PARALLEL STACK DECODING FOR MIMO SCHEMES

Abdellatif Salah †, Samuel Guillouard ‡, Ghaya Rekaya Ben Othman †

†TELECOM ParisTech - 46, rue Barrault, 75013 Paris, France
salah,rekaya@enst.fr
‡Thomson Corporate Research Cesson Sévigné France
samuel.guillouard@thomson.net

***Abstract*— Classical ML decoders for multiple input multiple output (MIMO) systems like the sphere decoder, the Schnorr-Euchner algorithm, the Fano and the stack decoders suffer from high complexity for high number of antennas and large constellation sizes. In this paper, we propose the use of parallel processing for stack decoding, to decode signals transmitted on linear MIMO channels to reduce time consumption of hardware architecture. It will be shown that the parallel stack decoder allows a 50% less of run time compared to the classical stack decoder.**

Index Terms - MIMO Antennas, Space Time Decoding, Sphere Decoding, Stack Decoding.

I. INTRODUCTION

Space time paradigm on wireless communications has seen a great explosion by a huge amount of both research oriented and applications papers. Many authors have considered combinations of space time codes with existing technologies such as Orthogonal Frequency Division Multiplexing (OFDM), LDPC codes and turbo codes, extension to frequency selective channels, multiple access channels and time varying channels, or performance studies under more realistic channel conditions. These are important avenues of research, especially since the WiMAX (IEEE 802.16) standard specifies an optional STBC Alamouti, and Golden Code transmission mode. Keeping the same usual standard linear system model, it's clear that the MIMO channel creates the opportunity for diversity and multiplexing gains but also increases the complexity to recover the transmitted symbols. Research community was been very active to establish detection strategies of QAM-modulated signals observed at the output of linear MIMO channels. Many popular leading algorithms were conceived such as the sub-optimal V-BLAST scheme, the ML sphere decoder (SD) and the ML Schnorr-Euchner (SE). Exhaustive Maximum Likelihood decoding is an optimal detection of signals transmitted over MIMO Channels, but a huge complexity is associated with this type of decoding since it's well known to be an NP-hard problem. SD and SE provide ML detection at a reduced computational complexity but it stays huge in practice and with expensive cost. The present paper is intended to improve another recently proposed decoder for MIMO schemes which is the stack decoder [6]. Here, we exploit a specific lattice representation to define stack decoding with parallel processing. Thus, many instructions can be carried out simultaneously which is not possible with SE and SD. This accelerates the decoder and permits to reach the ML point in less time.

This paper is organized as follows. In section II, we introduce the system model and the notations used later. Section III gives an overview of MIMO decoding. Stack decoding search strategy is given in section IV. In section V, we describe the new stack decoder based on a parallel processing. Finally, simulation results are presented in section VI.

II. SYSTEM MODEL

Consider a MIMO system that has M and N antenna elements at the transmitter and the receiver respectively. Each coefficient h_{ij} in the channel matrix represents the path fading between antenna j and the receiving antenna i. The received signal block can be written as follows :

$$\mathbf{y}_N = \mathbf{H}_{N\times M}\mathbf{x}_M + \mathbf{z}_N, \tag{1}$$

where $\mathbf{x}$ is the transmit signal vector. $\mathbf{H}$ is the channel matrix and $\mathbf{Z} \in N(0, \sigma^2\mathbf{I})$ is an additive white Gaussian noise. The system is considered coherent. The entries of the channel matrix $\mathbf{H}$ are complex.

Let's the invertible mapping $\Psi : \mathbb{C}^M \rightarrow \mathbb{R}^{\underline{M}}$ from complex valued vector $\mathbf{v}$ to real valued vector $\underline{\mathbf{v}}$ by stacking the real part of $\mathbf{v}$ over its imaginary part, be defined as :

$$\underline{\mathbf{v}} = \Psi(\mathbf{v}) = \begin{bmatrix} \mathfrak{Re}(\mathbf{v}) \\ \mathfrak{Im}(\mathbf{v}) \end{bmatrix}, \tag{2}$$

where $\underline{M} = 2M$.

When $N \neq 1$, the mapping $\Psi : \mathbb{C}^{N\times M} \rightarrow \mathbb{R}^{\underline{N}\times\underline{M}}$ and $\underline{N} = 2N$ from complex valued matrix $\mathbf{A}$ to real valued matrix $\underline{\mathbf{A}}$ is defined as follows :

$$\underline{\mathbf{A}} = \Psi(\mathbf{A}) = \begin{bmatrix} \mathfrak{Re}(\mathbf{A}) & -\mathfrak{Im}(\mathbf{A}) \\ \mathfrak{Im}(\mathbf{A}) & \mathfrak{Re}(\mathbf{A}) \end{bmatrix}. \tag{3}$$

Then, one can rewrite equation (1) by separating real and imaginary parts as follows :

$$\underline{\mathbf{y}}_{\underline{N}} = \underline{\mathbf{H}}_{\underline{N}\times\underline{M}}\underline{\mathbf{x}}_{\underline{M}} + \underline{\mathbf{z}}_{\underline{N}}. \tag{4}$$

We can obtain the same lattice representation when linear space-time codes are used.

III. MIMO DECODING

Optimal decoding of digital signals that have been distorted by linear filtering (wireless channel: $\mathbf{H}$) and corrupted by noise is a key and recurrent issue in digital communications. An optimal decoder is the one that finds the most likely candidate,

given the received observed signal. Such optimal decoder is said to be optimal in the sense of Maximum Likelihood (or ML).

Many kinds of MIMO detectors for multiple antenna channels are in use today. Hard output detectors estimate the transmitted symbol for each antenna and for each time period. These symbols are later converted into bits thanks to the de-mapper and the de-interleaver.

One can list the most current output detectors:

✓ Sub-optimal output decoders: estimate the transmitted symbols by linear equalizers followed by hard decision (e.g., Zero Forcing: ZF), Minimum Mean Square Error (MMSE) or non-linear equalizers (e.g., Decision feedback Equalizer: DFE). Those decoders don't provide optimum performances on MIMO channels even for high signal-to-noise ratios (SNR).

✓ Maximum likelihood output decoders: the maximum likelihood point can be found using exhaustive decoders which is very complex especially for high order constellation. To solve and simplify the exponentially complex search problem of ML exhaustive decoders for MIMO systems, lattice decoders (sphere decoder or Schnorr-Euchner decoder [1], [2], [3]) are used. The key idea of these algorithms is to enumerate lattice points inside a spherical search region centered on the received point.

✓ Sequential decoders: These decoders consist in a search inside a tree under a constraint of cost. In fact the decoding problem is converted to a search problem inside a tree (e.g., Fano decoder [4] and stack decoder [5], [6], [7]). One method of deriving the tree search structure emerges by considering the QR factorization of the lattice matrix. Thus, every matrix $\underline{\mathbf{H}}$ (verifying $\underline{M} \leq \underline{N}$) with linearly independent columns can be factored into :

$$\underline{\mathbf{H}} = \underline{\mathbf{Q}}\underline{\mathbf{R}}, \tag{5}$$

where $\underline{\mathbf{Q}}$ is $\underline{N} \times \underline{M}$ and orthogonal, and $\underline{\mathbf{R}}$ is $\underline{M} \times \underline{M}$ upper triangular and invertible.

Decoding problem aims at minimizing the cost function $|\underline{\mathbf{y}} - \underline{\mathbf{H}}\underline{\mathbf{x}}|^2$. Since the cost function is invariant under orthogonal transformation minimization problem can be written as

$$\begin{aligned} \arg\min_{\underline{\mathbf{x}} \in \Gamma^{\underline{M}}} |\underline{\mathbf{y}} - \underline{\mathbf{H}}\underline{\mathbf{x}}|^2 &= \arg\min_{\underline{\mathbf{x}} \in \Gamma^{\underline{M}}} |\underline{\mathbf{Q}}^T\underline{\mathbf{y}} - \underline{\mathbf{R}}\underline{\mathbf{x}}|^2 \\ &= \arg\min_{\underline{\mathbf{x}} \in \Gamma^{\underline{M}}} |\underline{\widetilde{\mathbf{y}}} - \underline{\mathbf{R}}\underline{\mathbf{x}}|^2, \end{aligned} \tag{6}$$

where Γ refers to symbols of alphabet (symbols in constellation) and $\underline{\widetilde{\mathbf{y}}} = \underline{\mathbf{Q}}^T\underline{\mathbf{y}}$ is called the orthogonally transformed target. The upper triangular structure of the factored matrix $\underline{\mathbf{R}}$ enables the decoder to decompose the equivalent cost function and to write it recursively as the sum of partial costs associated with the optimization variable. The accumulation of these costs over all $\underline{M}$ variables is precisely the value of the overall cost function. The summation lends naturally to a representation in a weighted $B - ary$ tree structure, where B is number of elements in Γ.

Note that the SE and the SD can be considered also as tree search algorithms.

IV. STACK DECODING

In the following, let's introduce the search strategy of the stack decoder. The stack algorithm is an example of Best First Search (BeFS [6]) decoder obtained by setting a limit cost. BeFS uses a cost function and always chooses the next node to be that with the best cost (the least one). It uses an agenda, but instead of taking the first node off the stack (and generating its successors) it will take the best node off, i.e., the node with the best cost. In the beginning, the decoder is in the root node. The decoder generates the children nodes and stores them in the stack. Nodes are ranked in increasing order with their costs. Then, the algorithm selects 'top' which is the node with lower cost (best-cost) inside the stack. The algorithm generates the children nodes of 'top' and deletes 'top' from the list. The algorithm ends when all vector components are found. (The size of 'top' is equal to the depth of the tree: leaf node reaches the top of the stack).

To get a complexity performance trade off, a bias is subtracted from the computed metric of each child node. The bias will favorite advanced paths in the tree. This variant is suboptimal compared to a ML decoding. Indeed, it enables a reduction of complexity.

If no constraint is stated ($bias = 0$), the stack decoder offers exact-ML performance. However, under some constraints ($bias \neq 0$), the stack decoder presents sub-optimal performances. In figure (3), a hardware architecture of the standard stack decoder is proposed. The stack architecture is composed of 3 main state machines(SM):

- the first one performs the nodes computation, especially the cost function,
- the second one ensures the stack management, i.e. the reordering of its elements according to their cost.
- The last one masters the overall processing, including the loading of the input data and the output of the decoding processing.

Due to the specificity of the stack algorithm, where the number of reading/writing operations in memory is large, it has been decided:

- to allow a concurrent operation of the 2 first state machines,
- to scale down the arithmetic processing capability so that both SMs operates in approximately the same duration for each new node computation.

The other components of the stack are memories or registers.

V. PARALLEL STACK DECODING

We propose here a new stack decoder based on a parallel processing. SD was parallel processed in [8] but with a loss in performance.

Parallel processing uses multiple compute resources simultaneously to solve a problem. The problem is broken into parts which are independent so that each compute resource can execute its part of the algorithm simultaneously with others. The lattice representation given by Ψ function in the first part of the paper imposes a major restriction on the tree search algorithm. Specifically, the search has to be executed serially from one level to another on the tree. Processing each level to

estimate symbols needs to have estimation of previous symbols which are necessary to calculate costs for each child. The standard stack decoding using the tree search starts at the top level (dimension =$\underline{M}$) and traverse down the tree with one level at a time, and computes for each step the costs of child nodes.

According to this lattice representation, it's impossible, for instance to calculate the cost for a node in level k without assigning an estimate for the levels before.

This approach means that decoding of any $\underline{x}(k)$ requires an estimate value for all preceding $\underline{x}(j)$ where $j = k-1, ..., \underline{M}$. The idea behind this work is to relax the tree search structure making it more flexible for parallelism. Thus, one can decode each pair of adjacent levels in the tree, and each level of this pair is independent from the other. For that, one should start by giving a second shape to the channel matrix representation. Instead of the Ψ function defined as in equation (3), we use another function Ω and we give another lattice representation defined in [8] as :

$$\begin{aligned} \underline{\tilde{\mathbf{H}}} &= \Omega(\mathbf{H}) \\ &= \begin{bmatrix} \Psi(H_{1,1}) & \cdots & \Psi(H_{1,M}) \\ \vdots & \ddots & \vdots \\ \Psi(H_{M,1}) & \cdots & \Psi(H_{M,M}) \end{bmatrix}, \end{aligned} \tag{7}$$

where we assume that Ψ can be applied to a complex component as for the matrices case.

Using this channel representation changes the order of detection of the received symbols to the following form

$$\begin{aligned} \underline{\tilde{\mathbf{y}}} &= \Omega(\mathbf{y}) \\ &= \begin{bmatrix} \mathfrak{Re}(\mathbf{y}_1) \\ \mathfrak{Im}(\mathbf{y}_1) \\ \vdots \\ \mathfrak{Re}(\mathbf{y}_M) \\ \mathfrak{Im}(\mathbf{y}_M) \end{bmatrix}, \end{aligned} \tag{8}$$

which means that the first and the second levels of the tree search correspond to the real and imaginary parts of $\underline{\mathbf{x}}(M)$.

After applying the QR decomposition, this structure becomes very interesting. In fact, due to the orthogonality between the columns of each set, all the elements $\underline{r}_{k,k+1}$ for $k = 1, 3, \cdots, \underline{M}$ in the upper triangular matrix are null. The localizations of these zeros are very important since they introduce orthogonality between the real and imaginary parts of every detected symbol. For example, for the 4×4 $\underline{\tilde{\mathbf{R}}}$ upper triangular matrix, we got the following form

$$\underline{\tilde{\mathbf{R}}} = \begin{bmatrix} \underline{r}_{1,1} & 0 & \underline{r}_{1,3} & \underline{r}_{1,4} \\ 0 & \underline{r}_{2,2} & \underline{r}_{2,3} & \underline{r}_{2,4} \\ 0 & 0 & \underline{r}_{3,3} & 0 \\ 0 & 0 & 0 & \underline{r}_{4,4} \end{bmatrix}. \tag{9}$$

Using this example, figure (1) defines the parallel stack decoder which will treat two layers in each step by duplicating the treatment and keeping only one memory to store children. The new lattice representation permits to calculate two dimensions in only one step thanks to the independence between the two last layers of the generating lattice matrix. Then, the run time is twice lower since two dimensions are crossed over for each node computation. Figure (4) shows the first step in parallel decoding. we use: n to design node, b to design branch, σ is the node cost and w is the branch weight. It consists on generating two layers simultaneously. Paths $[\underline{x}_3\underline{x}_4]$ are then generated, and one gets 4 candidates which are the combination of two possible estimation for each tree, costs are also combined (the sum). In the following step, the algorithm will select the best candidate among these four and will proceed similarly.

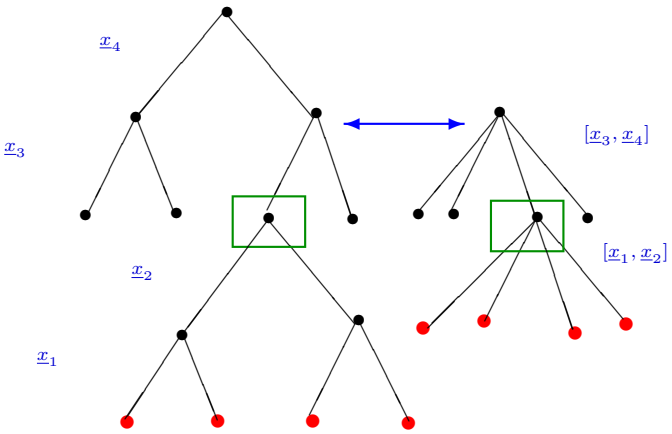


Fig. 1. Parallel processing : Four dimension in two steps

Assume the decoder chooses $[\underline{x}_3\underline{x}_4] = [1, 1]$. In the next step the decoder acts as follows: the list of new candidates found in the step described in figure(5) is added to the previous list. The algorithm continues to run by selecting the most promising node among all nodes in the complete list stored in the stack. This example gives an illustration overview of the general algorithm to understand the idea behind parallel processing for stack decoding. The proposed structure is optimal in performance and keeps the same ML results. In the figure(6), we propose a parallel stack decoder hardware architecture.

VI. SIMULATIONS AND RESULTS

In this part, results of hardware realizations of the two architectures described in figures (6) and (3) are given. All the modules in the design are coded in VHDL as it is a very useful tool with its concept of concurrency. More state machines are in use for the architecture of the parallel stack decoder in figure (6) exploiting the parallelism which enhances the decoder speed.

To illustrate the use of the new decoder, a 2×2 MIMO scheme using a convolutional code of rate $R = \frac{1}{2}$ for a $16-QAM$ constellation under a Rayleigh flat fading channel was used. Monte Carlo Simulations are used with 10^6 distinct channel realizations, where the channel realization remains constant over one signal block and changes randomly from block to block. We assume that channel state information is available at the receiver.

In figure (2), we compare the standard stack decoder architecture with the parallel proposed one in terms of the needed time to decode one codeword. Both of them provide ML performances. The parallel stack decoder outperforms the standard one in terms of rune time. It reduces decoding time for one codeword to the half of time elapsed by the standard stack decoder.

VII. CONCLUSION

In this paper, stack decoder is shown to be capable of supporting parallel processing. Our main contribution was to exploit a novel lattice matrix representation to make the parallel processing possible. By exploiting the advantage of the new sparser matrix a good improvement in run time was distinguished. The simulation results show that the proposed parallel decoder outperforms the classical one in terms of complexity, keeping the same ML performances. Moreover, due to the stack decoder properties, the parallel processing is also easily implemented in practice. This algorithm can be further enhanced by looking for space time codes introducing more zeros in the lattice representation matrix. Thus, parallel processing can be done for 3 layers or even more in only one step.

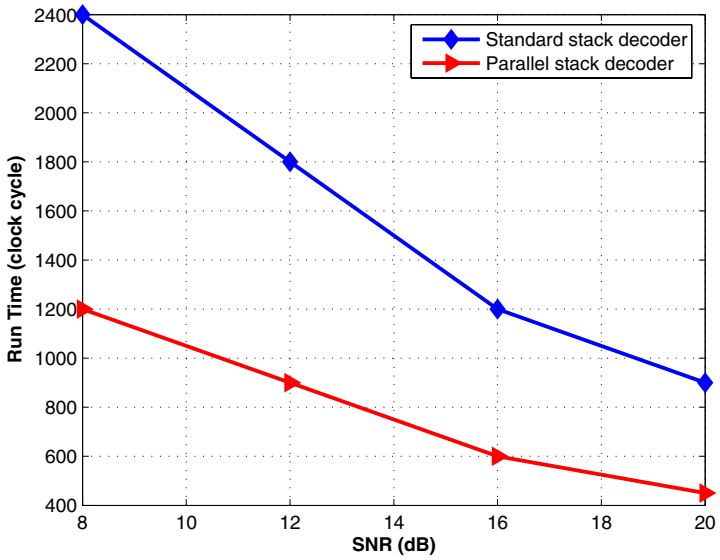


Fig. 2. Comparison of standard stack decoder and parallel stack decoder in terms of needed time to decode one codeword

REFERENCES

[1] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger. "Closest point search in lattices," *IEEE Transactions On Information Theory*, 48:2201-2214, August 2002.

[2] B.Hassibi and H.Vikalo."On the expected complexity of sphere Decoding," *in proceedings of Asilomar Conference on Signals, Systems, and Computers*, 2:1051-1055, 2001.

[3] J. Boutros and E. Viterbo."A Universal Lattice Code Decoder for Fading and Channels," *IEEE Transactions On Information Theory*, 45:1639-1642, July 1999.

[4] R. M. Fano."A heuristic discussion of probabilistic decoding," *IEEE Transactions On Information Theory*, 9:64-73, April 1963.

[5] F.Jelinek."Fast Sequential Decoding Algorithm Using a Stack," *IBM J. Res. Develop.*,13:675-685, November 1969.

[6] Arul D. Murugan, H. El Gamal, M. O. Damen, and G. Caire."A Unified Framework for Tree Search Decoding: Rediscovering the Sequential Decoder," *IEEE Transactions. On Information Theory*, 52:933-953 March 2006.

[7] K.Sh. Zigangirov."Some Sequential Decoding Procedures," *Probl. Peredach. Inform.*, 2 :13-25, 1966.

[8] L. Azzam, E.Ayanoglu"Reduced Complexity Sphere Decoding for Square QAM via a New Lattice Representation," *IEEE GLOBECOM 2007*, Washington, DC, November 2007.

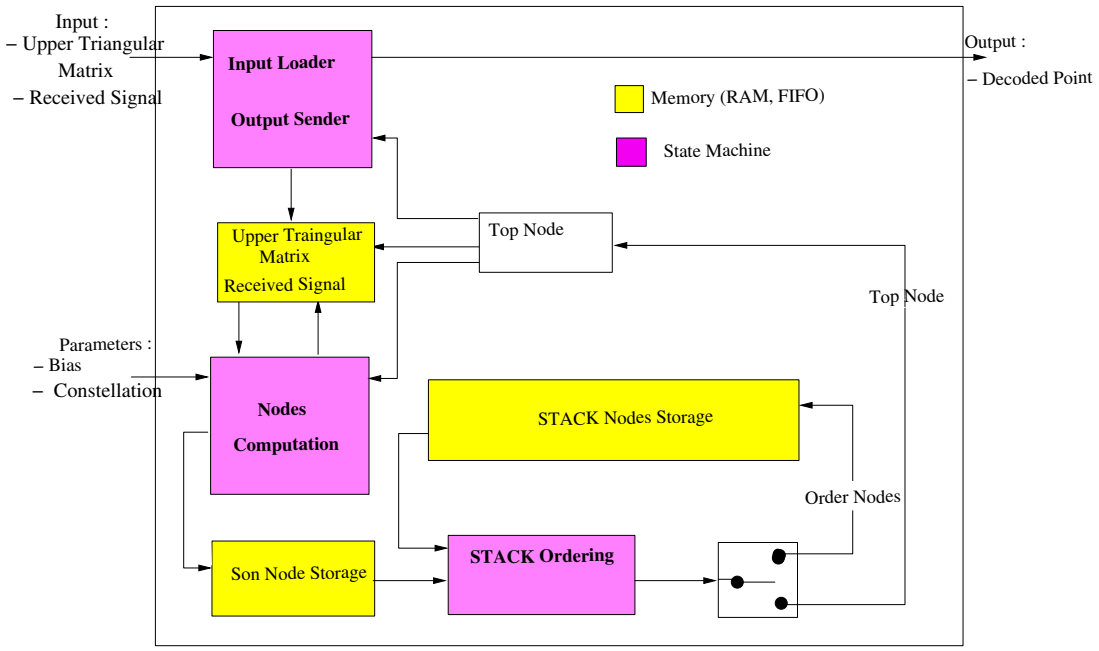
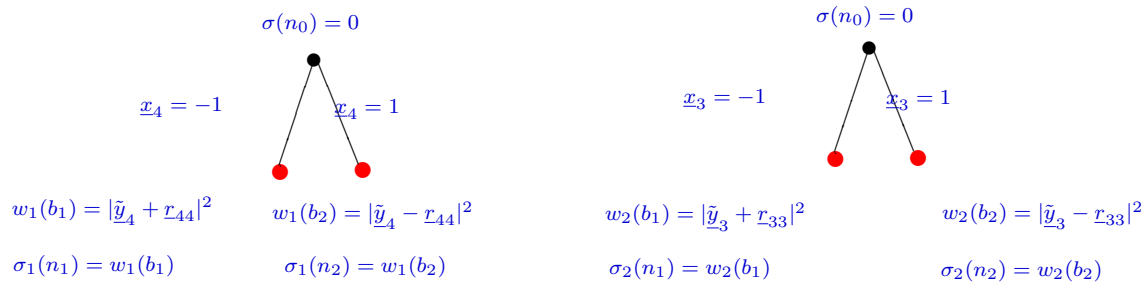
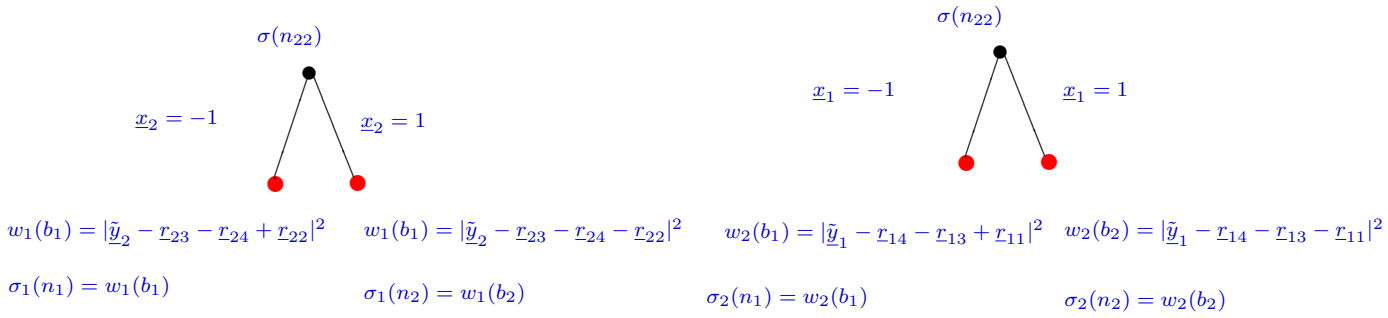


Fig. 3. The Standard Stack Decoder Hardware Architecture



$[\ \underline{x}_3\underline{x}_4]$	$\sigma(n_{kl}) = \sigma_1(n_k) + \sigma_2(n_l) + \sigma(n_0)$
[-1,1]	$\sigma(n_{21}) = \sigma_1(n_2) + \sigma_2(n_1) + \sigma(n_0)$
[-1,-1]	$\sigma(n_{11}) = \sigma_1(n_1) + \sigma_2(n_1) + \sigma(n_0)$
[1,1]	$\sigma(n_{22}) = \sigma_1(n_2) + \sigma_2(n_2) + \sigma(n_0)$
[1,-1]	$\sigma(n_{12}) = \sigma_1(n_1) + \sigma_2(n_2) + \sigma(n_0)$

Fig. 4. Parallel Processing Stack Decoding : estimation of $\underline{x}_4$ and $\underline{x}_3$



$[\ \underline{x}_1\underline{x}_2\underline{x}_3\underline{x}_4]$	$\sigma(n_{kl}) = \sigma_1(n_k) + \sigma_2(n_l) + \sigma(n_{22})$
[-1,1,1,1]	$\sigma(n_{21}) = \sigma_1(n_2) + \sigma_2(n_1) + \sigma(n_{22})$
[-1,-1,1,1]	$\sigma(n_{11}) = \sigma_1(n_1) + \sigma_2(n_1) + \sigma(n_{22})$
[1,1,1,1]	$\sigma(n_{22}) = \sigma_1(n_2) + \sigma_2(n_2) + \sigma(n_{22})$
[1,-1,1,1]	$\sigma(n_{12}) = \sigma_1(n_1) + \sigma_2(n_2) + \sigma(n_{22})$

Fig. 5. Parallel Processing Stack Decoding : estimation of $\underline{x}_1$ and $\underline{x}_2$

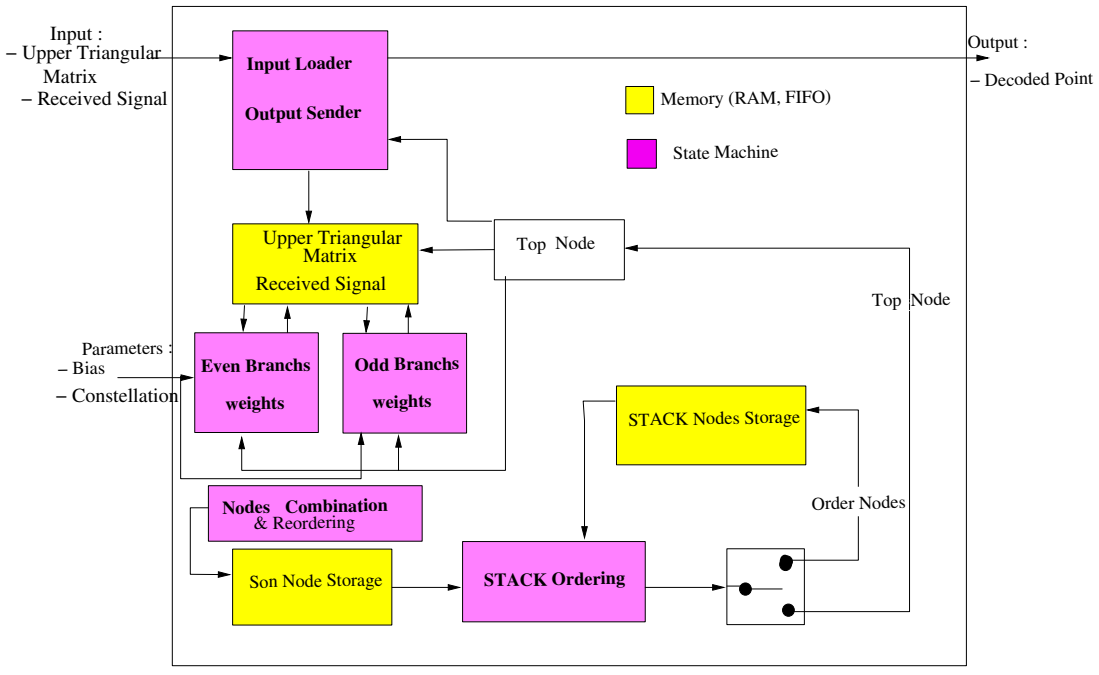


Fig. 6. Parallel Stack decoder hardware architecture