NASA/CR-1998-208024

IN-61-CR
169422

A SYNCHRONIZATION ALGORITHM AND IMPLEMENTATION FOR HIGH-SPEED BLOCK CODES APPLICATIONS

Part 4

Yu Zhang, Eric B. Nakamura,
Gregory T. Uehara and Shu Lin

April 20, 1998

This paper is submitted to Globecom 98

A Synchronization Algorithm and Implementation for High-Speed Block Codes Applications

Yu Zhang, Eric B. Nakamura, Gregory T. Uehara, and Shu Lin

University of Hawaii at Manoa

Please Contact:
Gregory T. Uehara
Holmes Hall 483
2540 Dole Street
Honolulu, Hawaii 96822
(808) 956-5972
FAX: (808) 956-3427
e-mail: uehara@spectra.eng.hawaii.edu

January 28, 1998

Abstract

Block codes have trellis structures and decoders amenable to high speed CMOS VLSI implementation. For a given CMOS technology, these structures enable operating speeds higher than those achievable using convolutional codes for only modest reductions in coding gain. As a result, block codes have tremendous potential for satellite trunk and other future high-speed communication applications. This paper describes a new approach for implementation of the synchronization function for block codes. The approach utilizes the output of the Viterbi decoder and therefore employs the strength of the decoder. Its operation requires no knowledge of the signal-to-noise ratio of the received signal, has a simple implementation, adds no overhead to the transmitted data, and has been shown to be effective in simulation for received SNR greater than 2 dB.

I. Introduction

As speed requirements in communication systems increase, methods to implement high performance error correction codes for reliable bandwidth efficient data transmission at high speeds are needed. Soft-decision decoded block codes have trellis structures amenable to IC implementation [3,5] and can achieve performance levels which approach those of convolutional codes.

For example, we have proposed a (64, 40, 8) subcode of the third-order Reed-Muller (RM) code to NASA for high-speed satellite communications [6]. This RM subcode can be used either alone or as an inner code of a concatenated coding system with the NASA standard (255, 233, 33) Reed-Solomon (RS) outer code. This yields a high performance (or low bit-error rate) system with reduced decoding complexity. It can also be used as a component code in a multilevel bandwidth efficient coded modulation system to achieve reliable bandwidth efficient data transmission. The proposed (64, 40, 8) RM subcode has a relatively simple, parallel trellis structure with a high degree of regularity. Consequently, a group of structurally identical and relatively simple Viterbi decoders can be designed to process the decoding in parallel. This not only reduces the decoding complexity but also increases the achievable decoding speed. For the AWGN channel using BPSK transmission, the (64, 40, 8) RM subcode with a code rate of 0.626 bits/symbol achieves only 0.5 dB less coding gain than the NASA standard rate-1/2 64-state convolutional code. We believe the aforementioned structural advantages result in a factor of eight (8) increase in attainable throughput. A prototype custom integrated circuit to implement the (64, 40, 8) RM subcode decoder is currently being designed[6].

An important function required in block code implementations and not present in convolutional code implementations is that of *block synchronization.* In a block code with N symbols per block, the received signal must first be separated into N symbol blocks prior to decoding as shown in Fig. 1. A method is needed to identify and appropriately section the blocks. While there is much literature describing the theory of block codes [3,4], little has been published in the area of block synchronization methods. Most previously published work which can be used on block synchronization describe methods which utilize synchronization bytes or sync fields [1,2].

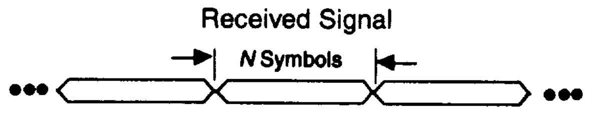

Figure 1 The received data signal is made up of blocks of *N* symbols.

Methods which use goodness measures such as the winning path metric are possible. Consider a decoder which chooses the winning path metric that is the largest. For a given SNR and over all *N* possible symbol phases, the winning path metric in the in-phase condition will be the largest. Methods can be devised in which the winning path metric in a particular symbol phase is compared with a decision threshold to determine whether or not synchronization has occurred. Unfortunately, this threshold will typically be a function of the SNR. Accurate estimation of the SNR is difficult, particularly in low SNR situations. Thus, such approaches are best suited to applications where the SNR is large.

In this paper, we describe a method for synchronization of block codes for binary signaling applications that does not need information about the SNR. It uses a single SNR *independent* threshold. The approach has a relatively simple implementation with the complex synchronization acquisition implemented at low speed and only simple delay/latching functions performed at the full symbol rate. Furthermore, the approach requires no additional information, such as a synchronization byte, to be added to the coded data. It utilizes the output of the Viterbi decoder and as such, employs the strength of the decoder. The approach has been shown in simulation to be effective for off- channel SNRs greater than 2 dB. This approach will be implemented in the prototype (64, 40, 8) RM decoder we are building.

The outline of the paper is as follows. Following this introduction, Section II presents possible approaches for block synchronization. In Section III, we describe the proposed method for the (64, 40, 8) RM subcode and present simulation results in Section IV for the approaches described in Sections II and III. Finally, we conclude with a summary in Section V.

II. Synchronization Approaches

There are two primary classes of approaches for block code synchronization which we call *extrinsic* and *intrinsic* synchronization methods. In *extrinsic* synchronization, additional bits are added in the form of a synchronization byte for the sole purpose of synchronization. In *intrinsic* synchronization, no extra bits are added. Instead information derived from the normal decode process is used for acquisition and tracking of synchronization. In this section, we will examine these approaches in further detail.

A. Extrinsic Synchronization: Extra Bits Added for Synchronization

One approach for block synchronization is to employ a synchronization or sync byte in which a well-defined pattern is transmitted every P blocks as shown in Fig. 2 for P equal to 3 (P is typically much greater than 3). The receiver simply identifies this byte and synchronizes to it. For large enough P, there is a relatively small overhead associated with this byte. However, the need to remove this sync byte introduces irregularity in the decoder structure and increases its complexity. More important is that the sync byte is difficult to find and track in low SNR situations and thus this approach is best suited for high SNR applications with low to moderate data rates.

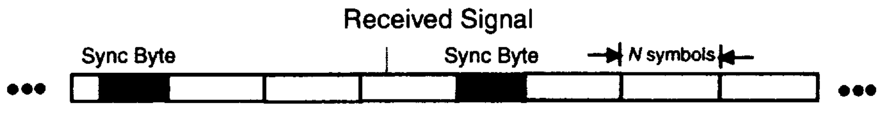

Figure 2 Intrinsic synchronization utilizing a sync byte.

B. Intrinsic Synchronization: No Extra Bits Added for Synchronization

A second class of synchronization approaches is shown conceptually in block diagram form in Fig.3. Viterbi Decoder outputs such as the winning path metric or winning codeword are used by the *Synchronization Detector* (SD) block to infer the presence or absence of synchronization. During acquisition, the Phase Shifter arbitrarily chooses a phase and the decoder operates on a received block. The appropriate Viterbi decoder outputs, which depend upon the particular implementation, are then passed to the Synchronization Detector. If synchronization is detected, that particular phase is maintained. If not, the Phase Shifter selects a new phase and the decoder then operates on a new block of data. This entire sequence is repeated until the Synchronization Detector determines that synchronization has occurred. Once the system is synchronized, tracking can be accomplished by monitoring the output of the SD. In a practical implementation, the Synchronization Detector may average or sum Viterbi decoder outputs over P blocks for increased robustness.

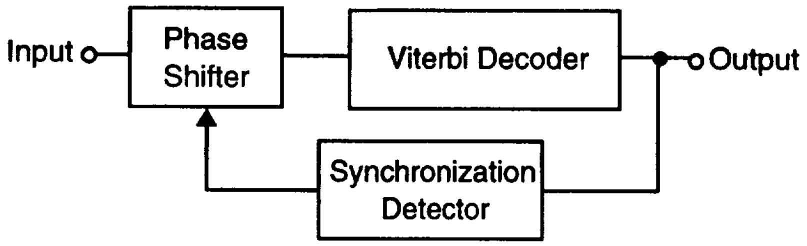

Figure 3 Block diagram of a decoder employing feedback for synchronization.

There are many possible implementations for the Synchronization Detector whose robust performance is essential to achieving the full benefits of the block code. Along with the received symbols, the winning path metric is a readily available output from the Viterbi decoder. With simple modifications, the decoder outputs can also include the winning (or most likely received) codeword. Both the winning path metric and the winning codeword can be used as inputs to the Synchronization Detector. We will now examine three possible approaches for implementation of the Synchronization Detector and the associated block diagram for each resulting decoder system.

Simple Threshold-PM (Path Metric) Based Synchronization Detector

This approach utilizes the winning path metric as the measure to infer synchronization. In a typical block code decoder which computes branch correlation metrics, the decoder chooses the path with the largest path metric calculated over all allowed paths. Whether or not synchronization has occurred, each data block input to the decoder will result in a path metric that is largest among each of the computed path metrics. At any given input SNR, the winning path metric in the synchronized condition will be larger than any of the winning path metrics in any unsynchronized condition. This information can be used to infer synchronization. If the winning path metric exceeds some threshold, synchronization is assumed to have occurred.

A conceptual plot of the *average* winning path metrics out of the decoder for both the in-phase and out-of-phase cases as a function of the received SNR is shown in Fig. 4a for a typical block code. Note from the figure that the threshold for the Synchronization Detector is well defined, albeit a function of the SNR. The figure also reflects the difference between the in-phase and out-of-phase metrics is relatively small with practical codes. Therefore, the SNR must be inferred accurately so that an appropriate threshold can be chosen.

One way to infer the SNR is to calculate an average bit-error-rate in the decoder. This can be accomplished by having the Viterbi decoder determine the most likely transmitted codeword and comparing this with hard decisions made on the received block to create an estimate of the raw bit-error-rate or BER. In a practical implementation, the number of errors can be summed over a fixed number of blocks to measure an average BER. This estimate of the BER can be used to generate an estimate of the SNR, which in turn can be used to choose the synchronization detector threshold. In practice, this process is noisy at best. In low SNR environments, this can lead to the choice of the wrong threshold and incorrect locking. As opposed to the continuous threshold shown in Fig. 4a, a number of discrete thresholds can be used as shown in Fig. 4b. The appropriate threshold in such a case could be chosen in a similar manner described above but determined using ranges of SNR.

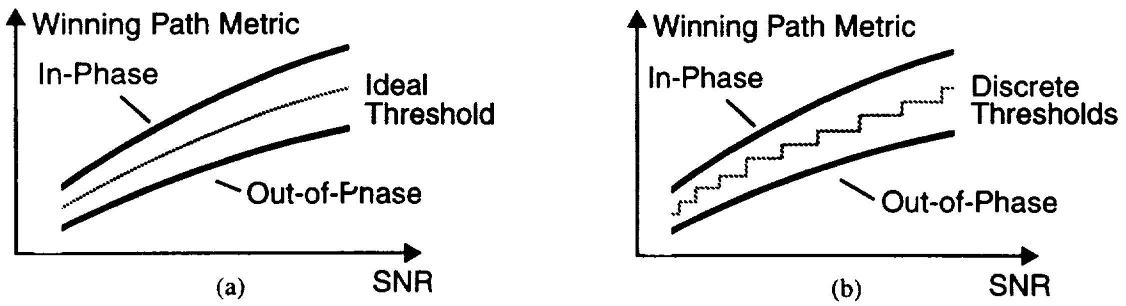

Figure 4 a. Conceptual illustration of a plot of the average *winning path metrics* as a function of the received SNR for the in-phase and out-of-phase conditions. b. Discrete-thresholds versus the ideal threshold in (a).

A block diagram of a decoder employing the Simple Threshold-PM synchronization detector is shown in Fig.5. A symbol phase is chosen arbitrarily and the winning path metric is summed for P blocks. This sum is compared with a threshold that is a function of the received SNR and derived in a manner described earlier. When the Path Metric Sum (indicated in the figure) exceeds the threshold, the current phase is maintained until a time when the path metric sum is smaller than the threshold and synchronization is lost. The acquisition process is then repeated[1] to resynchronize the system. The performance limitation of this approach is that the threshold is a function of the inferred SNR which is unreliable at low SNRs. Thus, this approach is best suited for applications in which the SNR is high.

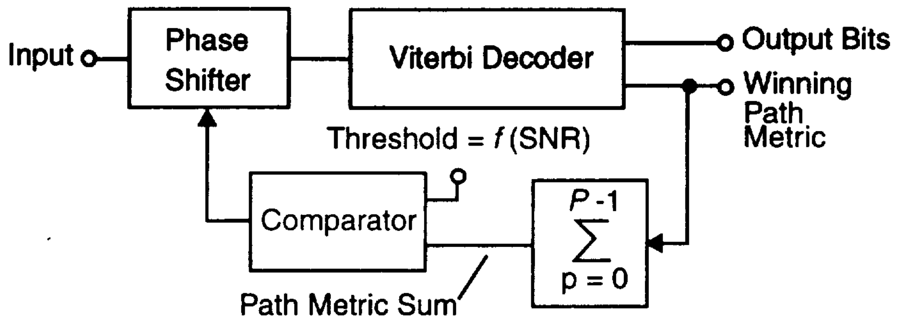

Figure 5 Block diagram of the use of the Simple Threshold-PM synchronization detector.

Simple Threshold-BER (Bit-Error Rate) Based Synchronization Detector

This approach is similar to the Simple Threshold-PM with the difference that the number of bit errors per P blocks is used instead of the summed winning path metrics. A method similar to that used to infer the BER (for the SNR estimation in the Simple Threshold-PM) is used generate the number of errors between the winning block and hard decisions made on the received symbols. At a given SNR, when the received block is in-phase with the decoder, the number of bit errors per this calculation method is smaller than when it is out-of-phase. Thus, a set of curves which have the similar dependence on the SNR as those in Fig. 4 can be generated and the appropriate threshold, which again is a function of the SNR, can be determined.

1. Many adhoc methods for tracking and re-synchronization have been considered and can easily be implemented. The focus of this discussion is the implementation of the Synchronization Detector.

The winning path metric and the BER are related through the SNR and it turns out that both this method and the Simple Threshold-PM are similar. However, in this method, since hard decisions are made on the received input block, information about the received sequence is thrown out and as a result, the Simple Threshold-PM will perform better than the Simple Threshold-BER.

Differential Decoder Reference Synchronization Detector

The large number of discrete thresholds required using the approach illustrated in Fig. 4b comes from the dependence of the winning path metric on the SNR for both the in-phase and out-of-phase conditions. One solution to this is to create a metric that is a weaker function of the SNR which can reduce the required number of thresholds. Observing that both the in-phase and out-of-phase plots of the winning path metrics, shown in Fig. 4a, are a function of SNR and have positive slopes, we examined the differential scheme shown conceptually in Fig. 6. Two decoders are used. The first is the main (data) decoder and the second operates with a symbol phase delay which is *not* a multiple of the N (number of symbols in a block). In this way, the second decoder is forced to operate *out-of-phase* from the first.

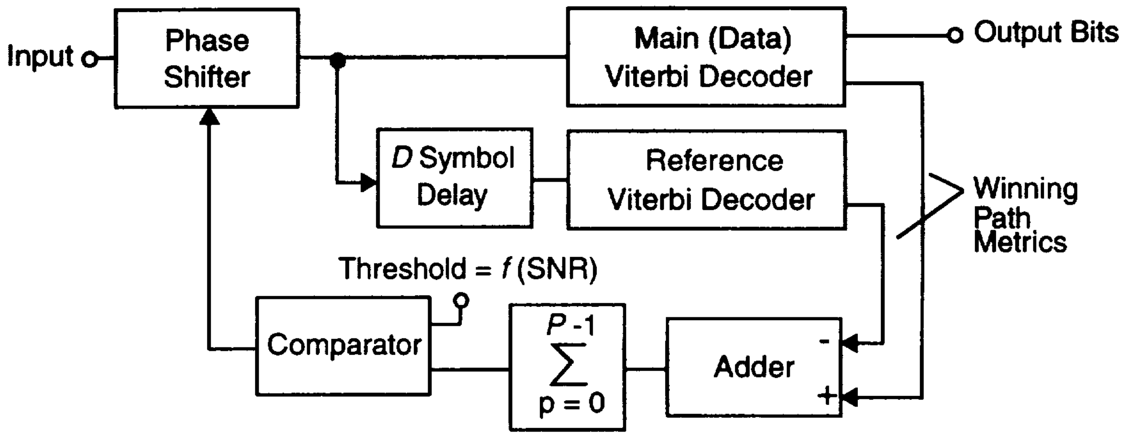

Figure 6 Block diagram of the use of the Differential Decoder Reference synchronization detector.

The *difference* between the winning path metrics from the main and reference decoders is generated and used to determine synchronization. Intuitively, the difference between the two curves of Fig. 4a results in the reduction of the common-mode dependence on the SNR. The result shown in Fig. 7b is a synchronization measure that is a weaker function of the SNR than both curves in Fig. 4a. This achieves

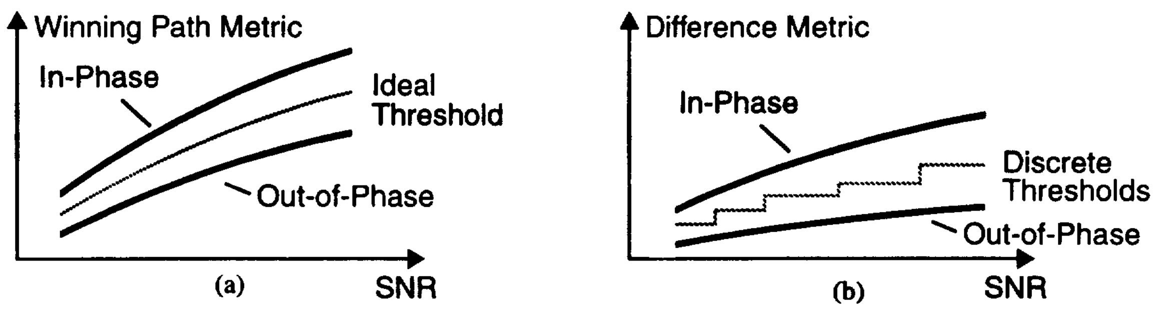

Figure 7 a. Fig. 4a repeated for convenience. b. Conceptual illustration of a plot of the *average* difference metrics as a function of the received SNR for the in-phase and out-of-phase conditions.

the desired result of the need for fewer discrete thresholds. However, more than one threshold is still needed, requiring the undesirable need to once again infer the SNR. Furthermore, this solution is quite expensive hardware-wise as it requires a second decoder used exclusively for synchronization.

III. Proposed Approach: Differential Correlation Reference (DCR) Synchronization Detector

In this section, we describe what we call the Differential Correlation Reference (DCR) synchronization detector. This approach exploits the following observation: the Viterbi decoder in effect calculates the correlation between the BPSK received signal and all possible transmitted binary codeword sequences and chooses the sequence with the largest correlation.

Let us define the *correlation reference* as the correlation between the received block and the binary sequence among all 2^N sequences (where N is the number of symbols in a block) for which the correlation is largest. As it turns out, the implementation of the correlation reference generator is quite simple. For any given received sequence, the correlation reference is generated simply by taking the absolute value of each input and summing over all N received symbols. The correlation reference for the curves of Fig. 4a has been added in Fig. 8a. Note in the figure that starting from the low SNR end of the plot and moving right, how the in-phase plot converges toward and the out-of-phase plot diverges away from the correlation reference. This makes the correlation reference the ideal reference for the differential structure described

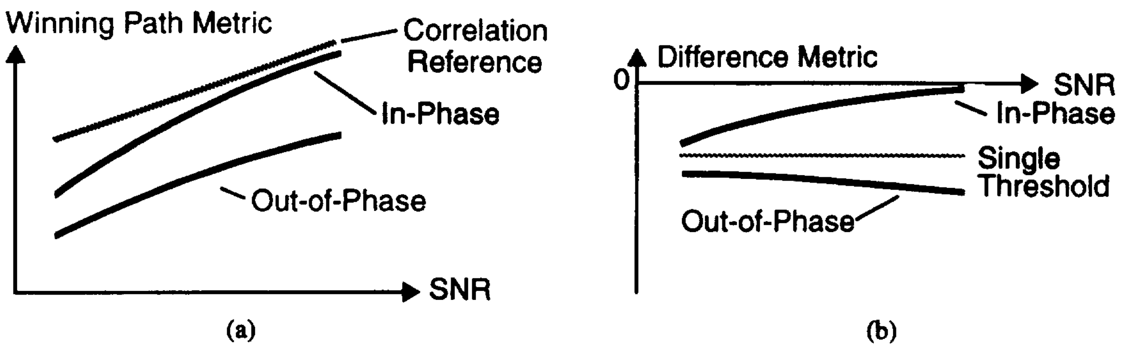

Figure 8 a. Conceptual illustration of a plot of the *average* winning path metrics as a function of the received SNR for the in-phase and out-of-phase conditions together with the correlation reference. b. In-phase and out-of-phase difference metric as a function of SNR. Note: A *single* threshold is possible.

earlier. Subtracting both in-phase and out-of-phase curves from the correlation reference results in the two curves shown in Fig. 8b. This allows a single threshold to be used to determine synchronization.

The differential correlation reference synchronization detector is shown conceptually in Fig. 9. Instead of a second decoder, as used in the Differential Decoder Reference approach, this requires the addition of only the absolute value function and an adder. Note from Fig. 8b that the difference between the in-phase and out-of -phase cases can be quite small. This difference is easily increased by increasing *P*, the number of blocks over which the synchronization control signal is generated.

The key disadvantage of the proposed synchronization detector is the time it can take to acquire synchronization. Since there are *N* possible symbol phases and *P* blocks are averaged per phase, in the

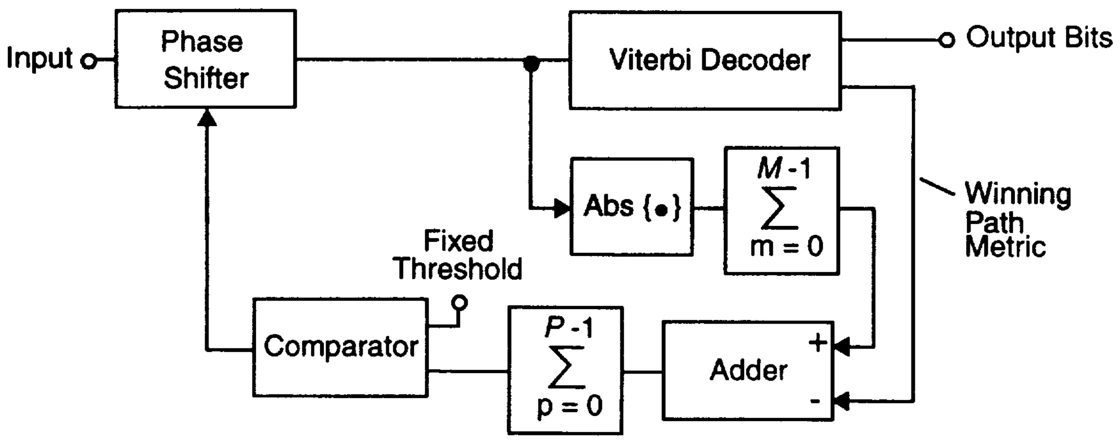

Figure 9 Block diagram of the Differential Correlation Reference synchronization detector.

worst case, acquisition will take *NP* blocks. The larger the *P*, the more robust the resulting detector as more averaging is performed.

The DCR synchronization detector has three key advantages over the other methods considered. First, its hardware cost is relatively small. The correlation reference is quite simple to compute by making a few modifications to the Branch Metric Unit which generates branch metrics for the Add-Compare-Select (ACS) Units of the decoder. Second, it allows the choice of a single threshold to detect the in-phase condition for all SNRs. As a result, SNR information is not needed at all. Third, it falls in the class of synchronization methods we call intrinsic synchronization and therefore, no extra information needs to be added to the transmitted block.

IV. Simulation Results

In this section, we present simulation results obtained with the (64,40,8) RM subcode for the intrinsic synchronization methods described in Sections II and III. The synchronization detector output for three of the methods are plotted for each of the 64 phases as a function of SNR. In each case, the in-phase condition results in the largest output. A line indicating possible threshold levels is shown for each method. These simulations each used *P* equal to 50. These curves are a function of the random channel noise and data patterns and while they are typical, are only the output from a single run.

The results for the summed path metric output is shown in Fig. 10a. The ideal threshold is clearly a strong function of the SNR. As such, any method employing this synchronization function would need accurate estimation of the SNR. A typical output of the Differential Decoder Reference synchronization detector is shown Fig. 10b. For this simulation, a (64, 35, 8) RM subcode decoder was used for the reference decoder. As can be seen from this diagram, it is possible to use only four thresholds. The appropriate threshold would be chosen based on the SNR. This relaxes the accuracy with which the SNR needs to be known. Yet, it nonetheless requires SNR information.

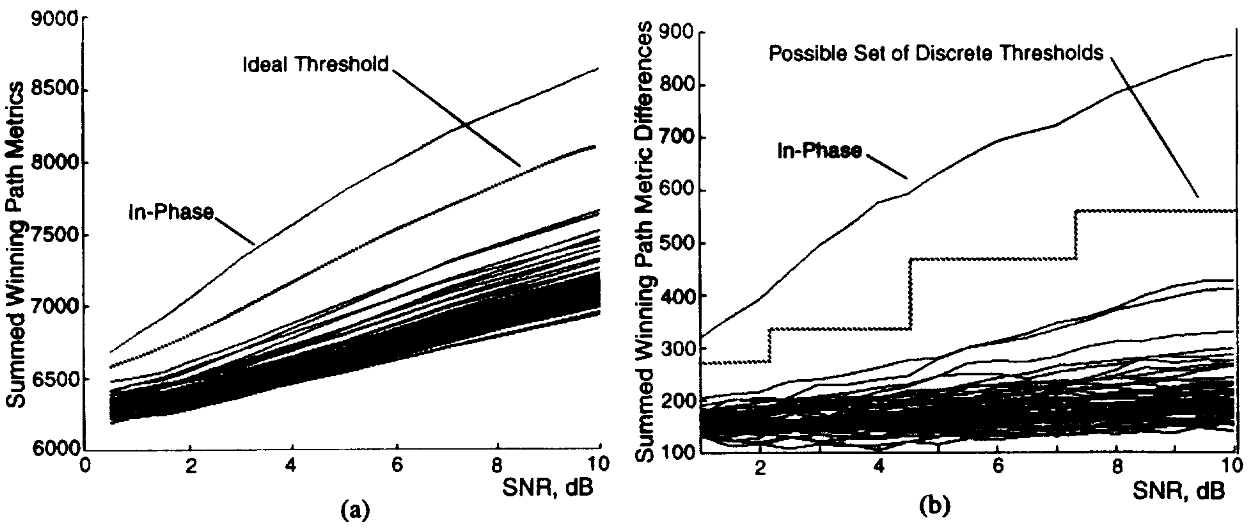

Figure 10 **Simulations results for *P* equal 50 for: a. Summed path metric output as a function of SNR for the in-phase and out-of-phase conditions; and b. Differential Decoder Reference synchronization detector output.**

Simulation results for the Differential Correlation Reference are shown in Fig. 11. The threshold optimized for an SNR of 2 dB is shown in the figure. The diverging nature of the in-phase and each of the out-of-phase curves allows the use of a single threshold and eliminates the need for information about the SNR.

Of practical interest is the probability with which this synchronization detector will make an error. The probability of synchronization error is the sum of the probability that the detector output when in-

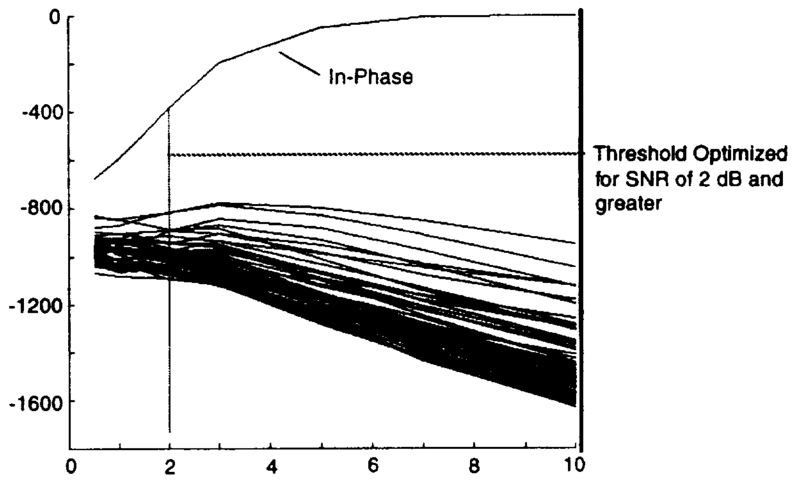

Figure 11 Simulations results for *P* equal 50 for the Differential Correlation Reference synchronization detector.

phase drops below the threshold and the detector output when out-of-phase increases above the threshold. The separation of the two curves in Fig. 11 is directly proportional to the averaging factor *P*. For larger values of *P*, the separation is greater and the probability of error is decreased as the distance between the curves and the threshold is increased. However, the system performance implication of a larger *P* is a longer average time for acquisition.

We envision an SNR of 2 dB to be below the minimum SNR generally used in practice. The distance of the topmost out-of-phase output to the threshold stays approximately constant up to an SNR of about 5 dB at which point the distance increases. We chose the SNR equal 2 dB case as a worst case condition under which to verify the robustness of the proposed approach with *P* equal to 50. Under this condition, ten-thousand(10,000) simulations were run with a decision threshold equal to -580. Of the 10,000 simulations performed, there were *no* synchronization detector errors. Since the simulated channel noise is modeled as additive white Gaussian noise, the winning path metrics which are linear combinations of samples of the channel output have a Gaussian distribution. From the simulation results, we calculated the means and corvariances for the in-phase and out-of-phase conditions in order to estimate the false lock probability. This error probability was calculated by summing the probability that the in-phase result is smaller than the threshold and the out-of -phase result is greater than the threshold. Using this approach, We estimate the false lock probability to be below 10^{-10}.

As the averaging factor is increased, rather large performance gains can be obtained since the distance between the in-phase and out-of-phase conditions increases linearly with *P*. Since this affects acquisition time in only a linear manner, for most applications, a larger *P* (such as 50) can be worth the potentially longer acquisition time. Gear shifting methods are also possible in which different values of *P* are used during acquisition and tracking.

V. Summary

In this paper, we surveyed potential approaches for block code synchronization and presented an intrinsic method that does not require information about the received SNR. This method: uses a single threshold *independent* of the SNR (dependent instead upon *P*, the number of blocks averaged per symbol phase); has a relatively simple implementation; and employs the strength of the decoder. Simulation results were presented for the prototype (64, 40, 8) RM subcode decoder currently being designed. This approach can easily be modified to be used in decoders which utilize minimum Euclidean distance metrics. The proposed algorithm and implementation make practicable the use of block codes for satellite trunk and other very high speed communication applications.

VI. Acknowledgments

This work was supported by NASA Grant NAG5-2938.

The authors are grateful to Mr. Warner Miller of NASA Goddard Space Center for his support and encouragement of this work along with Prof. Marc Fossorier for technical discussions.

REFERENCES

[1] R.A.Scholtz, "Frame Synchronization techniques", *IEEE Trans. Communications*, Vol. COM-28, pp 1204-1212, Aug.1980.

[2] T.Schaub and A.Hansson, "Frame Synchronization For Spontaneous Transmissions",Conf. Rec. GLOBECOM'90, San Diego, CA, Paper 407.6, Dec.1990.

[3] M. P. C. Fossorier and S. Lin, "Coset codes viewed as terminated convolutional codes", *IEEE Trans.Communications*, vol.44, No.9, Sept. 1996.

[4] S.Lin and D.J.Constello Jr., *Error Control Coding Fundamentals and Applications*, Englewood Cliffs, NJ: Prentice-Hall, 1983.

[5] H. Moorthy, S. Lin, and G. T. Uehara, "Good trellises for IC implementation of Viterbi decoders for linear block codes", *IEEE Trans. Communications*, vol. 45, no. 1, pp. 52 - 62, Jan. 1997.

[6] S. Lin, G. T. Uehara, E. Nakamura, and W. P. Chu, "Circuit design approaches for implementation of a subtrellis IC for Reed-Muller subcode", *Nasa Tech. Rep. 96-001,* Feb. 1996.