 1\right)}{\left[\phi_i^{(K)}\right]^{K-1}\left[(K+1)\,\phi_i^{(K)} - 2K\right]}$$

Let $\kappa_i^{(K)} = R_i I\left(\phi_i^{(K)} - 1\right)$

$$\kappa_i^{(K)} = \sum_{u=1}^{K}\left(\left[\phi_i^{(K)}\right]^u - 2\left[\phi_i^{(K)}\right]^{u-1} + 1\right)G_K(K-u)$$

But,

$$\sum_{u=1}^{K}\left[\phi_i^{(K)}\right]^u G_K(K-u)$$
$$= \left[\phi_i^{(K)}\right]^K G_K(0) + \sum_{u=0}^{K-1}\left[\phi_i^{(K)}\right]^u G_K(K-u) - G_K(K)$$
$$\sum_{u=1}^{K}\left[\phi_i^{(K)}\right]^{u-1} G_K(K-u) = \sum_{u=0}^{K-1}\left[\phi_i^{(K)}\right]^u G_K(K-u-1)$$
$$\sum_{u=1}^{K} G_K(K-u) = \sum_{u=0}^{K-1} G_K(u) = G_K(K)$$

Substituting,

$$\kappa_i^{(K)} = \left[\phi_i^{(K)}\right]^K G_K(0) + \sum_{u=0}^{K-1}\left[\phi_i^{(K)}\right]^u [G_K(K-u) - 2G_K(K-u-1)]$$

$$\kappa_i^{(K)} = \left[\phi_i^{(K)}\right]^K G_K(0) + \left[\phi_i^{(K)}\right]^{K-1}[G_K(1) - 2G_K(0)] + \left[\phi_i^{(K)}\right]^{K-2}[G_K(2) - 2G_K(1)] + \sum_{u=0}^{K-3}\left[\phi_i^{(K)}\right]^u [G_K(K-u) - 2G_K(K-u-1)]$$

When $K-3 \geq u \geq 1$, $[G_K(K-u) - 2G_K(K-u-1)] = -1$ (by definition). It is easy to show that when $u = 0$, $[G_K(K-u) - 2G_K(K-u-1)] = -1$. This is only true for $K \geq 3$. (that's why we addressed $K = 2$ separately)

$$\kappa_i^{(K)} = \left[\phi_i^{(K)}\right]^K - \sum_{u=0}^{K-1}\left[\phi_i^{(K)}\right]^u + (K-1)\left[\phi_i^{(K)}\right]^{K-2}$$

But since $p(\phi_i^{(2)}) = 0$,
$\left[\phi_i^{(K)}\right]^K - \sum_{u=0}^{K-1}\left[\phi_i^{(K)}\right]^u = 0$

Hence,

$$\kappa_i^{(K)} = (K-1)\left[\phi_i^{(K)}\right]^{K-2}$$
$$\gamma_i^{(K)} = (K-1)\left(\left[\phi_i^{(K)}\right]\left[(K+1)\,\phi_i^{(K)} - 2K\right]\right)^{-1}$$
$$\gamma_i^{(K)} = (K-1)\alpha_i^{(K)}\left(\phi_i^{(K)} - 1\right)^{-1}$$

Now note that,
$G_K(0) = 1$

Also,

$$G_K(0) = \sum_{i=1}^{K}\gamma_i^{(K)} = (K-1)\sum_{i=1}^{K}\alpha_i^{(K)}\frac{1}{\phi_i^{(K)} - 1}$$

Hence,

$$\sum_{i=1}^{K}\alpha_i^{(K)}\frac{1}{\phi_i^{(K)} - 1} = [K-1]^{-1}$$

■

Theorem 2.5: Let $p(\lambda) = \lambda^K - \sum_{i=0}^{K-1}\lambda^i$

Let $\phi_i^{(K)}$be the distinct roots of $p(\lambda) = 0$. (Furthermore, we will assume that $\phi_1^{(K)}$is the unique positive real root of $p(\lambda) = 0$). Define $\beta_i^{(K)}$ as follows

$$\beta_i^{(K)} = \frac{\alpha_i^{(K)}\phi_i^{(K)}}{\phi_i^{(K)} - 1} = \frac{1}{(K+1)\,\phi_i^{(K)} - 2K}$$

Then

$$\bar{F}_K(n) = \left\langle \beta_1^{(K)}\left[\phi_1^{(K)}\right]^n - [K-1]^{-1}\right\rangle$$

Proof: $\bar{F}_K(n) = \sum_{i=0}^{n} F_K(i) = \sum_{i=1}^{K}\alpha_i^{(K)}\sum_{j=0}^{n}\left[\phi_i^{(K)}\right]^j$

$$\bar{F}_K(n) = \sum_{i=1}^{K}\alpha_i^{(K)}\frac{\left[\phi_i^{(K)}\right]^{n+1} - 1}{\phi_i^{(K)} - 1}$$

$$\bar{F}_K(n) = \sum_{i=1}^{K}\alpha_i^{(K)}\frac{\left[\phi_i^{(K)}\right]^{n+1}}{\phi_i^{(K)} - 1} - \sum_{i=1}^{K}\alpha_i^{(K)}\frac{1}{\phi_i^{(K)} - 1}$$

By Lemma 2.5.1,

$$\bar{F}_K(n) = \sum_{i=1}^{K}\alpha_i^{(K)}\frac{\left[\phi_i^{(K)}\right]^{n+1}}{\phi_i^{(K)} - 1} - [K-1]^{-1}$$

Substituting,

$$\bar{F}_K(n) = \sum_{i=1}^{K}\beta_i^{(K)}\left[\phi_i^{(K)}\right]^n - [K-1]^{-1}$$

Let,

$$d_K(n) = \sum_{i=2}^{K}\beta_i^{(K)}\left[\phi_i^{(K)}\right]^n$$

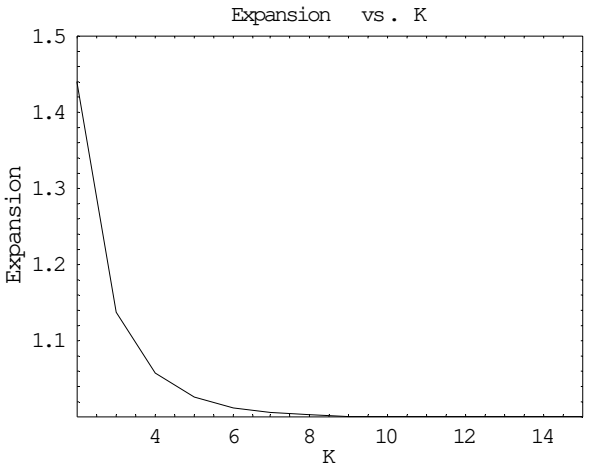


Fig. 2.1. Minimum Achievable Expansion E^* as a Function of K

Hence,
$\bar{F}_K(n) = \beta_1^{(K)} \left[\phi_1^{(K)}\right]^n + d_K(n) - [K-1]^{-1}$
By Lemma 2.5.2, 2.5.3 and 2.5.4 (see appendix),
$\bar{F}_K(n) = \left\langle \beta_1^{(K)} \left[\phi_1^{(K)}\right]^n - [K-1]^{-1} \right\rangle$ ■

Corollary 2.5.1: The closed form formula for $\bar{N}_K(T)$ is,
$\bar{N}_K(T) = \left\langle \beta_1^{(K)} \left[\phi_1^{(K)}\right]^{T+1} - K[K-1]^{-1} \right\rangle$

Proof: Follows trivially from Theorem 2.3 and Theorem 2.5. ■

Now that we have a closed form expression for the number of sequences whose length is less than or equal to T, we are ready to establish a tight bound on the minimum achievable rate, E^*. This is done so in the theorem below.

Theorem 2.6: $M(T) \approx \left\lfloor \log_2 \beta_1^{(K)} + (T+1)\log_2 \phi_1^{(K)} \right\rfloor$
$E^* = \lim_{T\to\infty} T/M(T) = 1/\log_2 \phi_1^{(K)}$

Proof: Follows trivially from Corollary 2.5.1. ■

We have plotted the minimum achievable expansion E^* versus K in Figure 2.1. It is evident, that as K is increased E^* falls exponentially to 1. From a pragmatic point of view, this establishes that even for relatively small K ($K = 6$), the minimum achievable expansion is very close to 1 (1.012). We have already seen that frequency modulation permits pipelined utilization of the channel, but amplitude modulation doesn't. This establishes that the price we pay for this advantage can be negligible (provided we can encode and decode data in a near optimal manner).

III. Encoding and Decoding using Variable Length to Variable Length Prefix-Free Codes

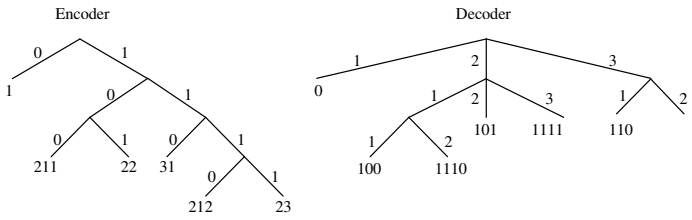
Now that we have established what rates can be accomplished using frequency modulation, we need to design encoding and decoding algorithms. In practice, the size of the binary vectors that we would like to encode can be very large. And in fact, the entire vector might not be available at the time transmission is to commence. So we will focus in on online encoding and decoding. Specifically, we will use variable length to variable length prefix-free codes. Before we define the problem formally and present our algorithm we will present some notation.

Notation 3.1: N is the set of Natural Numbers
A is the alphabet set and we will assume $A \subseteq N$
Σ_A^+ is the set of non-null strings over A
For all $x \in \Sigma_A^+$, $|x|$ is the number of alphabets in x
For all $x \in \Sigma_A^+$, $\|x\|$ is the sum of the alphabets in x
$\Pi_A^l = \{x | x \in \Sigma_A^+ \text{ and } \|x\| = l\}$
$\Sigma_A^l = \{x | x \in \Sigma_A^+ \text{ and } |x| = l\}$
Assume $s \in \Sigma_A^+$ and $l \geq \|s\|$. We define $P_A(s,l) = \{x | x \in \Sigma_A^+ \text{ and } \|x\| = l \text{ and } s \text{ is a prefix of } x\}$
Assume $s \in \Sigma_A^+$ and $l \geq |s|$. We define $Q_A(s,l) = \{x | x \in \Sigma_A^+ \text{ and } |x| = l \text{ and } s \text{ is a prefix of } x\}$

Example 3.1: Let $A = \{1,2,3\}$. Let $x = 123221$. $|x| = 6$ and $\|x\| = 1+2+3+2+2+1 = 11$. $\Pi_A^3 = \{111, 12, 21\}$ and $\Sigma_A^2 = \{11, 12, 13, 21, 22, 23, 31, 32, 33\}$. $P_A(12,5) = \{1211, 122\}$ and $Q_A(12,3) = \{121, 122, 123\}$.

Definition 3.1: Let S be the source alphabet and C be the code alphabet. Let $M_S \subseteq \Sigma_S^+$ such that M_S is a full prefix-free set and $M_C \subseteq \Sigma_C^+$ such that M_C is a prefix-free set (it need not be full). A one to one mapping from M_S to M_C is a variable length to variable length prefix-free code. This mapping will referred to as the encoding function, ξ, and its inverse, ξ^{-1}, is the corresponding decoding function. (Note that M_S must be a full prefix-free set in order to ensure every source message can be encoded).

Example 3.2: Let $S = \{0,1\}^5$ and $C = \{1,2,3\}$. And $M_S = \{0, 100, 101, 110, 1110, 1111\}$ and $M_C = \{1, 211, 22, 31, 212, 23\}$. We define $\xi : M_S \to M_C$ such that $\xi(0) = 1$, $\xi(100) = 211$, $\xi(101) = 22$, $\xi(110) = 31$, $\xi(1110) = 212$ and $\xi(1111) = 23$. Note that ξ is a variable length to variable length prefix-free code. In the diagram below we have shown the encoding and decoding function, ξ and ξ^{-1}, implemented as prefix trees.



Encoder and Decoder Implemented using Prefix Trees

It must be noted that the decoder is not a complete tree since no symbols is attached to 32. But this is not a problem since 32 is not a symbol in the encoder and hence will

[5]In this paper we are assuming that the vector to be transmitted is in base 2. The algorithms presented in this paper can be generalized trivially to deal with higher bases.

never be transmitted. In the worst case this variable length to variable length prefix-free code accomplishes an expansion, E, of 1.33. Also the encoder has to examine at most 4 bits (in case the binary vector starts with a 1110 or 1111) before it can encode part of the vector to be transmitted. Hence, the delay associated with the encoder, T_E is 4 time units. Similarly the decoder might have to wait upto 5 time units (in case it receives a 212, 23 or 32) before it can decode part of the received message. Hence, the delay associated with the decoder, T_D is 5 time units.

Problem 3.1: We are given K, the desired expansion E, and the maximum permissible delays associated with the encoder, T_E and the maximum permissible delay associated with the decoder T_D. The problem is to design the smallest variable length to variable length prefix-free code which achieves the desired expansion (or better) and has the given delays (or better). That is we need to find $M_S \subseteq \Sigma_S^+$ such that M_S is a full prefix-free set and $M_C \subseteq \Sigma_C^+$ such that M_C is a prefix-free set (it need not be full). And a one to one mapping from M_S to M_C which we will refer to as the encoding function ξ. The criteria we are minimizing is $|M_S| = |M_C|$.

Let $l \in \{1, 2, ..., T_D\}$ and $d \in \{1, 2, ..., T_E\}$. Let x_d^l be the number of leafs (non negative integer) in the encoder at depth d which have a symbol of length l. Formally,

$$x_d^l = |\{x | x \in M_S \text{ and } |x| = d \text{ and } \|\xi(x)\| = l\}|$$
$$\text{For all } l \text{ and } d,\ x_d^l \geq 0$$

Also, we define x^l to be the number of symbols of length l and we define x_d to be the number of leafs in the encoder at depth d. Formally,

$$x^l = |\{x | x \in M_C \text{ and } \|x\| = l\}|$$
$$x_d = |\{x | x \in M_S \text{ and } |x| = d\}|$$
$$x^l = \sum_{i=1}^{T_E} x_i^l \text{ and } x_d = \sum_{i=1}^{T_D} x_d^i$$

The problem is to design the smallest encoder decoder pair. So we would like to,

$$\min \sum_{i=1}^{T_D} \sum_{j=1}^{T_E} x_j^i$$

Of course the values taken on by x_j^i cannot be arbitrary. We know that the encoder must form a full prefix tree and the decoder must form a prefix tree. Furthermore ξ must accomplish an expansion of E. These conditions impose certain constraints on the values that can be taken on by x_j^i.

Theorem 3.1: We are given M_S which is a full prefix-free set. $x_1, ..., x_{T_E}$ must satisfy the following constraint

$$\sum_{i=1}^{T_E} 2^{(T_E - i)} x_i = 2^{T_E}$$

Proof: We know that $\bigcup_{s \in M_S} Q_S(s, T_E) \subseteq \Sigma_S^{T_E}$. Also $\Sigma_S^{T_E} \subseteq \bigcup_{s \in M_S} Q_S(s, T_E)$. Assume this were not the case. Hence, there exists an $x \in \Sigma_S^{T_E}$ such that $x \notin \bigcup_{s \in M_S} Q_S(s, T_E)$. Hence there does not exist an $s \in M_S$ such that s is a prefix of x. Hence M_S is not full. This is a contradiction. Hence, $\bigcup_{s \in M_S} Q_S(s, T_E) = \Sigma_S^{T_E}$.

$$\text{Hence,} \left|\bigcup_{s \in M_S} Q_S(s, T_E)\right| = \left|\Sigma_S^{T_E}\right|$$

Assume $s_1,\ s_2 \in \Sigma_S^+$ and $|s_1| \leq L$ and $|s_2| \leq L$. We know that if s_1 is not a prefix of s_2 and s_2 is not a prefix of s_1 then $Q_S(s_1, L) \cap Q_S(s_2, L) = \phi$. Hence for all $s_1,\ s_2 \in M_S,\ Q_S(s_1, T_E) \cap Q_S(s_2, T_E) = \phi$. Also assume $s \in \Sigma_S^+$ and $|s| \leq L$. $|Q_S(s, L)| = 2^{L - |s|}$. Hence for all $s \in M_S$, $|Q_S(s, T_E)| = 2^{T_E - |s|}$.

$$\text{Hence, } \left|\bigcup_{s \in M_S} Q_S(s, T_E)\right| = \sum_{s \in M_S} |Q_S(s, T_E)|$$
$$= \sum_{s \in M_S} 2^{T_E - |s|} = \sum_{i=1}^{T_E} 2^{(T_E - i)} x_i$$

$$\text{Also } \left|\Sigma_S^{T_E}\right| = 2^{T_E}. \text{ Hence, } \sum_{i=1}^{T_E} 2^{(T_E - i)} x_i = 2^{T_E}. \quad \blacksquare$$

Theorem 3.2: We are given M_C which is a prefix-free set. $x^1, ..., x^{T_D}$, must satisfy the following constraints

$$\text{For all } 1 \leq L \leq T_D,\ \sum_{l=1}^{L} F_K(L - l + 1) x^l \leq F_K(L + 1)$$

Proof: We are given L, such that $1 \leq L \leq T_D$. Define $\bar{M}_C(L) = \{x | x \in M_C \text{ and } \|x\| \leq L\}$. We know that $\bigcup_{s \in \bar{M}_C(L)} P_C(s, L) \subseteq \Pi_C^L$

$$\text{Hence,} \left|\bigcup_{s \in \bar{M}_C(L)} P_C(s, L)\right| \leq \left|\Pi_C^L\right|$$

Since M_C is a prefix-free set and $\bar{M}_C(L) \subseteq M_C$, $\bar{M}_C(L)$ is prefix-free set. Assume $s_1,\ s_2 \in \Sigma_C^+$ and $\|s_1\| \leq L$ and $\|s_2\| \leq L$. We know that if s_1 is not a prefix of s_2 and s_2 is not a prefix of s_1 then $P_C(s_1, L) \cap P_C(s_2, L) = \phi$. Hence, for all $s_1,\ s_2 \in \bar{M}_C(L),\ P_C(s_1, L) \cap P_C(s_2, L) = \phi$. Also assume $s \in \Sigma_C^+$ and $\|s\| \leq L$. If $\|s\| < L$, $|P_C(s, L)| = N_K(L - \|s\|) = F_K(L - \|s\| + 1)$. If $\|s\| = L$, $|P_C(s, L)| = 1 = F_K(1) = F_K(L - \|s\| + 1)$. Hence $|P_C(s, L)| = F_K(L - \|s\| + 1)$.

$$\text{Hence, } \left|\bigcup_{s \in \bar{M}_C(L)} P_C(s, L)\right| = \sum_{s \in \bar{M}_C(L)} |P_C(s, L)|$$
$$= \sum_{s \in \bar{M}_C(L)} F_K(L - \|s\| + 1) = \sum_{l=1}^{L} F_K(L - l + 1) x^l$$

$$\text{Hence, } \left|\bigcup_{s \in \bar{M}_C(L)} P_C(s, L)\right| = \sum_{l=1}^{L} F_K(L - l + 1) x^l$$

$$\text{Also } \left|\Pi_C^L\right| = F_K(L + 1)$$

$$\text{Hence, } \sum_{l=1}^{L} F_K(L - l + 1) x^l \leq F_K(L + 1) \quad \blacksquare$$

Theorem 3.3: Since ξ accomplishes an expansion of E, x_j^i must satisfy the following constraint

$$i/j > E \implies x_j^i = 0$$

Proof: Assume there exists i and j such that $i/j > E$ and $x_j^i \neq 0$. Hence there exists $s \in M_S$ such that $|s| = j$

and $\|\xi(s)\| = i$. Hence $\|\xi(s)\|/|s| = i/j > E$. Hence ξ does not accomplish an expansion of E. This is a contradiction. ■

It must be noted that the objective function to be minimized as well as the constraints listed are all linear. We use standard integer linear programming techniques to compute the values of x_j^i that minimize the given objective function. We still need to establish that given the values of x_j^i, we can construct a variable length to variable length prefix-free code. This involves constructing M_S, M_C and ξ.

Theorem 3.4: Given $x_1, ..., x_{T_E}$ that satisfy the following constraint

$$\sum_{i=1}^{T_E} 2^{(T_E-i)} x_i = 2^{T_E}$$

We can construct M_S such that M_S is a full prefix-free set and for $1 \leq i \leq T_E$, $|\{s|s \in M_S \text{ and } |s| = i\}| = x_i$.

Proof: The proof is by induction.

Base Case: Number of variables is 1. Hence $T_E = 1$. Since, $\Sigma_{i=1}^{T_E} 2^{(T_E-i)} x_i = 2^{T_E}$, $x_1 = 2$. Let $M_S = \{0, 1\}$.

Inductive Hypothesis: Number of variables is T_E. We are given $x_1, ..., x_{T_E}$ such that $\Sigma_{i=1}^{T_E} 2^{(T_E-i)} x_i = 2^{T_E}$. We can construct M_S such that M_S is a full prefix-free set and for $1 \leq i \leq T_E$, $|\{s|s \in M_S \text{ and } |s| = i\}| = x_i$.

Now consider the case when the number of variables is $T_E + 1$. We know that $\Sigma_{i=1}^{T_E+1} 2^{(T_E+1-i)} x_i = 2^{T_E+1}$. Since 2^{T_E+1} is even, $\Sigma_{i=1}^{T_E+1} 2^{(T_E+1-i)} x_i$ must be even. $\Sigma_{i=1}^{T_E+1} 2^{(T_E+1-i)} x_i = x_{T_E+1} + 2\Sigma_{i=1}^{T_E} 2^{(T_E-i)} x_i$. Also $2\Sigma_{i=1}^{T_E} 2^{(T_E-i)} x_i$ is even, hence x_{T_E+1}must be even. For $i \in \{1, 2, ..., T_E - 1\}$ define $\bar{x}_i = x_i$ and $\bar{x}_{T_E} = x_{T_E} + x_{T_E+1}/2$. Now,

$$\sum_{i=1}^{T_E} 2^{(T_E-i)} \bar{x}_i = \bar{x}_{T_E} + \sum_{i=1}^{T_E-1} 2^{(T_E-i)} \bar{x}_i$$
$$= x_{T_E+1}/2 + x_{T_E} + \sum_{i=1}^{T_E-1} 2^{(T_E-i)} x_i$$
$$= \sum_{i=1}^{T_E+1} 2^{(T_E-i)} x_i = (\sum_{i=1}^{T_E+1} 2^{(T_E+1-i)} x_i)/2 = 2^{T_E}$$

By the inductive hypothesis we can construct $\bar{M}_S$ such that $\bar{M}_S$ is a full prefix-free set and for $1 \leq i \leq T_E$, $\left|\{s|s \in \bar{M}_S \text{ and } |s| = i\}\right| = \bar{x}_i$. Or equivalently, for $1 \leq i \leq T_E - 1$, $\left|\{s|s \in \bar{M}_S \text{ and } |s| = i\}\right| = x_i$ and $\left|\{s|s \in \bar{M}_S \text{ and } |s| = T_E\}\right| = x_{T_E} + x_{T_E+1}/2$. Define $U \subseteq \bar{M}_S$ such that $|U| = x_{T_E+1}/2$ and for all $u \in U$, $|u| = T_E$. Let $V = \{u0, u1 | u \in U\}$. Now let $M_S = (\bar{M}_S - U) \cup V$. It is easy to show that for $1 \leq i \leq T_E + 1$, $|\{s|s \in M_S \text{ and } |s| = i\}| = x_i$ and M_S is a full prefix-free set. ■

Theorem 3.5: We are given $x^1, ..., x^{T_D}$, and these variables satisfy the following constraint

For all $1 \leq L \leq T_D$, $\sum_{l=1}^{L} F_K(L - l + 1) x^l \leq F_K(L + 1)$

We can construct M_C such that M_C is prefix-free set and for $1 \leq i \leq T_D$, $|\{s|s \in M_C \text{ and } \|s\| = i\}| = x^i$.

Proof: The proof is by induction.

Base Case: Number of variables is 1. Hence $T_D = 1$. Hence $F_K(1)x^1 \leq F_K(2)$. Hence $x^1 \leq 1$. If $x^1 = 0$, let $M_C = \{\}$. If $x^1 = 1$, let $M_C = \{1\}$.

Inductive Hypothesis: Number of variables is T_D. We are given $x^1, ..., x^{T_D}$ such that for all $1 \leq L \leq T_D$, $\Sigma_{l=1}^{L} F_K(L - l + 1) x^l \leq F_K(L + 1)$. We will assume we can construct M_C such that M_C is prefix-free set and for $1 \leq i \leq T_D$, $|\{s|s \in M_C \text{ and } \|s\| = i\}| = x^i$.

Now consider the case when the number of variables is $T_D + 1$. For $i \in \{1, 2, ..., T_D\}$ define $\bar{x}^i = x^i$. For all $1 \leq L \leq T_D$, $\Sigma_{l=1}^{L} F_K(L - l + 1) \bar{x}^l \leq F_K(L + 1)$. Hence, by the inductive hypothesis we can construct $\bar{M}_C$ such that $\bar{M}_C$ is prefix-free set and for $1 \leq i \leq T_D$, $\left|\{s|s \in \bar{M}_C \text{ and } \|s\| = i\}\right| = \bar{x}^i = x^i$. We know that,

$$\sum_{l=1}^{T_D+1} F_K(T_D + 2 - l) x^l \leq F_K(T_D + 2)$$

Or equivalently,

$$F_K(1)x^{T_D+1} \leq F_K(T_D + 2) - \sum_{l=1}^{T_D} F_K(T_D + 2 - l) x^l$$

Hence, $x^{T_D+1} \leq F_K(T_D + 2) - \sum_{l=1}^{T_D} F_K(T_D + 2 - l) x^l$

Now let $U = \Pi_C^{T_D+1} - \bigcup_{s \in \bar{M}_C} P_C(s, T_D + 1)$

Note that $P_C(s, T_D + 1) \subseteq \Pi_C^{T_D+1}$

Hence, $|U| = \left|\Pi_C^{T_D+1}\right| - \left|\bigcup_{s \in \bar{M}_C} P_C(s, T_D + 1)\right|$

Assume s_1, $s_2 \in \Sigma_C^+$ and $\|s_1\| \leq T_D + 1$ and $\|s_2\| \leq T_D + 1$. We know that if s_1 is not a prefix of s_2 and s_2 is not a prefix of s_1 then $P_C(s_1, T_D + 1) \cap P_C(s_2, T_D + 1) = \phi$. Hence for all s_1, $s_2 \in \bar{M}_C$, $P_C(s_1, T_D + 1) \cap P_C(s_2, T_D + 1) = \phi$. Also assume $s \in \Sigma_C^+$ and $\|s\| < T_D + 1$. $|P_C(s, T_D + 1)| = N_K(T_D + 1 - \|s\|) = F_K(T_D + 2 - \|s\|)$.

$$\text{Hence,} \left|\bigcup_{s \in \bar{M}_C} P_C(s, T_D + 1)\right| = \sum_{s \in \bar{M}_C} |P_C(s, T_D + 1)|$$

$$= \sum_{s \in \bar{M}_C} F_K(T_D + 2 - \|s\|) = \sum_{l=1}^{L} F_K(T_D + 2 - l) x^l$$

$$\text{Hence, } \left|\bigcup_{s \in \bar{M}_C} P_C(s, T_D + 1)\right| = \sum_{l=1}^{L} F_K(T_D + 2 - l) x^l$$

$$\text{Also } \left|\Pi_C^{T_D+1}\right| = F_K(T_D + 2)$$

$$\text{Hence, } |U| = F_K(T_D + 2) - \sum_{l=1}^{L} F_K(T_D + 2 - l) x^l$$

Hence, $x^{T_D+1} \leq |U|$. It suffices to pick x^{T_D+1} elements from U and add them to $\bar{M}_C$ to generate M_C. ■

Theorem 3.6: Given M_S and M_C, we can construct $\xi : M_S \rightarrow M_C$, such that ξ is one to one function and $\max_{x \epsilon M_S} \|\xi(x)\|/|x| \leq E$.

Proof: We will construct a one to one function ξ by assigning x_j^i elements of size j from M_S to x_j^i elements of size i from M_C. It is easy to show that this can be done and ξ will achieve an expansion of E (or better). ■

Next we will illustrate how the techniques described above can be used to construct optimal encoders and decoders.

Example 3.3: We are given $K = 3$, $T_E = 5$, $T_D = 6$ and $E = 1.25$. We need to design the smallest encoder decoder pair which will satisfy the above constraints. Let $l \in \{1, 2, ..., T_D\}$ and $d \in \{1, 2, ..., T_E\}$. Let x_d^l be the number of leafs in the encoder at depth d which have a symbol of length l. For all l and d, $x_d^l \geq 0$. Without loss of generality we can assume that the non-zero variables are x_1^1, x_2^2, x_3^3, x_4^5 and x_5^6. (Assume $x_3^2 = 1$. That is we assigned a label of length 2 at depth 3. Assume the label is 11. Since M_C is prefix-free, 111, does not belong to M_C. Hence we could replace 11 by 111. This would set $x_3^2 = 0$ and increase x_3^3 by 1. The size of the code would be unaltered and it would still accomplish an expansion of E or better).

The problem now is to
$\min\ x_1^1 + x_2^2 + x_3^3 + x_4^5 + x_5^6$

subject to

$16x_1^1 + 8x_2^2 + 4x_3^3 + 2x_4^5 + x_5^6 = 32$
$x_1^1 \leq 1$
$x_1^1 + x_2^2 \leq 2$
$2x_1^1 + x_2^2 + x_3^3 \leq 4$
$7x_1^1 + 4x_2^2 + 2x_3^3 + x_4^5 \leq 13$
$13x_1^1 + 7x_2^2 + 4x_3^3 + x_4^5 + x_5^6 \leq 24$

The solution to this integer program is

$\{x_1^1 = 1, x_2^2 = 0, x_3^3 = 0, x_4^5 = 6, x_5^6 = 4\}$

We are given $x_1 = 1, x_2 = 0, x_3 = 0, x_4 = 6$ and $x_5 = 4$. We will first construct M_S such that M_S is a full prefix-free set and for $1 \leq i \leq T_E$, $|\{s | s \in M_S \text{ and } |s| = i\}| = x_i$. The proof of Theorem 3.4 suggests a recursive procedure. We first need to construct a set in which $x_1 = 1, x_2 = 0, x_3 = 0, x_4 = 8$. In order to this we need to construct a set in which $x_1 = 1, x_2 = 0, x_3 = 4$. In order to this we must construct a set in which $x_1 = 1, x_2 = 2$. And finally to do this we need to construct a set in which $x_1 = 2$. This is easy

$x_1 = 2$
$M_S = \{0, 1\}$

In order to generate a full prefix-free set in which $x_1 = 1, x_2 = 2$, we will select $x_2/2 = 2/2 = 1$ element of size from $\{0, 1\}$ and use it to generate 2 elements of size 2. We apply this technique repeatedly, till we have the desired solution.

$x_1 = 1, x_2 = 2$
$M_S = \{0, 10, 11\}$
$x_1 = 1, x_2 = 0, x_3 = 4$
$M_S = \{0, 100, 101, 110, 111\}$
$x_1 = 1, x_2 = 0, x_3 = 0, x_4 = 8$
$M_S = \{0, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111\}$
$x_1 = 1, x_2 = 0, x_3 = 0, x_4 = 6, x_5 = 4$
$M_S = \{0, 1000, 1001, 1010, 1011, 1100, 1101,$
$11100, 11101, 11110, 11111\}$

We are given $x^1 = 1, x^2 = 0, x^3 = 0, x^4 = 0, x^5 = 6$ and $x^6 = 4$. Now we construct M_C such that M_C is prefix-free set and for $1 \leq i \leq T_D$, $|\{s | s \in M_C \text{ and } \|s\| = i\}| = x^i$. The proof of Theorem 3.5 suggests the following procedure. First construct a prefix-free set M_C such that $x^1 = 1$. Next modify this set to construct a new set such that $x^1 = 1$ and $x^5 = 6$. Next modify this set to construct a new set such that $x^6 = 4$.

$x^1 = 1$
$M_C = \{1\}$

To generate a prefix-free set in which $x^1 = 1$ and $x^5 = 6$, we need to add 6 elements to M_C from $\Pi_C^5 - P_C(1, 5) = \{2111, 212, 221, 23, 311, 32\}$. A similar procedure is used to generate a prefix-free set in which $x^1 = 1, x^5 = 6$ and $x^6 = 4$. The steps are shown below.

$x^1 = 1$ and $x^5 = 6$
$M_C = \{1, 2111, 212, 221, 23, 311, 32\}$
$x^1 = 1, x^5 = 6$ and $x^6 = 4$
$M_C = \{1, 2111, 212, 221, 23, 311, 32, 2112, 213, 222, 312\}$

We have constructed M_S and M_C. In order to generate ξ we need to assign $x_1^1 = 1$ element of size 1 from M_S to 1 element of size 1 in M_C. We map $0 \leftrightarrow 1$. We need to assign $x_4^5 = 6$ elements of size 4 from M_S to 6 elements of size 5 in M_C. We map $1000 \leftrightarrow 2111, 1001 \leftrightarrow 212, 1010 \leftrightarrow 221$ and so on and so forth. The full code is shown in the table below

$0 \leftrightarrow 1$	$1011 \leftrightarrow 23$	$11101 \leftrightarrow 213$
$1000 \leftrightarrow 2111$	$1100 \leftrightarrow 311$	$11110 \leftrightarrow 222$
$1001 \leftrightarrow 212$	$1101 \leftrightarrow 32$	$11111 \leftrightarrow 312$
$1010 \leftrightarrow 221$	$11100 \leftrightarrow 2112$	

Example 3.4: We are given $K = 3$, $T_E = 9$, $T_D = 10$ and $E = 8/7 = 1.14$. We need to design the smallest encoder decoder pair which will satisfy the above constraints. Let $l \in \{1, 2, ..., T_D\}$ and $d \in \{1, 2, ..., T_E\}$. Let x_d^l be the number of leafs in the encoder at depth d which have a symbol of length l. For all l and d, $x_d^l \geq 0$. Without loss of generality we can assume that the non-zero variables are x_1^1, x_2^2, x_3^3, x_4^4, x_5^5, x_6^6, x_7^8, x_8^9 and x_9^{10}.

The problem now is to
$\min\ x_1^1 + x_2^2 + x_3^3 + x_4^4 + x_5^5 + x_6^6 + x_7^8 + x_8^9 + x_9^{10}$

subject to

$256x_1^1 + 128x_2^2 + 64x_3^3 + ... + 4x_7^8 + 2x_8^9 + x_9^{10} = 512$
$x_1^1 \leq 1$
$x_1^1 + x_2^2 \leq 2$
$2x_1^1 + x_2^2 + x_3^3 \leq 4$

$4x_1^1 + 2x_2^2 + x_3^3 + x_4^4 \leq 7$
$7x_1^1 + 4x_2^2 + 2x_3^3 + x_4^4 + x_5^5 \leq 13$
$13x_1^1 + 7x_2^2 + 4x_3^3 + 2x_4^4 + x_5^5 + x_6^6 \leq 24$
$44x_1^1 + 24x_2^2 + 13x_3^3 + 7x_4^4 + 4x_5^5 + 2x_6^6 + x_7^8 \leq 81$
$81x_1^1 + 44x_2^2 + 24x_3^3 + 13x_4^4 + 7x_5^5 + 4x_6^6 + x_7^8 + x_8^9 \leq 149$
$149x_1^1 + 81x_2^2 + 44x_3^3 + 24x_4^4 + 13x_5^5 + 7x_6^6 + 2x_7^8 + x_8^9 + x_9^{10} \leq 274$

This integer program has no solution. Hence, there does not exist a variable length to variable length prefix-free code which has the given delays and accomplishes the given expansion. It must be noted that the desired expansion of 1.14 is above the theoretically achievable expansion of 1.137.

IV. Results

In Table 4.1 and Table 4.2 we present some interesting encoders and decoders designed using the techniques outlined in the previous section. In Table 4.3 we summarize the dependence of the size of the encoder and decoder measured in the number of leafs versus K and the expansion achieved. The table is not exhaustive. Its purpose is to illustrate that using relatively small values of K and relatively small encoder decoders, one can accomplish very low expansions. Furthermore, it illustrates that if we fix K, we can improve the bandwidth by increasing the size of the encoder and decoder. Alternately, if we fix the desired expansion, we can decrease the size of the encoder and decoder simply by increasing K.

V. Conclusions

We have designed a pragmatic scheme for high bandwidth, pipelined asynchronous data transfer which can be used even when the amount of skew is unknown a priori. The novel idea here is to encode data in the frequency of the signal as opposed to amplitude.

We have established that even for relatively small values of K, this scheme is very efficient in terms of rate. In doing so we have extended the work of Capocelli et. al. [7] and Spickerman et. al. [26], [27]. We have derived a formula for the sum of the first n generalized Fibonacci numbers in terms of the golden mean (or its analog when $K \geq 3$) and K itself.

For the purposes of encoding and decoding, we first introduced the concept of a variable length to variable length prefix-free code. Given the delays associated with the encoder and decoder and the desired rate, we presented an algorithm which can construct the smallest variable length to variable length prefix-free code or equivalently the smallest encoder and decoder. Using the above outlined techniques, we have shown that even small encoders and decoders can be used to encode and decode the data efficiently.

$K=2$ $E=1.5$ $T_E=3$ $T_D=4$	$K=3$ $E=1.33$ $T_E=4$ $T_D=5$	$K=3$ $E=1.25$ $T_E=5$ $T_D=6$	$K=4$ $E=1.20$ $T_E=6$ $T_D=7$
$00 \leftrightarrow 111$	$0 \leftrightarrow 1$	$0 \leftrightarrow 1$	$0 \leftrightarrow 1$
$01 \leftrightarrow 12$	$100 \leftrightarrow 211$	$1000 \leftrightarrow 2111$	$10 \leftrightarrow 2$
$10 \leftrightarrow 21$	$101 \leftrightarrow 22$	$1001 \leftrightarrow 212$	$1100 \leftrightarrow 31$
$110 \leftrightarrow 112$	$110 \leftrightarrow 31$	$1010 \leftrightarrow 221$	$11010 \leftrightarrow 321$
$111 \leftrightarrow 22$	$1110 \leftrightarrow 212$	$1011 \leftrightarrow 23$	$11011 \leftrightarrow 33$
	$1111 \leftrightarrow 23$	$1100 \leftrightarrow 311$	$11100 \leftrightarrow 411$
		$1101 \leftrightarrow 32$	$11101 \leftrightarrow 42$
		$11100 \leftrightarrow 2112$	$111100 \leftrightarrow 322$
		$11101 \leftrightarrow 213$	$111101 \leftrightarrow 34$
		$11110 \leftrightarrow 222$	$111110 \leftrightarrow 412$
		$11111 \leftrightarrow 312$	$111111 \leftrightarrow 43$

TABLE 4.1
Some variable length to variable length prefix-free codes

$K=4$ $E=1.16$ $T_E=9$ $T_D=10$	$K=5$ $E=1.16$ $T_E=9$ $T_D=10$	$K=5$ $E=1.14$ $T_E=10$ $T_D=11$
$0 \leftrightarrow 1$	$0 \leftrightarrow 1$	$0 \leftrightarrow 1$
$10 \leftrightarrow 2$	$10 \leftrightarrow 2$	$10 \leftrightarrow 2$
$1100 \leftrightarrow 31$	$110 \leftrightarrow 3$	$110 \leftrightarrow 3$
$110100 \leftrightarrow 3211$	$11100 \leftrightarrow 41$	$11100 \leftrightarrow 41$
$110101 \leftrightarrow 322$	$111010 \leftrightarrow 421$	$1110100 \leftrightarrow 4211$
$110110 \leftrightarrow 331$	$111011 \leftrightarrow 43$	$1110101 \leftrightarrow 422$
$110111 \leftrightarrow 34$	$111100 \leftrightarrow 511$	$1110110 \leftrightarrow 431$
$111000 \leftrightarrow 4111$	$111101 \leftrightarrow 52$	$1110111 \leftrightarrow 44$
$111001 \leftrightarrow 412$	$1111100 \leftrightarrow 422$	$1111000 \leftrightarrow 5111$
$111010 \leftrightarrow 421$	$1111101 \leftrightarrow 44$	$1111001 \leftrightarrow 512$
$111011 \leftrightarrow 43$	$1111110 \leftrightarrow 512$	$1111010 \leftrightarrow 521$
$1111000 \leftrightarrow 3212$	$1111111 \leftrightarrow 53$	$1111011 \leftrightarrow 53$
$1111001 \leftrightarrow 323$		$11111000 \leftrightarrow 4212$
$1111010 \leftrightarrow 332$		$11111001 \leftrightarrow 423$
$1111011 \leftrightarrow 4112$		$11111010 \leftrightarrow 432$
$1111100 \leftrightarrow 413$		$11111011 \leftrightarrow 45$
$1111101 \leftrightarrow 422$		$11111100 \leftrightarrow 5112$
$1111110 \leftrightarrow 44$		$11111101 \leftrightarrow 513$
$11111110 \leftrightarrow 3213$		$11111110 \leftrightarrow 522$
$11111111 \leftrightarrow 324$		$11111111 \leftrightarrow 54$

TABLE 4.2
More variable length to variable length prefix-free codes

E/K	2	3	4	5	6	7	8
1.50	5						
1.33		6					
1.25		11	7				
1.20		23	11	8			
1.16		74	20	12	9		
1.14			40	20	13	10	
1.13			81	37	21	14	11
1.11				73	37	22	15
1.10					70	38	23

TABLE 4.3
Trade-off Between Expansion, K and Size of Encoder and Decoder

Acknowledgments

We would like to thank Matthew Cook for many suggestions and improvements.

References

[1] R. L. Adler, D. Coppersmith and M. Hasner, "Algorithms for Sliding Block Codes – an Application of Symbolic Dynamics to Information Theory", IEEE Transactions on Information Theory, vol. 29, pp. 5-22, 1983.

[2] D. Altenkamp and K. Mehlhorn, "Codes: Unequal Probabilities, Unequal Letter Costs", J. ACM, vol. 27, no. 3, pp. 412-427, July 1980.

[3] N. M. Blachman, "Minimum-Cost Encoding of Information", IRE Trans. Inform. Theory, vol. PGIT-3, pp. 139-149, 1954.

[4] M. Blaum and J. Bruck, "Coding for Skew Tolerant Parallel Asynchronous Communications", IEEE Transactions on Information Theory, vol. 39, no. 2, pp. 379-388, March 1993.

[5] M. Blaum, J. Bruck and L.H. Khachatrian, "Constructions of Skew-Tolerant and Skew-Detecting Codes", IEEE Transactions on Information Theory, vol. 39, no. 5, pp. 1752-1757, September 1993.

[6] M. Blaum and J. Bruck, "Coding for Delay-Insensitive Communication with Partial Synchronization", IEEE Transactions on Information Theory, vol. 40, no. 3, pp. 941-945, May 1994.

[7] R. M. Capocelli and P. Cull, "Generalized Fibonacci Numbers are Rounded Powers", Third International Conference on Fibonacci Numbers and Their Applications, Pisa, Italy, pp. 57-62, 1988.

[8] N. Cot, "Complexity of the variable length encoding problem", Proc. 6th Southeast Conference on Combinatorics, Graph Theory and Computing, Congressus Numerantium XIV, Utilitas Mathematica Publishing, Winnepeg, MB, Canada, pp. 211-244, 1975.

[9] N. Cot, "Characterization and Design of Optimal Prefix Codes", Ph.D. Dissertation, Stanford University, Stanford, CA., 1977.

[10] P. A. Franaszek, "Sequence-state Coding for Digital Transmission", Inform. Contr., vol. 1-J, pp. 155-164, 1969.

[11] E. N. Gilbert, "Coding with Digits of Unequal Cost", IEEE Transactions on Information Theory, vol. 41, no. 2, pp. 596-600, 1995.

[12] M. J. Golin and N. Young, "Prefix Codes: Equiprobable Words, Unequal Letter Costs", SIAM J. Comput., vol. 25, no. 6, pp. 1281-1292, 1996.

[13] M. J. Golin and G. Rote, "A Dynamic Programming Algorithm for Constructing Optimal Prefix-Free Codes with Unequal Letter Costs", IEEE Transactions on Information Theory, vol. 44, no. 5, pp. 1770-1781, 1998.

[14] D. H. Huffman, "A Method for the Construction of Minimum Redundancy Codes", Proceedings of the IRE, vol. 40, no. 9, pp. 1098-1101, 1952.

[15] S. Kapoor and E. M. Reingold, "Optimum Lopsided Binary Trees", J. ACM, vol. 36, pp. 573-590, 1989.

[16] R. M. Karp, "Minimum Redundancy Coding for the Discrete Noiseless Channel", IRE Trans. Inf. Theory", vol. 7, no. 1, pp. 27-38, 1961.

[17] W. H. Kautz, "Fibonacci Codes for Synchronization Control", IEEE Transactions on Information Theory, pp. 284-292, April 1965.

[18] D. E. Knuth, "Efficient Balanced Codes", IEEE Transactions on Information Theory, vol. 32, pp. 51-53, 1986.

[19] R. M. Krause, "Channels which transmit letters of unequal duration", Inf. Control, vol. 5, no.1, pp. 13-24, 1962.

[20] D. A. Lelewer and D. S. Hirschberg, "Data Compression", ACM Computing Surveys, vol. 19, no. 3, pp. 261-296, 1987.

[21] D. Lind and B. Marcus, "An Introduction to Symbolic Dynamics and Coding", Cambridge University Press, 1985.

[22] R. S. Marcus, "Discrete Noiseless Coding", M. S. Thesis, MIT, Electrical Engineering Dept., 1957.

[23] K. Mehlhorn, "An Efficient Algorithm for Constructing Nearly Optimal Prefix Codes", IEEE Trans. Inf. Theory, vol. 26, no. 5, pp. 513-517, 1980.

[24] Y. Perl, M. R. Garey and S. Even, "Efficient Generation of Optimal Prefix Code: Equiprobable Words using Unequal Cost Letters", J. ACM, vol. 22, no. 2, pp. 202-214, 1975.

[25] E. Sperner, "Ein Satz uber Untermengen einer endlichen Menge", Math. Z. vol. 27, pp. 544-548, 1928.

[26] W. R. Spickerman, "Binet's Formula for the Tribonacci Sequence", The Fibonacci Quarterly, no. 2, pp. 118-120, May 1982.

[27] W. R. Spickerman and R. N. Joyner, "Binet's Formula for the Recursive Sequence of Order K", The Fibonacci Quarterly, no. 4, pp. 327-331, 1984.

[28] B. Varn, "Optimal Variable Length Codes (Arbitrary Symbol Cost and Equal Codeword Probability)", Inf. Control. vol. 19, no. 4, pp. 289-301, 1971.

[29] T. Verhoeff, "Delay-insensitive codes – an overview", Distributed Computing, pp. 3:1-8, 1988.

Appendix

Lemma 2.5.2: Define $d_K(n)$ as follows,

$$d_K(n) = \begin{cases} \sum_{i=1}^{K} d_K(n-i) & n \geq 1 \\ -\beta_1^{(K)} \left[\phi_1^{(K)}\right]^n + [K-1]^{-1} & (1-K) \leq n \leq 0 \end{cases}$$

For all $n \geq 0$,

$$d_K(n) = \bar{F}_K(n) - \beta_1^{(K)} \left[\phi_1^{(K)}\right]^n + [K-1]^{-1}$$

Or equivalently, $d_K(n) = \sum_{i=2}^{K} \beta_i^{(K)} \left[\phi_i^{(K)}\right]^n$

Proof: The proof is by induction.

Base Case: $n = 0$. Follows by substitution.

Inductive Hypothesis: For $0 \leq m \leq (n-1)$ we assume,

$$d_K(m) = \bar{F}_K(m) - \beta_1^{(K)} \left[\phi_1^{(K)}\right]^m + [K-1]^{-1}$$

Now consider the case for $m = n$,

Case 1: $1 \leq n \leq K-1$.

$$d_K(n) = \sum_{i=1}^{K} d_K(n-i) = \sum_{i=1}^{n-1} d_K(i) + \sum_{i=n-K}^{0} d_K(i)$$

$$= \sum_{i=1}^{n-1} \left[2^{i-1} - \beta_1^{(K)} \left[\phi_1^{(K)}\right]^i + [K-1]^{-1}\right]$$

$$+ \sum_{i=n-K}^{0} \left[-\beta_1^{(K)} \left[\phi_1^{(K)}\right]^i + [K-1]^{-1}\right]$$

$$= 2^{n-1} - \beta_1^{(K)} \left[\phi_1^{(K)}\right]^n + [K-1]^{-1}$$

$$= \bar{F}_K(n) - \beta_1^{(K)} \left[\phi_1^{(K)}\right]^n + [K-1]^{-1}$$

Case 2: $n \geq K$.

$$d_K(n) = \sum_{i=1}^{K} d_K(n-i)$$
$$= \sum_{i=1}^{K} \left[\bar{F}_K(n-i) - \beta_1^{(K)} \left[\phi_1^{(K)}\right]^{n-i} + [K-1]^{-1} \right]$$
$$= \sum_{i=1}^{K} \bar{F}_K(n-i) - \sum_{i=1}^{K} \beta_1^{(K)} \left[\phi_1^{(K)}\right]^{n-i} + \sum_{i=1}^{K} [K-1]^{-1}$$
$$= \bar{F}_K(n) - \beta_1^{(K)} \left[\phi_1^{(K)}\right]^{n} + [K-1]^{-1} \quad \blacksquare$$

Lemma 2.5.3: For $n \geq 2$,
$d_K(n) = 2d_K(n-1) - d_K(n-K-1)$

Proof: $d_K(n) = \sum_{i=1}^{K} d_K(n-i)$
$$= \sum_{i=1}^{K} d_K(n-i) + d_K(n-K-1) - d_K(n-K-1)$$
$$= d_K(n-1) + \sum_{i=1}^{K} d_K(n-1-i) - d_K(n-K-1)$$
$$= 2d_K(n-1) - d_K(n-K-1) \quad \blacksquare$$

Lemma 2.5.4: For all $n \geq 0$, $|d_K(n)| < 1/2$

Proof: If $n \geq 2$ and if for all i, $2 \leq i \leq K$, $|d_K(n-i)| \leq 1/2$ then $|d_K(n)| < 1/2$. The proof of this statement is identical to the analogous proofs in Capocelli and Cull [7] and thus omitted. Hence, it suffices to establish that for all K, if $2-K \leq i \leq 1$, $|d_K(i)| \leq 1/2$. When $K = 2$, $d_K(0) = -0.17$ and $d_K(1) = 0.10$.

For the rest of the proof we will assume $K \geq 3$.
Case 1: $2-K \leq i \leq 0$. We need to show,

$$-1/2 - [K-1]^{-1} \leq -\beta_1^{(K)} \left[\phi_1^{(K)}\right]^{i} \leq 1/2 - [K-1]^{-1}$$

Since $-\beta_1^{(K)} < -\beta_1^{(K)} \left[\phi_1^{(K)}\right] < ..... < -\beta_1^{(K)} \left[\phi_1^{(K)}\right]^{2-K}$, it suffices to prove,

(a) $-\beta_1^{(K)} \left[\phi_1^{(K)}\right]^{2-K} \leq 1/2 - [K-1]^{-1}$

$-\beta_1^{(K)} \left[\phi_1^{(K)}\right]^{2-K} < 0$. $1/2 - [K-1]^{-1} \geq 0$ for $K \geq 3$.

(b) And $\beta_1^{(K)} \leq 1/2 + [K-1]^{-1}$. Or equivalently,

$$\left(\left[(K+1)\,\phi_1^{(K)} - 2K \right] \right)^{-1} \leq (K+1)\,/\,[2\,(K-1)]$$

Capocelli and Cull [7] establish that,
$2 - 2/2^K < \phi_1^{(K)} < 2 - 1/2^K$.

Note that,
$$\left(\left[(K+1)\,\phi_1^{(K)} - 2K \right] \right)^{-1} < \left(\left[(K+1)\,(2 - 2/2^K) - 2K \right] \right)^{-1}$$

Hence, it would suffice to show that,
$$\left(\left[(K+1)\,(2 - 2/2^K) - 2K \right] \right)^{-1} \leq (K+1)\,/\,[2\,(K-1)]$$

Simplifying the above inequality yields,
$(K+1)^2 \leq 2^{K+1}$

This is true for $K \geq 3$.

Case 2: $i = 1$. We need to show that,
$-1/2 \leq 1 - \beta_1^{(K)}\phi_1^{(K)} + [K-1]^{-1} \leq 1/2$

Substituting the value of $\beta_1^{(K)}$ and simplifying reduces this inequality to,
$2K(3K-1)/(3K^2+1) \leq \phi_1^{(K)} \leq 2K(K+1)/(3+K^2)$

(a) $\phi_1^{(K)} \leq 2K(K+1)/(3+K^2)$

$\phi_1^{(K)} < 2$. But $2K(K+1)/(3+K^2) \geq 2$ for $K \geq 3$.

(b) $2K(3K-1)/(3K^2+1) \leq \phi_1^{(K)}$

Capocelli and Cull [7] establish that,
$2 - 2/2^K < \phi_1^{(K)} < 2 - 1/2^K$.

Hence, it would suffice to show that,
$2K(3K-1)/(3K^2+1) \leq 2 - 2/2^K$

Simplifying the above inequality yields,
$(3K^2+1)/(K+1) \leq 2^K$

This inequality holds for $K \geq 3$. $\blacksquare$