Reduced The Complexity of *Soft Input Soft Output Maximum A posteriori* Decoder of Linear Block Code By Using Parallel Trellises Structure

Samir Abdul Cathem Khother **Haider Jabbar Abd**
Electrical Engineering Department , Collage of Engineering, Babylon University
haiderlaser@yahoo.com

Abstract

This search present trellis structures of linear block code capable of achieving high decoding speed while satisfying a constraint on the structural complexity of the trellis in terms of the maximum number of states at any particular depth. First we describe minimal trellis of linear block code that minimizes one or more measures of trellis complexity for the code. We identify the primitive structures that can appear in a minimal trellis , then we applied the sectionalized to trellis, where only uniform sectionalizations of the code trellis diagram are considered .Next, parallel and structurally identical subtrellises for a code without exceeding the maximum state complexity of the minimal trellis of the code is discussed. The complexity of decoder based on a sectionalized trellis diagram for a code is described.

In This paper we describe how to apply SISO(soft input soft output) max-log-MAP decoder using parallel structure of sectionalized trellis for block code. The (8,4) and (16,5)RM (Reed Muller) code are included here because they offers parallel and structurally identical subtrellises without cross connections among them that will reduces the decoding complexity and improves decoding speed. This paper also gives simulation results for iterative decoding of parallel concatenated block code of these two codes over AWGN channel by using SISO max-log-MAP decoder based on parallel trellises structure.

Keywords: SISO,SNR,BER , MAP and RM code

الخلاصة

هذه البحث يعرض هياكل شجيرية للشفرات الخطية المقطعية القادرة على تحقيق سرعة تشفير عالية بينما تحافظ على هيكل التعقيد ثابت بدلالة اعظم عدد للحالات او لكل عمق عملي . بالبداية نحن أوضحنا الشجرة الاقتصادية للشفرات المقطعية الخطية التي تقل واحد او اكثر من مقايس التعقيد لشجرة الشفرة ،ثم تعريف هياكل متكررة التي تظهر في الشجرة الاقتصادية ،ثم قمنا بتطبيق عملية التقطيع على الشجرة ،حيث فقط التقطيع المتساوي لمخطط الشجرة للشفرة قد تم أخذه بالاعتبار .تعقيد المشفر بالاعتماد على المخطط الشجري لشفرة قد تم توضيحه . من هذه الورقة نحن أوضحنا كيفية تطبيق المشفر من نوع – SISO-Log MAP باستخدام الهياكل الشجيرية المتوازية المقطعة لشجرة الشفرة المقطعية ،الشفرات (8,4)RM و (16,5)RM قد تم تضمينها هنا لأنها تقدم هياكل شجرية متماثلة بدون تقاطع بينها ذلك سوف يقلل التعقيد للمشفر ويحسن سرعة المشفر .هذه الورقة تعطي نتائج محاكاة للمشفر التكراري للشفرات التلاحقية المتوازية للشفرات المقطعية لهاتين الشفرتين خلال قناة كاوسية الضوضاء باستعمال المشفرSISO-max-log بالاعتماد على الهياكل الشجيرية المتوازية .

الكلمات المفتاحية:-مشفر MPA ،هياكل الشجيرة المتوازية ، ترميز MAP،نسبة الاشارة الى الضوضاء ، المرمز الخطي .

I Introduction

Modern powerful decoding scheme, like turbo codes, necessitate the utilization of iterative decoding algorithms. (Bendetto *et al.*, 2008; Divsalar and Pollara, 2007) describe a general SISO MAP module that can be used in iterative decoding of serial concatenation convolutional codes (SCCC) and parallel concatenation convolutional codes (PCCC). For practical application, this decoding algorithm must be simplified and its decoding complexity and delay must be reduced. Usually max-log-MAP used to reduce the decoding complexity as compare with MAP decoder with negligible degradation in BER performance (Ye Lin *et.al.*, 2006).

Any linear block code can theoretically be decoded by applying the max-log-MAP algorithm to a trellis for the code. Decoding complexity remains, however, a serious impediment to the use of trellis decoders for block codes. In this paper we focus on optimal sectionalization of a trellis to minimize computational complexity that minimizing the total number of decoding operations.

II Minimal Trellis of Binary Linear Block Codes and its Primitive Structures.

A trellis is a powerful method of describing a block code. For the construction of a trellis, the encoder starts from some initial state, usually all-zero state, denoted by s_0. At each time instant, $i \in \{0,1,\ldots,N\}$, the encoder takes on exactly one state s_i. There are a finite number of allowed states at time instant i denoted as $\sum_i(C)$.As the encoder process moves from $i-1$to i a code bit c_i is emitted (Peter, 2009) .

A-Minimal Trellis of Block Codes.

The generator matrix of a linear block code *C* is said to be in a trellis oriented form, if the following conditions were hold (Peter, 2009; Udayan, 2006):

1- The leading '1' of any row appears in a column before the leading '1' of any rows appears before it.
2- No two rows have their trailing '1's in the same columns.

Any generator matrix can be put in a trellis oriented generator matrix (TOGM) by using Gaussian elimination on row, or made any row permutation (Peter, 2009; Udayan, 2006).

Let $r_1,r_2,\ldots,r_k$ be the k rows of TOGM where $r_\ell = [\, r_{\ell 1}, r_{\ell 2}, \ldots, r_{\ell N}\,]$, $1 \le \ell \le k$, then the span of a row r_ℓ in TOGM is defined as the smallest interval {i,i+1,…,j}, which contains all the non zero elements of a row r_ℓ. This is denoted by span(r_ℓ) . For a row r_ℓ in TOGM whose span is [i,j], the active span of a row r_ℓ denoted by aspan(r_ℓ) is defined as aspan(r_ℓ)=[i,j-1] , for i<j. The aspan(r_ℓ)=ϕ for i=j(Peter, 2009; Udayan, 2006).

Note that TOGM of a given generator matrix is not unique, then the TOGM of a given generator matrix is called minimal span generator matrix (MSGM) if all the spans are as short as possible (Peter, 2009; Udayan, 2006). A minimal trellis for the code can be constructed from the MSGM. The trellis has *n* + 1 levels of vertices and *n* levels of edges. The vertex levels, called depths, are numbered from 0 to *n*; the edge levels, called stages, are numbered from 1 to *n* (Peter, 2009; Udayan, 2006).

The vertex span (active span) of the *j*th row is the set of depths at which the *j*th information symbol can affect the encoder state (Kiely *et al.*, 2005).

B- Past and Future Subcodes

The *i*th past and future subcodes, denoted P*i* and F*i*, can be define as:the sets of all codewords whose vertex (active) spans are contained in [0*; i*-1] and [*i*+1*; n*], respectively. The dimensions of these codes can be easily determined from the MSGM: $f_i = Dim(F_i)$ is the number of rows for which the leftmost nonzero entry lies in column *i*+1 or later, and $p_i = Dim(P_i)$ is the number of rows for which the rightmost nonzero entry lies in column *i* or earlier. This implies that *pi* and *fi* are monotonic(Kiely *et al.*, 2005).

$$0 = p_0 \le p_i \le \ldots \le p_N = K = f_0 \ge f_1 \ge \ldots \ge f_N = 0 \qquad (1)$$

and never change by more than 1 from one index to the next.

For each $1 \le i \le N$ the left- and right-basis indicators are defined as, $l_i, p_i \in \{0,1\}$ to identify the positions where the future and past dimensions change(Kiely *et al.*, 2005):

$$l_i = f_{i-1} - f_i \quad r_i = p_i - p_{i-1} \qquad (2)$$

For any *i*, *li* = 1 if and only if the edge span of some row of the MSGM *begins* in column *i*. Similarly, *ri* = 1 if and only if the edge span of some row of MSGM *ends*

in column *i*. The columns where *li* = 1 and the columns where *ri* = 1 each forms a basis for the column space of MSGM, and these sets are called the left basis and the right basis, respectively. The positions of the left and right basis columns can be regarded as information positions when the generator matrix is used to encode the information left to right or right to left, respectively (Kiely *et.al.,* 2005; Luna *et.al.,* 2008).

C- Primitive Structures of a Minimal Trellis

There are four basic building blocks that can be used to construct the minimal trellis for any block code. At any given stage *i*, all primitive structures are of the same type, which is determined by the values of *li* and *ri*. The primitive structures are(Kiely *et.al.,* 2005):

1-Simple extension (−): This primitive structure appears at stage *i* when *li*=0*; ri*=0 .Simple extensions at stage *i* imply a single edge out of each vertex at depth *i*-1 and a single edge into each vertex at depth *i*; hence, the number of vertices remains constant.

2-Simple expansion (<): This corresponds to li = 1; ri = 0. There are 2 edges out of each vertex at depth i-1, and a single edge into each vertex at depth i, hence, multiplying by 2 the number of states from one vertex depth to the next.

3-Simple merger (>): This corresponds to li = 0; ri = 1. A simple merger is a time-reversed simple expansion, reducing the number of states by a factor of 2.

4-Butterfly (×): This corresponds to li = 1; ri = 1. There are 2 edges out of each vertex at depth i -1 and 2edges into each vertex at depth i; hence, the number of states is constant.

III Trellis Complexity of Binary Block Codes

The complexity of a trellis decoding for a block code *C* follows almost immediately from the complexity of a trellis representation on which it is based. Total number of vertices, V(*C*), and total number of edges, E(*C*), of N-sections trellis of binary linear block code *C* are defined as (Kiely *et.al.,* 2005; Luna *et.al.,* 2008):

$$\mathrm{E}(C)=\sum_{i=1}^{N}2^{e_i} \tag{3}$$

$$\mathrm{V}(C)=\sum_{i=0}^{N}2^{v_i} \tag{4}$$

Where 2^{e_i} is the total number of edges in section *i*, e_i is the edge space dimension of section *i*. For minimal trellis the vertex space dimension at depth *i* is(Kiely *et.al.,* 2005):

$$v_i = K - f_i - p_i \tag{5}$$

the edge space dimension at stage *i* is(Kiely *et.al.,* 2005):

$$e_i = K - f_i - p_{i-1} \tag{6}$$

and the total number of mergers for trellis of binary block code *M(C)* is (Kiely *et.al.,* 2005):

$$M(C) = N_{>} + 2N_{\times} = E(C) - V(C) + 1 \tag{7}$$

IV Trellis Sectionalization and Parallel Trellises Structure.

Equation (11) shows that the decoding complexity depend mainly on the number of edge and number of vertexes, so that we must reduce E(*C*), and V(*C*). The first method that used to reduce the trellis complexity is using minimal trellis which discussed in section II, the second method is using trellis sectionalization methodology.

A- Sectionalized Trellis

In the section II, a trellis is constructed where each state transition is associated with one code bit. Some times it is favorable to group more than one bit together for a state transition, thus the boundary locations Q={q_0=0, q_1, q_2, ……,q_{V-1}, q_V=N} in the trellis, where $q_{i-1}<q_i$, $1 \le i \le V$, and V denotes the total number of sections. A section of the trellis consists states $\sum_{q_{i-1}}(C)$ and $\sum_{q_i}(C)$ and all possible transitions between any state $s_{q_{i-1}}$ and any state s_{q_i} . There are $n_i = q_i - q_{i-1}$ code bits associated with section *i*, forming the sequence $[c_{q_{i-1}+1},.....,c_{q_i}]$. A trellis is called uniformly sectionalized if $n_i = q_i - q_{i-1} = N/V$ for all *i*(Peter and Daniel, 2007; Peter 2009) .

B- Parallel Trellises structure.

The third method to reduce the decoding complexity is using trellis sectionalization with parallel trellises structure. A V sections trellis for a linear block code can be decomposed into parallel and structurally identical subtrellises without cross connections among them and without exceeding the maximum state complexity of the trellis. Each subtrellis has much smaller state complexity and connectivity than the full code trellis. This parallel decomposition allows devising identical smaller decoders to process the subtrellises in parallel independently without communication between them. This also simplifies IC implementation and speeds up the decoding process (Moorthy *et.al.,* 2006**)**.

V SISO max-log-MAP Decoder for Binary linear Block Codes.

In (Bendetto *et.al.,* 2008; Divsalar and Pollara, 2007), Benedetto et.al., described a technique for decoding convolutional codes based on continuously updating the maximum a posterior probability of input and output code bits using soft input soft output MAP algorithm. This algorithm is a symbol-by-symbol trellis based decoding algorithm, therefore we will modified this algorithm to make it work on the trellis of linear block code. For practical application, max-log-MAP is preferred to MAP algorithm since max-log-MAP uses log-likelihood ratio of each bit. Log-Likelihood Ratio (LLR) of a binary bit $z \in$ {-1,+1} is defined as:

$$\lambda(z) = \ln\left(\frac{p(z=1)}{p(z=-1)}\right) \tag{8}$$

The SISO max-log-MAP module, shown in Fig.(1), is a three port device that accepts as inputs the LLR's of the information bits $\lambda(U,I)$ and code bits $\lambda(C,I)$, and outputs as updates LLR's for the information bits $\lambda(U,O)$.

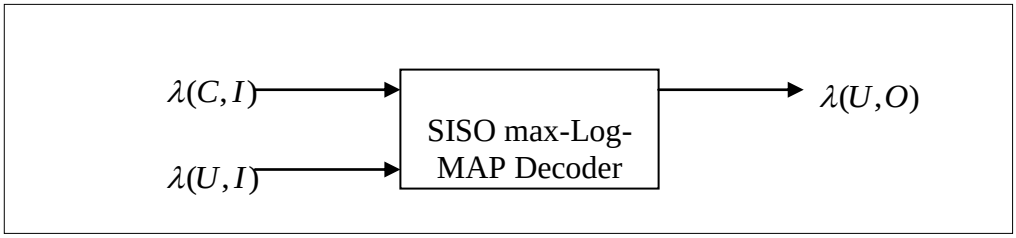

Fig.(1) SISO max-Log-MAP Decoder Module

A- SISO max-Log-MAP Decoder Based Sectionalized Trellis.

The dynamics of a time variant *V* section trellis of (N,K)binary linear block code with state space dimension$\{v_0, v_1, \ldots, v_t, \ldots v_V\}$ and edge space dimension$\{e_1, \ldots, e_t, \ldots e_V\}$ is completely specified by a single trellis section, which describes the transition (edge) between states of the trellis at time instants t-1 and t. A trellis section at time t is characterized by the following:

1- A set of $2^{v_{t-1}}$ starting state.

2- A set of 2^{e_t} edges which represent all possible transition between the trellis states.

3- A set of 2^{v_t} ending state.

The following functions are associated with each edge e, as shown in Fig.(2)

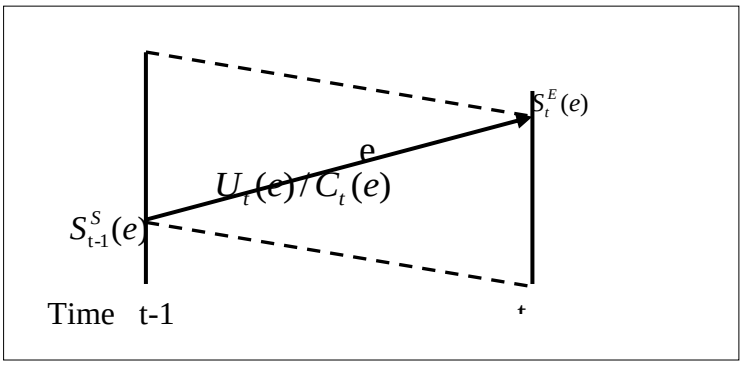

Fig.(2) An edge of a trellis section

1-The starting state $S_{t-1}^S(e)$ of edge e.

2-The ending state $S_t^E(e)$ of edge e .

3-The information bits (if any) $U_t(e)=[u_t^1(e),u_t^2(e).....u_t^{k_t}(e)]$ where $u_t^i(e)\in\{-1,+1\}$ is the i^{th} information bit associated with edge e at time t and k_t is number of information bits (if any) at section t.

4- The code bits $C_t(e)=[c_t^1(e),c_t^2(e).....c_t^{n_t}(e)]$ where $c_t^i(e)\in\{-1,+1\}$ is the i^{th} code bit associated with edge e at time t and n_t is number of code bits at section t.

The two input vectors of SISO max-log-MAP decoder are defined as:

1- $\lambda(C,I)=[\lambda(c^1,I),....,\lambda(c^N,I)]$ is a prior information LLR of a codeword.

2- $\lambda(U,I)=[\lambda(u^1,I),....,\lambda(u^K,I)]$ is a prior information LLR of information word.

To apply SISO max-log-MAP algorithm on *V*-section trellis we must divide each input sequence into *V*-group as follow:

$\lambda(C,I)=\{\lambda_1(C,I),...,\lambda_t(C,I),....,\lambda_V(C,I)\}$ where

$\lambda_t(C,I)=[\lambda_t(c^1,I),....,\lambda_t(c^{n_t},I)]$

and $\lambda(U,I)=\{\lambda_1(U,I),...,\lambda_t(U,I),....,\lambda_V(U,I)\}$ where $\lambda_t(U,I)=[\lambda_t(u^1,I),....,\lambda_t(u^{k_t},I)]$ and the output vector, $\lambda(U,O)=[\lambda(u^1,O),....,\lambda(u^K,O)]$, is the extrinsic information LLR of information. The SISO max-Log-MAP based on *V* section trellis algorithm performs first the two recursions:

1-The forward recursion of state s at time t is given by:

$$\alpha_t(s)=\max_{e:S_t^E(e)=s}\left\{\alpha_{t-1}(S_{t-1}{}^S(e))+\sum_{j=1}^{k_t}u_t^j(e)\lambda_t(u^j,I)+\sum_{j=1}^{n_t}c_t^j(e)\lambda_t(c^j,I)\right\}\quad t=1,2,\ldots,V \qquad (9)$$

Since the encoder starts with a known state s_o, the forward recursion will be initialized as $\alpha_o(s_0)=0$.

2- The backward recursion of state s at time t is:

$$\beta_{t-1}(s)=\max_{e:S_{t-1}^S(e)=s}\left\{\beta_t(S_t^E(e))+\sum_{j=1}^{k_t}u_t^j(e)\lambda_t(u^j,I)+\sum_{j=1}^{n_t}c_t^j(e)\lambda_t(c^j,I)\right\}\quad t=V\text{-}1,V\text{-}2,\ldots,0 \qquad (10)$$

Since the trellis is terminated to known state (s_o), then the backward recursion will be initialized as $\beta_V(s_0)=0$.

The output of the decoder is the extrinsic information LLR of information bits which can be divided into *V*-group $\lambda(U,O)=\{\lambda_1(U,O),...,\lambda_t(U,O),....,\lambda_V(U,O)\}$ where $\lambda_t(U,I)=[\lambda_t(u^1,O),....,\lambda_t(u^{k_t},O)]$ and $\lambda_t(u^j,O)$ is the extrinsic information of j_{th} information bit at t-section which can be obtained as:

$$\lambda_t(u^j,O)=\max_{e:u_t^j(e)=1}\left\{\alpha_{t-1}(S_{t-1}^S(e))+\sum_{\substack{i=1\\ i\neq j}}^{k_t}u_t^i(e)\lambda_t(u^i,I)+\sum_{i=1}^{n_t}c_t^i(e)\lambda_t(c^i,I)+\beta_t(S_t^E(e))\right\}$$

$$-\max_{e:u_t^j(e)=-1}\left\{\alpha_{t-1}(S_{t-1}^S(e))+\sum_{\substack{i=1\\ i\neq j}}^{k_t}u_t^i(e)\lambda_t(u^i,I)+\sum_{i=1}^{n_t}c_t^i(e)\lambda_t(c^i,I)+\beta_t(S_t^E(e))\right\}$$

t=1,2,..,V ; j=1,2,.., k_t (11)

B- SISO max-Log-MAP Decoder Based on Parallel Trellises Structure.

To apply max-log-MAP algorithm to such trellis, the output LLR of SISO decoder (Eq.(17))can be put in a form (Ye Lin, Shu Lio and Marc Fossorier, 2006):

$$\lambda(U,O)=\lambda^+(U,O)-\lambda^-(U,O) \quad (12)$$

Suppose a *V* sections trellis of an (N,K) binary linear block code is decomposed into W subtrellises T(1), T(2),…, T(W). The max-log-MAP decoder for the subtrellis T(w), $1\le w\le W$, find a pairs of $\lambda^+(U,O)(w)$ and $\lambda^-(U,O)(w)$ for each bit. The final surviving pairs of the entire trellis, ($\lambda^+(U,O),\lambda^-(U,O)$) are found by[3]:

$$\lambda^+(U,O)=\max\{\lambda^+(U,O)(w):1\le w\le W\} \quad (13)$$

$$\lambda^-(U,O)=\max\ \{\lambda^-(U,O)(w):1\le w\le W\} \quad (14)$$

Finally, the log-likelihood ratio of each bit is found by using Eq.(18) as shown in Fig(3).

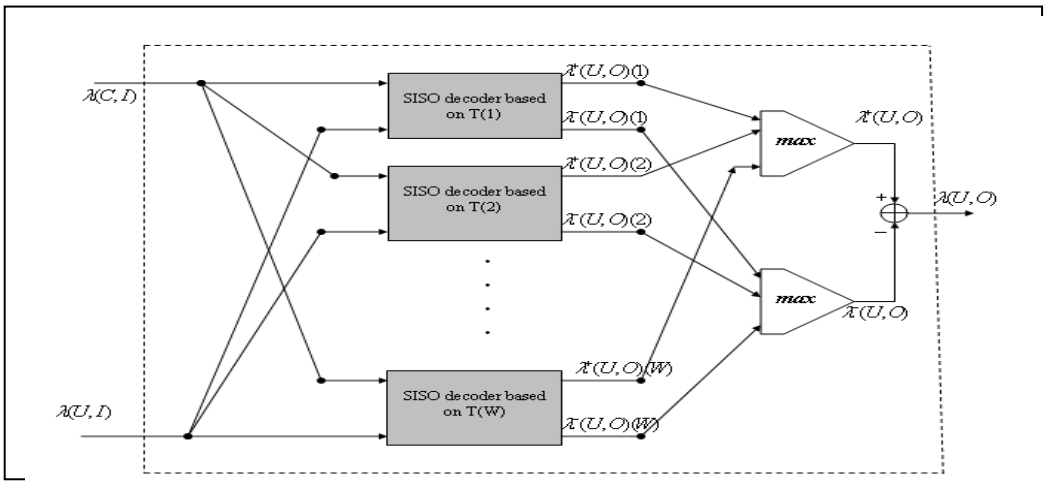

Fig.(3) SISO max-log-MAP decoder based on parallel trellises structure

From the stand point of speed, the effective computational complexity of decoding a received sequence, is defined as the computational complexity of a single parallel subtrellis plus the cost of the final comparison (max()) among the survivors generated by each of the subtrellis decoder. The time required for final comparison is generally small relative to the time required for processing a subtrellis. Furthermore,

since all the subtrellises processed in parallel, the decoding speed is, therefore, limited only by the time required to process one subtrellis. Consequently, this approach does not only simplify the decoding complexity but also gain speed (Ye Lin, Shu Lio and Marc Fossorier, 2006).

VI Decoding Complexity of (8,4) and (16,5) RM codes Based on parallel Trellises.

Reed Muller code is one of powerful codes that used in error correction coding and it its trellis can be decompensate into parallel trellises structure easily.

A- (8,4) RM code.

The trellis oriented generator matrix of (8,4)RM code is:

$$TOGM = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \quad (15)$$

By using the analysis present in section II

1-Active span of each row: aspan(r_1)=[1,3], aspan(r_2)=[2,6], aspan(r_3)=[3,5] and aspan(r_4)=[5,7].

2- Trellis complexity and primitive structure of the trellis can be found by finding the i_{th} left- and right-basis (l_i, r_i) then apply Eq.(2) to find The *i*th past and future subcodes, then apply Eqs.(5,6) to find the vertex space dimension v_i at depth *i* and the edge space e_i dimension at stage *i* is given in the table(2)

Table (2) Complexity analysis of (8,4)RM code

index i	0	1	2	3	4	5	6	7	8
l_i	X	1	1	1	0	1	0	0	0
r_i	X	0	0	0	1	0	1	1	1
f_i	4	3	2	1	1	0	0	0	0
p_i	0	0	0	0	1	1	2	3	4
v_i (SCP)	0	1	2	3	2	3	2	1	0
e_i	X	1	2	3	3	3	3	2	1
structure		<	<	<	>	<	>	>	>

By using Eqs.(3,4,7) total number of vertices, V(*C*) =34, total number of edges E(*C*) = 44 and total number of mergers *M(C)*=44-34+1=11.

The sectionalized trellis of (8,4) RM code, if the bounding location Q={0,2,4,6,8} (not include vertices 3,5 which have maximum state space dimension), is shown in Fig.(4).

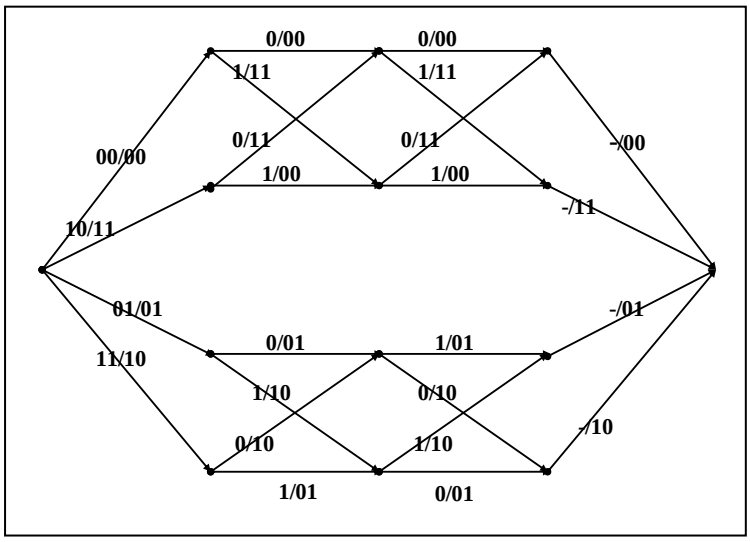

Fig.(4) Sectionalized trellis of (8,4) RM code

Where the state complexity profile (SCP) of this trellis is (0,2,2,2,0), and SCP of each subtrellis is v_i ={0,1,1,1,0} and edge space dimension={1,2,2,1}. Therefore

Number of vertices for each subtrellis, V(*C*) =8.
Number of edges for each subtrellis ,E(*C*) =12.
Number of mergers for each subtrellis ,*M(C)*=5.

To explain the complexity of SISO max-log-MAP decoder based on parallel trellises structures, trellis of (8,4) RM code has a four-section, 14-state minimal trellis diagram, which consists of two parallel and structurally identical eight-state subtrellises each is 12 edges and 5 mergers cross connections among them. As a result, we can devise two identical eight-state SISO max-log-MAP decoders to process the two subtrellises in parallel without communication between them. At the output, there are four LLR's (two for each subtrellis decoder) and the two comparators and subtractor will find the final will $\lambda(U,O)$. Therefore the complexity of SISO max-log-MAP for one subtrellis is:

1- Computing the forward recursion ($\alpha_t(s)$) requires:

>>Number memory =V(*C*) =8. Not that only 6 of them need to be calculated ($\alpha_0(s_0)=0$ and $\alpha_4(s_0)$ not useful) .Not that each memory will store floating number.

>>Number of *max()* operation = *M(C)*-1=4 (no need to calculate $\alpha_4(s_0)$).

>>Number of multiplications and additions:

a-The term $(\sum_{j=1}^{k_t} u_t^j(e)\lambda_t(u^j,I))$ need k_t additions and multiplications for each edge = (2×2)+(1×4)+(1×4) = 12 additions and 12 multiplications. (Note k_t =[2,1,1,0])

b-The term $\sum_{j=1}^{n_t} c_t^j(e)\lambda_t(c^j,I)$ need n_t additions and multiplications for each edge = (2×2)+(2×4)+(2×4)+(2×2)=24 additions and 24 multiplications. (Note n_t = [2,2,2,2])

c- There additions between terms (Eq.(13)) for each forward recursion=(3×6)=18

Then the total delay of computing forward recursions=$T_{max()}$ +T_{ADD} +$T_{MUL.}$ In the modern processing system these three time almost equal, then total delay=(4+2×(12+24)+18) ×T = 94×T .

2-Complexity for calculating backward recursion ($B_t(s)$) is same as the complexity of forward recursion.

3- Computing the ($\lambda^+(U,O)$) requires:

>>Number memory =4 (K=4)

>>Number of *max()* operation =4 (one for each bit)

>>Number of multiplications and additions:

a-The term ($\sum_{\substack{i=1 \\ i\neq j}}^{k_t} u_t^{i^i}(e)\lambda_t(u^i,I)$) need $k_t - 1$ additions and multiplications for each edge = (1×2) = 2 additions and 2 multiplications.

b- The term $\sum_{j=1}^{n_t} c_t^j(e)\lambda_t(c^j,I)$ need n_t additions and multiplications for each edge=(2×2)+(2×4)+(2×4)+(2×2)=24 additions and 24 multiplications.

c- Four additions between terms (Eq.(17)) for each value=(4×4)=16.

Then the total delay of computing forward recursions=$T_{max()}$ +T_{ADD} +$T_{MUL,}$ then total delay=(4+2×(2+24)+16) T = 72T

4- Complexity for calculating ($\lambda^-(U,O)$) is same as the complexity of ($\lambda^+(U,O)$).

The total delay of SISO max-log-MAP decode for each subtrellis=2×(94+ 72)T= 332T and total number of memory is =2×(6+ 4)=20. Since there are two parallel subtrellises then:

a-If the two decoder run at same time then we need 40memory unit to store($\alpha()$, $\beta()$, λ^+ and λ^-)unit plus 2×(4+8) memory unit to store ($\lambda(U,I)$ and $\lambda(C,I)$) as input for each decoder, while the delay of the SISO max-log-MAP decoder is 332T+(2×4)T+(1×4)T=344T (the final two *max()* operation and subtraction for each $\lambda(U,O)$ values) .

b-If the decoder of subtrellis run successively then we need only 20 memory unit plus (4+8)memory unit to store ($\lambda(U,I)$ and $\lambda(C,I)$) (the same memory locations used for each decoders) while an over all delay is (2×332T)+(2×4)T+(1×4)T=676T.

B- (16,5) RM code

The trellis oriented generator matrix of (16,5)RM code is:

$$TOGM = \begin{vmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{vmatrix} \quad (16)$$

1-Active span of each row: aspan(r_1)=[1,7], aspan(r_2)=[2,14], aspan(r_3)=[3,13] , aspan(r_4)=[5,11] and aspan(r_5)=[9,15].

2- Trellis complexity and primitive structure of this trellis is given in the table (3).

Table (3) Complexity analysis of (16,5)RM code

index i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
l_i	X	1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0
r_i	X	0	0	0	0	0	0	0	1	0	0	0	1	0	1	1	1
f_i	5	4	3	2	2	1	1	1	1	0	0	0	0	0	0	0	0
p_i	0	0	0	0	0	0	0	0	1	1	1	1	2	2	3	4	5
v_i (SCP)	0	1	2	3	3	4	4	4	3	4	4	4	3	3	2	1	0
e_i	X	1	2	3	3	4	4	4	4	4	4	4	4	3	3	2	1
structure		<	<	<	-	<	-	-	>	<	-	-	>	-	>	>	>

By using Eqs.(3,4,7) total number of vertices, V(*C*) =150, total number of edges E(*C*) =172 and total number of mergers *M(C)*=23

The sectionalized trellis of (16,5) RM code, if the bounding location Q={0,4,8,12,16} (not include vertices 5,6,7,9,10,11 which have maximum state space dimension), is shown in Fig.(5).

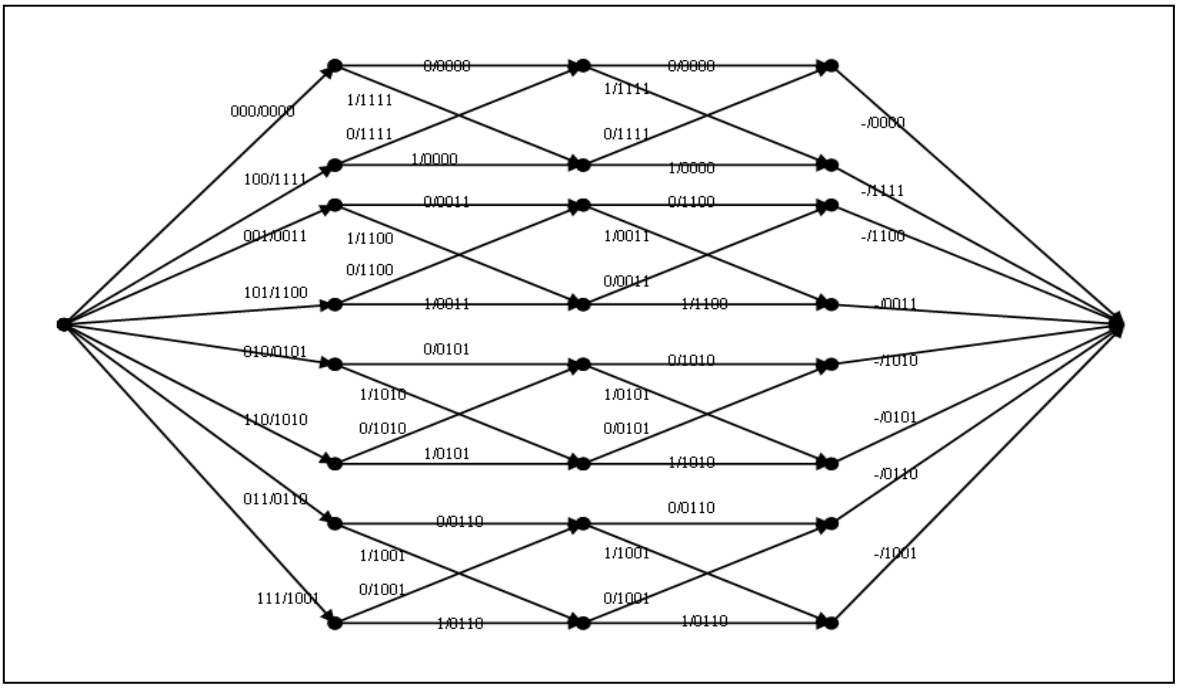

Fig.(5) Sectionalized trellis of (16,5) RM code

Where the state complexity profile (SCP) of this trellis is (0,4,4,4,0), and SCP of each subtrellis is v_i ={0,1,1,1,0} and edge space dimension={1,2,2,1}. Therefore

Number of vertices for each subtrellis, V(*C*) =8
Number of edges for each subtrellis ,E(*C*) =12
Number of mergers for each subtrellis ,*M(C)*=5

To explain the complexity of SISO max-log-MAP decoder based on parallel trellises structures, trellis of (16,5) RM code has a four-section, 26-state minimal trellis diagram, which consists of four parallel and structurally identical eight-state subtrellises without cross connections among them. As a result, we can devise four identical eight-state SISO max-log-MAP decoders to process the four subtrellises in parallel without communication between them. At the output, there are eight LLR's,

and the six comparator (three to find λ^{+}, and three to find λ^{-}) with subtractor will find $\lambda(U,O)$. Therefore the complexity of SISO max-log-MAP for one subtrellis is:

1- Computing the forward recursion ($\alpha_t(s)$) requires:

>>Number memory =V(C) =8. Not that only 6 of them need to be calculated ($\alpha_0(s_0)=0$ and $\alpha_4(s_0)$ not useful) Not that each memory will store floating number.

>>Number of *max()* operation = *M(C)*-1=4 (no need to calculate $\alpha_4(s_0)$)

>>Number of multiplications and additions:

a-The term ($\sum_{j=1}^{k_t} u_t^j(e)\lambda_t(u^j,I)$) need k_t additions and multiplications for each edge =(3×2)+(1×4)+(1×4)=14 additions and 14 multiplications.

b-The term $\sum_{j=1}^{n_t} c_t^j(e)\lambda_t(c^j,I)$ need n_t additions and multiplications for each edge = (4×2)+(4×4)+(4×4)+(4×2) = 48 additions and 48 multiplications.

c- There additions between terms (Eq.(9)) for each forward recursion=(3×6)=18

Then the total delay of computing forward recursions=$T_{max()}$ +T_{ADD} +$T_{MUL.}$ In the modern processing system these three time almost equal, then total delay=(4+2×(14+48)+18) ×T = 146T

2-Complexity for calculating backward recursion ($B_t(s)$) is same as the complexity of forward recursion.

3- Computing the ($\lambda^{+}(U,O)$) requires:

>>Number memory =5 (K=5)

>>Number of *max()* operation =5 (one for each bit)

>>Number of multiplications and additions:

a-The term ($\sum_{\substack{i=1 \\ i\neq j}}^{k_t} u_t^i(e)\lambda_t(u^i,I)$) need k_t-1 additions and multiplications for each edge = (2×2) = 4 additions and 4 multiplications.

b- The term $\sum_{j=1}^{n_t} c_t^j(e)\lambda_t(c^j,I)$ need n_t additions and multiplications for each edge=(4×2)+(4×4)+(4×4)+(4×2)=48 additions and 48 multiplications.

c- Four additions between terms (Eq.(17)) for each value=(4×5)=20

Then the total delay of computing forward recursions=$T_{max()}$ +T_{ADD} +$T_{MUL,}$ then total delay=(5+2×(4+48)+20) ×T = 129T

4- Complexity for calculating ($\lambda^{-}(U,O)$) is same as the complexity of ($\lambda^{+}(U,O)$).

The total delay of SISO max-log-MAP decode for each subtrellis=2×(146+ 129)T= 550T and total number of memory is 2×(6+5)=22. Since there are four parallel subtrellises then:

a-If the four decoder run at same time then we need 4×22memory unit to store($\alpha()$, $\beta()$, λ^{+} and λ^{-})unit plus 4×(5+16) memory unit to store ($\lambda(U,I)$ and $\lambda(C,I)$) as input for each decoder, while the delay of the SISO max-log-MAP decoder is 550T+(2×3×5)T+(1×5)T=585T (the final six *max()* operation and one subtractor for each $\lambda(U,O)$ values) .

b-If the decoders of subtrellis run successively then we need only 22 memory unit plus (5+16)memory unit to store ($\lambda(U,I)$ and $\lambda(C,I)$) (the same memory locations used for each decoders) while an over all delay is (4×550T)+(2×3×5)T+(1×5)T=2235T.

VII Simulation of Turbo Decoding of Parallel Concatenated Block Code using SISO max-log-MAP algorithm

Turbo codes can be constructed from two identical or two different component codes $C1$ and $C2$, in this paper we will discuss *Parallel Concatenated Block Codes* (PCBCs). It makes sense to take only codes in nonsystematic form as component codes, (do not allow a separation of information bits and parity bits). The information sequence **U1,** with K2 row and K1 column, are encoded using of encoder1 with parameter (N1,K1) to produce codeword **C1**,with K2 rows and N1 of column, then the information sequence **U1** pass throw row-by-column interleaver to produce **U2,** with K1 row and K2 column, then it encoded using of encoder2 with parameter (N2,K2) to produce codeword **C2**,with K1 rows and N2 of column((Peter, 2009; Ping, and Yeug , 1999).

The over all rate of encode is given by:

$$Rate = \frac{K1 \times K2}{(K2 \times N1) + (K1 \times N2)} = \frac{1}{\frac{N1}{K1} + \frac{N2}{K1}} = \frac{1}{\frac{1}{Rate1} + \frac{1}{Rate2}} \quad (17)$$

After the output code words (**C1**,**C2**) are modulated and transmitted throw channel, which consider AWGN channel with A.C noise power σ^2, the received sequence is multiply by the factor$(2/\sigma^2)$ and demultiplexed into $\lambda(C1,I)$, $\lambda(C2,I)$ then it passes throw turbo decoder.

During the first iteration, $\lambda(U1,I)$ is set to zero, since no a prior information is available on the input information bits of the encoder1. The extrinsic LLR's of the information bits of the decoder1, $\lambda(U1,O)$, are passed through the interleaver to obtain the LLR's of information bits of the encoder2, $\lambda(U2,I)$. The output of decoder2 , $\lambda(U2,O)$, after deinterleaving, is multiply by *Alfa* where it is a scaling factor used to reduce the effects of the extrinsic information in soft decoder in the first decoding steps when the BER is relatively high. It takes a small value in the first decoding step and increases as the BER tends to zeros. The values of *Alfa* with the decoding step are chosen as *Alfa*= [0.02, 00.3, 0.05, 0.07, 0.09,….].Then the output of deinterleaver is feedback to the lower entry (corresponding to information bits of the code1) of SISO decoder of decoder1 to start the second iteration .The final decision (for each iteration) is found by adding $\lambda(U2,O)$ and $\lambda(U1,I)$ then passed throw hard limiter then it compare with input information **U1** to find BER for each iteration at given SNR as shown in Fig.(6)((Peter , 2009; Ping, 2009; Crozier, and Hunt , 2008).

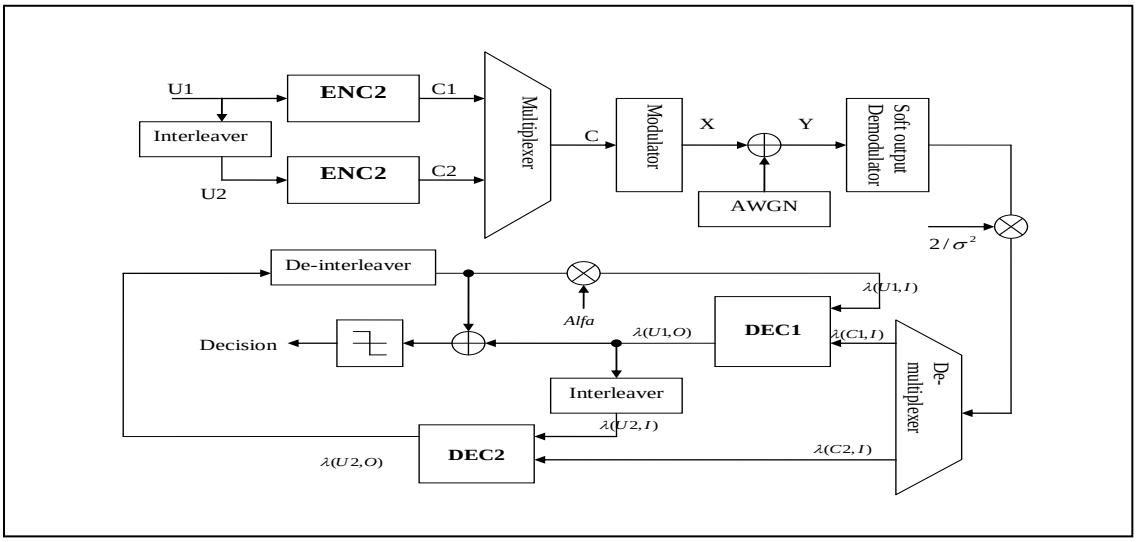

Fig.(6) Turbo decoding of parallel concatenated block code

This system is software implemented using MATLAB7 programming language where two code are used (8,4) and (16,5)RM codes:

The performance of turbo decoding parallel concatenated of two (8,4)RM codes using SISO max-log-MAP algorithm based on parallel trellises structure shown in Fig.(4) is shown in Fig.(7).

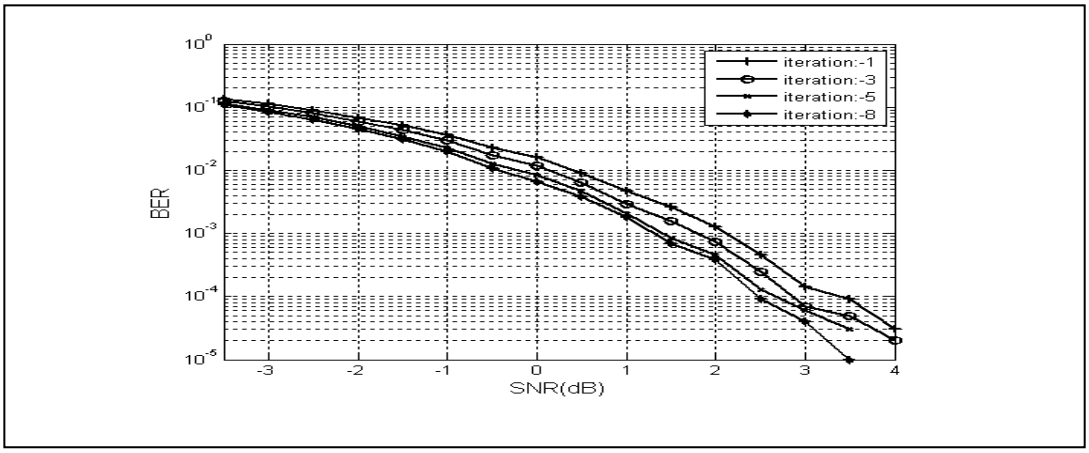

Fig.(7)BER performance of (8,4)RM turbo code

The performance of turbo decoding parallel concatenated of two (16,5)RM codes using SISO max-log-MAP algorithm based on parallel trellises structure shown in Fig.(5) is shown in Fig.(8).

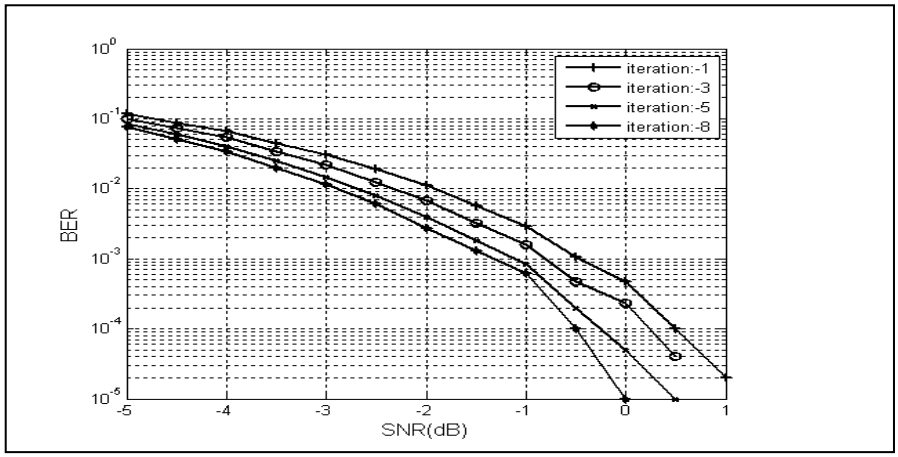

Fig.(8)BER performance of (16,5)RM turbo code

VIII Conclusions

We have presented an approach for decomposing the minimal trellis of a binary linear block code into parallel trellises structures that achieving low decoding complexity. This approach allows parallel processing of the subtrellises and does not increase the maximum number of states. Hence, it has significant speed advantage. Since the application of this method depends only on the generator matrix of the code, it can be applied to arbitrary linear block codes.

An iterative trellis based decoding, SISO max-log-MAP, algorithm of block codes was presented. The complexity of this algorithm was reduced by using parallel trellises structure. These results in reduced decoding complexity effer block codes with simple trellis representations. It was shown that the reduced-complexity decoder provides near optimal performance when a limit is placed on the number of iterations allowed. The computational complexity shows the of use parallel decoding of these subtrellises gives high decoding speed but need lager number of memories and the use of successive decoding of these subtrellises need few number of memories but decoding speed is low as compare with parallel processing as given in the table(4):

Table (4) Decoding complexity of (8,4) and (16,5)RM codes

Code	Parallel decoding		Successive decoding	
	Number of Memory	Delay	Number of Memory	Delay
(8,4)RM	64	344T	32	676T
(16,5)RM	172	585T	43	2235T

Reference

Benedetto S.; Divsalar D.; Montorsi G. and Pollara F., 2008."***Soft Input Soft Output Maximum A posteriori (MAP) Module to Decode Parallel and Serial Concatenated Codes***", TDA Progress Report 42-127, PP.1-20, November 15.

Berger Y. and Be'ey, Y. 2004. "***Soft Trellis-Based Decoding for Linear Block Codes***", IEEE Trans. on Inform. Theory, Vol.40, No.3, PP.764-773, May.

Crozier, S. Ken Gracie, and Hunt A. 2008. "Efficient Turbo Decoding Techniques" Commun. Research Center, pp.44-53.

Divsalar D. and Pollara, F.2007 . "***Hybrid Concatenated Codes and Iterative Decoding***", TDA Progress Report 42-130, PP.1-23, August 15.

Kiely, A. B.; Dolinar, S. and Kroot, L.E. 2005 . " ***Minimum Trellis for Block Codes and Their Duals***", TDA Progress Report 42-121, PP.148-158, May 15.

Luna, A. A. ; Fontani, F.M. and Wicker, S.B. 2008 . "***Iterative Maximum-Likelihood Trellis Decoding for Block Codes***", IEEE, Trasn. On Commun., Vol.47, No.3, PP.338-342, March.

Moorthy, H.T.; Lin, S. and Uehara, G.T. 2006 **. "** ***Good Trellis for IC Implementation for Viterbi Decoders for Linear Block Codes***", IEEE, Trans. Commun., Vol.45, No.1, PP.52-63, January .

Peter J. S., 2009 . "***Iterative Decoding of Parallel Concatenated High Rate Linear Block Codes***", MSC. Thesis, Univ. Notre Dame.

Peter J. S., and Daniel J.C., Jr., 2007 . "***MAP Decoding of Linear Block Codes Based on a Sectionalized Trellis of the Dual Code***", Univ. of Notre Dame, IN46556, PP.1-8.

Ping, L. and Yeug K. L. 2009 . ***"Symbol-by-Symbol APP Decoding of Golay code and Iterative Decoding of Concatenated Golay Codes***" IEEE Trans. Commun., Vol.45, No.7, PP.2558-2562, November.

Tapp, T.L.; Luna, A.A. ; Wang X.; and Wicher, S.B.,2007 . "***Extended Hamming BCH Soft Decision Decoders for Mobile Data Applications***", IEEE Trans. Commun., Vol.47, No.3, PP.333-337, March.

Ten Brink S. 2007. ***"Convergence Behavior of Iteratively Decoded Parellel Concatenated Codes***" IEEE Trans. Commun., Vol.49, No.10, PP.1727-1737, October.

Udayan D.G., 2006 . "***Low Complexity Techniques for Channel Decoding and Equalization***", Ph.D Thesis Univ. of Texas A &M, August.

Ye Lin, Shu Lio and Marc Fossorier, 2006 . "***MAP Algorithms for Decoding Linear Block Codes Based on Sectionalized Trellis Diagrams***", Univ. of Hawai at Monoa, PP.1-28.