Dynamically Configurable Systolic-like Architecture for Close to Optimal Soft MIMO Detection

***Abstract*— A novel configurable processor architecture is presented for detecting Spatially Multiplexed (SM) data in a high order (4x4) multiple-input-multiple-output (MIMO) wireless systems. It is a customized architecture to match an algorithm that provides "soft" values to the Forward Error Decoder (FEC), with systolic-like data and control flow. The processor is able to switch between three different modulation schemes (QPSK, 16-QAM, and 64-QAM) without any configuration latency. Preliminary synthesis results indicate that each detector core uses only about 18 Kilo Gate Equivalent (KGE) in addition to around 1800 storage flip-flops. Multiple detector cores can be stacked to achieve very high throughput in a multi-carrier MIMO system. Moreover, owing to the careful choice of the underlying algorithm the Packet Error Rate (PER) is close to optimal. The processor is very well suited for baseband processing at wireless base-stations and mobile handsets alike.**

I. Introduction

Most future wireless systems will likely have MIMO as a key enabling technology because of its ability to provide high data rates at no extra bandwidth. One of the key component of a SM based MIMO receiver is the detector to *demultiplex* the transmitted data. Broadly speaking detectors are of two varieties: hard output and soft output. In terms of Packet Error Rate (PER), the soft detectors perform much better than their hard counterparts [1]. As it is, high speed implementation of soft detectors is a difficult challenge. On top of this, future wireless standards/devices will require to configure to support various system parameters such as modulation and coding schemes (MCS) [2]. Recently there have been many high speed configurable hard detectors presented in open literature [3,4]. Unfortunately, relatively few configurable soft detectors have been presented so far, some of the notable ones are presented in [5,6]. The detector in [5] uses Layered ORthogonal Detection (LORD) algorithm [7], while the one in [6] uses a *linear* Minimum Mean Square Error (MMSE) approach to generate soft output. LORD outperforms the lower complexity MMSE based soft detector by a significant margin in a wide variety of channel conditions and MCS [7]. Authors in [8] also provides an excellent comparison of various detectors for various MCS and channel conditions; and concludes that the linear detectors are attractive only for channels with high diversity, low code rate and low order modulation schemes.

Configurability and high throughput requirements often place conflicting demands on the hardware architect. The choice of algorithm is vital to meet the error rate performance and baseband throughput requirements. Algorithm space for MIMO detection is extremely vast. After significant research we have concluded that the LORD algorithm holds a lot of promise to solve the problem of designing configurable soft MIMO detector hardware. In this paper we present a high speed configurable systolic-like processor that can switch between three modulation schemes QPSK, 16-QAM, and 64-QAM on-the-fly.

The paper is organized as follows: Section II describes the basics of the channel model and the detection algorithm. In Section III we present the architectural details of the detector. Section IV concludes the paper by identifying future research directions.

II. MIMO Detection

A. Channel Model and MIMO detection

A model for MIMO system with M_T transmit and M_R receive antennas can be expressed as shown in (1)[1].

$$\mathbf{y} = \mathbf{Hs} + \mathbf{n} \tag{1}$$

where $\mathbf{y}=[y_1, y_2, ..., y_{M_R}]^T$ is a $M_R \times 1$ received vector, $\mathbf{s}=[s_1, s_2, ..., s_{M_T}]^T$ is $M_T \times 1$ transmitted vector, $\mathbf{n}$ is $M_R \times 1$ zero mean complex Gaussian noise vector, and $\mathbf{H}$ is a $M_R \times M_T$-dimensional complex matrix. In this paper we will assume $M_T = M_R = 4$, unless specified otherwise.

Each entry s_i $(i = 1, 2, ..M_T)$ in the vector $\mathbf{s}$ is an η-ary Quadrature Amplitude Modulated (QAM) symbol. Each QAM symbol is constructed by *mapping* $\log_2 \eta$ data bits onto a complex number. The objective of the MIMO detector is to estimate $\hat{\mathbf{s}}$ of $\mathbf{s}$ based on the the observation of $\mathbf{y}$ along with the knowledge of $\mathbf{H}$. It has been shown that the optimal or the Maximum Likelihood (*ml*) hard estimate $\hat{\mathbf{s}}_{ml}$ of $\mathbf{s}$ is given by (2) [1]:

$$\hat{\mathbf{s}}_{ml} = arg \min_{\mathbf{s} \in \Omega^{M_T}} \|\mathbf{y} - \mathbf{Hs}\|^2 \tag{2}$$

A straightforward approach to solving (2) is an exhaustive search over all possible candidate vectors $\mathbf{s}$. We can circumvent the exhaustive search by evaluating only a small subset of all the possible vectors. This can be achieved by noting that $\mathbf{H}$ can be triangularized using QR decomposition (QRD): $\mathbf{H} = \mathbf{QR}$. where, $\mathbf{R}$ is an upper triangular matrix, and $\mathbf{Q}^H$ is the Hermitian of a unitary matrix $\mathbf{Q}$. Hence, the cost function given by (2) can now be rewritten as [1],

$$\hat{\mathbf{s}} = \|\mathbf{y} - \mathbf{Hs}\|^2 = \|\hat{\mathbf{y}} - \mathbf{Rs}\|^2, and\ \hat{\mathbf{y}} = \mathbf{Q^H y} \tag{3}$$

Above equation can be further expanded as shown in (4)-(6).

$$d_i(s^{(i)}) = d_{i+1}(s^{(i+1)}) + |e_i(s^{(i)})|^2 \tag{4}$$

$$|e_i(s^{(i)})|^2 = |c_{i+1}(s^{(i+1)}) - R_{ii}.s_i|^2 \tag{5}$$

$$c_{i+1}(s^{(i+1)}) = \hat{y}_i - \sum_{k=i+1}^{M_T} R_{ik}.s_k \quad (6)$$

It is well known that using (4)-(6) the *ml* estimate can be treated as a search on a η-ary tree (with *i*=4 denoting children of the dummy root node and *i*=1 denoting leaf nodes) with *pruning* large portions of the tree (sphere detector) [10]. The term $d_i(s^{(i)})$ will be called Partial Euclidean Distance (PED) for $i > 1$, and Euclidean Distance (ED) for $i = 1$. The *ml* hard estimate is the *best* path/vector (path with least ED) from root to the leaf node.

The objective of a soft MIMO detector is to compute the *reliability* associated with each hard output bit. This reliability is expressed in terms of the Log-Likelihood Ratio (LLR) of each bit, and is defined as $L(x_{i,j}) = ln\frac{P(x_{i,j}=1|\mathbf{y})}{P(x_{i,j}=0|\mathbf{y})}$, where $x_{i,j}$ is j^{th} bit in label of the i^{th} constituent QAM symbol of **s**. *Max-Log-Map* approximation of this can be expressed as [1]:

$$L(x_{i,j}) \approx \min_{\mathbf{s}\in\mathbf{X}_{i,j}^{(0)}}\{d(\mathbf{s})\} - \min_{\mathbf{s}\in\mathbf{X}_{i,j}^{(1)}}\{d(\mathbf{s})\} \quad (7)$$

where $d(\mathbf{s})=||\mathbf{y} - \mathbf{Hs}||^2$, and the variables $\mathbf{X}_{i,j}^{(0)}$ and $\mathbf{X}_{i,j}^{(1)}$ are sets of vector, with j^{th} bit in the label of i^{th} QAM symbol in **s**, as 0 and 1 respectively.

One of the two terms in (7) will always correspond to the *ml* hard estimate because its $(i,j)^{th}$ label has to be either a 1 or a 0. If we denote the ED $d(\hat{\mathbf{s}}^{ml})$ with just d^{ml} then (7) can be rewritten as [11]:

$$L(x_{i,j}) = d^{ml} - d_{i,j}^{\overline{ml}}, x_{i,j}^{ml} = 0 \quad (8)$$

$$= d_{i,j}^{\overline{ml}} - d^{ml}, x_{i,j}^{ml} = 1 \quad (9)$$

Where $d_{i,j}^{\overline{ml}}$ is the ED of the best path with $(i,j)^{th}$ bit that is complement of the $(i,j)^{th}$ bit of the *ml* path. It is clear that to evaluate (7), we need to compute $\hat{\mathbf{s}}^{ml}$, d^{ml}, and $d_{i,j}^{\overline{ml}}$ for *i*=1,2,..M_T and *j*=1,2,...,$log_2\eta$ [12].

B. Layered ORthogonal Decoding (LORD)

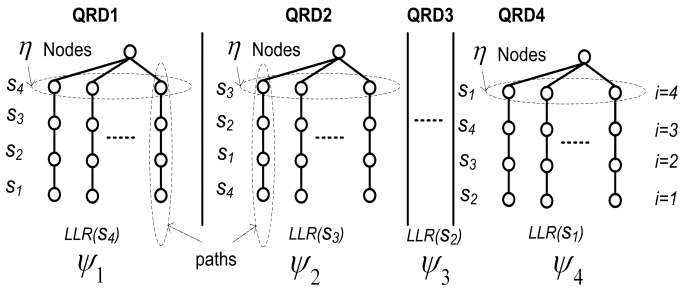

Fig. 1. LORD Algorithm Flow

Recently a new algorithm called LORD has been proposed in [7,9]. Note that in [9] the authors focus on the iterative turbo-MIMO systems. However, in the same paper they have shown that LORD (with some enhancements) performs close to optimal even in a non-iterative MIMO system (which is the system we consider in this paper).

LORD searches for $\hat{\mathbf{s}}^{ml}$, d^{ml}, and $d_{i,j}^{\overline{ml}}$ in a much reduced space (fig.1). It computes the LLRs for *individual* QAM symbols sent from four different transmit antennas. It does this by generating four sets (each of cardinality η) of paths and then searching for the required terms in these sets. A set of paths is constructed by evaluating all the children of the root node and best child of these nodes down the tree as shown. For example, to compute LLRs for the QAM symbol s_4 (transmitted from antenna no.4) it constructs the set ψ_1 of paths and searches for the LLR terms within it. Note that $\hat{\mathbf{s}}^{ml}$ is now actually a scalar $\hat{s_4}^{best}$, d^{ml} is d^{best}, and $d_{i,j}^{\overline{ml}}$ is $d_{i,j}^{\overline{best}}$ (we use superscript *best* because it is the best in the constructed set and not necessarily *ml*). It then permutes the columns of **H** matrix such that s_3 appears at *i*=4 of the tree, this entails another QRD (QRD2) computation. The above explained process is then repeated to get the LLRs for s_3. After four such iterations, LLRs for all the QAM symbols in **s** can be computed.

Authors in [9] have also suggested *metric-recycling* enhancement that significantly improves its error rate performance. The main observation is that the LLR terms computed in one set may be improved in another set. For example, if d^{best} and $d_{i,j}^{\overline{best}}$ are associated with $s_4 = a$ in ψ_1, it is possible that $s_4 = a$ (at $i = 3$) can occur in the ψ_4 with better d^{best} and/or $d_{i,j}^{\overline{best}}$. In this case the algorithm uses the better EDs to compute the LLRs. This essentially means that the enhanced LORD searches for the LLR terms in the augmented set $\psi = \psi_1 \cup \psi_2 \cup \psi_3 \cup \psi_4$, and not surprisingly its error rate performance improves.

III. On-the-Fly Configurable MIMO Detector

A. Detector Architecture

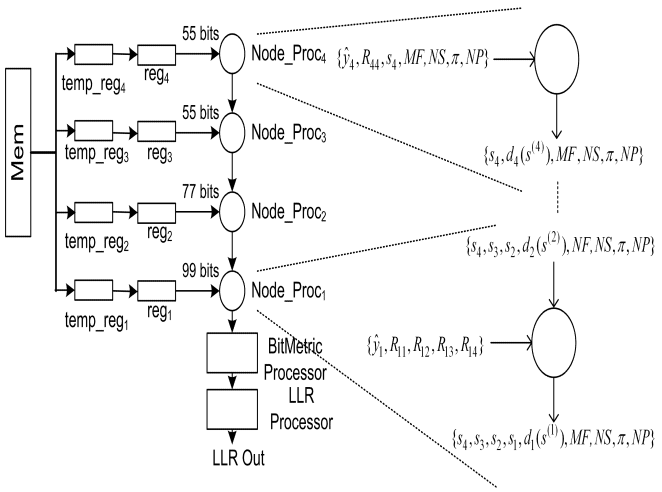

Fig. 2. High Level Architecture of the Detector

Fig.2 shows the high level architecture of the proposed decoder. It consists of a array of tightly coupled heterogeneous processors arranged as shown. Each node processor is fed the data $(\hat{y}_i, R_{ik}, s_k)$ via dedicated flip-flop registers $(reg_4, reg_3$ etc.). The control instructions MF, NS, π, NP are fed through the topmost node processor. The temporary registers ($temp\_reg_4$ etc.), communicate with memory which

stores the matrix and control data for multiple received vectors. The new matrix data is loaded into the dedicated registers at every new permutation (signal NP denotes this event). π is a two bit signal that informs the detector about the permutation order. This signal is used to *de-permute* elements in **s** so that the correct QAM symbols gets compared during metric-recycling (see section II.B). The signal NS denotes the arrival of new MIMO symbol into the detector array. MF is a two bit signal that indicates the modulation format of the vector symbol being processed. The signal NP/NS stay high only during the first cycle of a new permutation/vector symbol. Each of the processor is pipelined to improve the operating frequency. The node processors computes (6)-(4) for *i*=4,3,2,1. These node processors switch between modulation schemes by a specially designed "slicer" (which acts on the value of MF) [3]. We will omit further discussion on node processors for the sake of brevity, please refer to [3,5] for more details.

BitMetric Processor: The BitMetric processor (fig.3) carries out the task of bit selections and metric comparisons to compute the LLR terms. For the sake of clarity in the figures, let $a_{i,j}$ denote the $(i,j)^{th}$ bit of the current best path, $b_{i,j}$ denote the $(i,j)^{th}$ bit of the new (or incoming) path, and let $c_{i,j}$ denote $d_{i,j}^{\overline{best}}$. Furthermore, let d_a denote d^{best}, and d_b denote the metric of the incoming path. The processor operates concurrently on the data stored in memory locations $a_{i,j}$, $b_{i,j}$, $c_{i,j}$ and d_a, d_b, where $i = 1,2,...4$ and $j = 1,2,...log_2 64$. Similar to [11], the operation of bit selection and ED comparisons can be expressed as follows: If $d_a > d_b$, it means the incoming path is the new best path. Hence, for all i,j where $a_{i,j}$ and $b_{i,j}$ differ, d_a is assigned to $c_{i,j}$. This would be followed by assignments $a_{i,j}$=$b_{i,j}$, and d_a=d_b. If $d_a < d_b$, it means the incoming path cannot be a new best path. It may however, still effect the EDs $c_{i,j}$. Hence, for all i,j where $a_{i,j}$ and $b_{i,j}$ are complements of each other, the processor will assign $c_{i,j}$=d_b if $d_b < c_{i,j}$. Above procedure for LLR update can be broken down into two fundamental steps.

1) Decide on the memory locations to be updated,i.e. the ordered pairs (i,j)'s: These locations are always the ones where the $a_{i,j}$ and $b_{i,j}$ differ, and hence can be implemented using an X-OR operation (in a cell in bit manager).
2) Since (i,j)'s are now known we have to decide what to update $c_{i,j}$ with.
   a) If $d_a > d_b$: Replace $c_{i,j}$ with d_a. Then, replace $a_{i,j}$ with $b_{i,j}$, and d_a with d_b
   b) If $d_a < d_b$: Replace $c_{i,j}$ with d_b if $c_{i,j} > d_b$. Comparator *cmp1* checks for these conditions and instructs, via MUX M3 (in central manager), whether d_a or d_b is the candidate for replacing $c_{i,j}$. Compare select unit *cmpsel* further "filters" the candidate by comparing it with the existing value of $c_{i,j}$ (in a cell in bitmetric manager).

The π^{-1} block de-permutes the incoming bits for correct alignment of QAM symbols in different ψ_i's (see fig.1). The

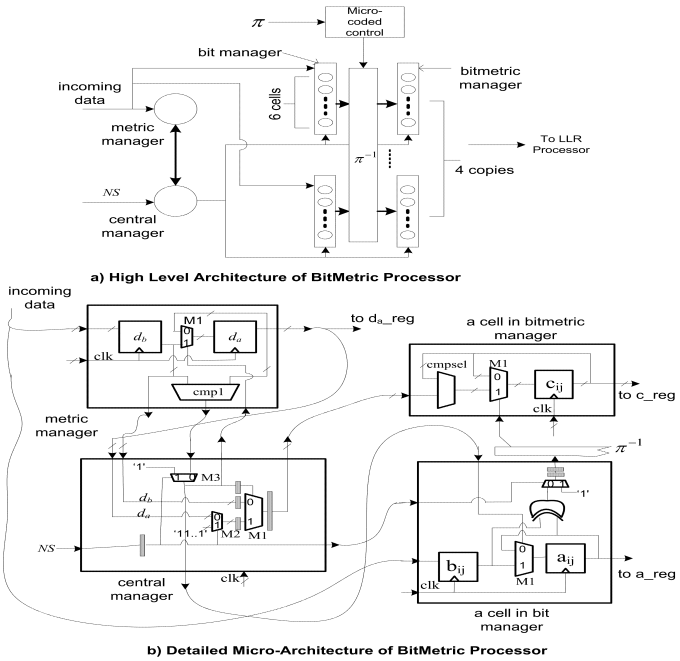

Fig. 3. BitMetric Processor

signal NS is used to enable the processing of new received vector by initializing the BitMetric processor. *LLR Processor*: The LLR processor consists of storage elements, $c\_reg_{i,j}$ for $d_{i,j}^{\overline{best}}$, $a_{i,j}$ for bits in $\hat{s}_i{}^{best}$, and d_a_reg for d^{best}, where $i = 1,2,3,4$ and $j = 1,2..,log_2 64$. The processor serially reads $d_{i,j}^{best}$, $a_{i,j}$ from the addresses based on the value of MF. For example, if MF=0 then it reads from $i = 1,2,3,4$, $j = 1,2$, and if MF=2 then locations $i = 1,2,3,4$, $j = 1,2...,6$ are read from. The final LLR values are computed using (8,9).

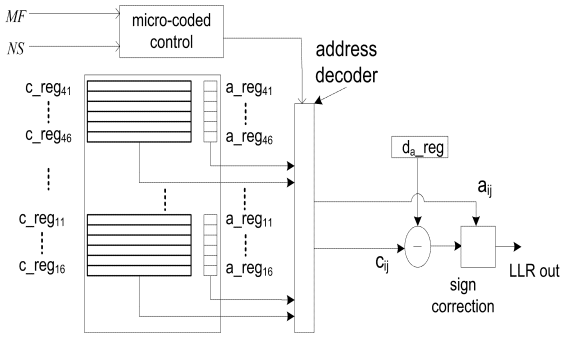

Fig. 4. LLR Processor

Throughput Analysis: Let the cumulative number of pipeline stages in node processor array be n1, n2 in the BitMetric Processor, and n3 in the LLR processor. Generally, an η-ary QAM vector symbol will spend n1+4η cycles in the node processor array, n2+4η cycles in the BitMetric processor. Finally, the serialized LLR processor will take n3+1+4$log_2\eta$ cycles to compute the LLRs. Note that, the BitMetric processor will be ready to write the data into the LLR processor every data 4η cycles. The node processors and the BitMetric processor will not stall even while switching between two modulation schemes as long as we have $4\eta_1 > n3 + 1 + 4log_2(\eta_2)$,

where η_1 and η_2 corresponds to the new old vector symbols respectively. Aforementioned condition will be satisfied if $\eta_1 > \eta_2$ for a reasonable value of n3 (such as 2 or 3). This can be ensured by reading the vector symbols from the memory in their increasing modulation order. The architecture has many

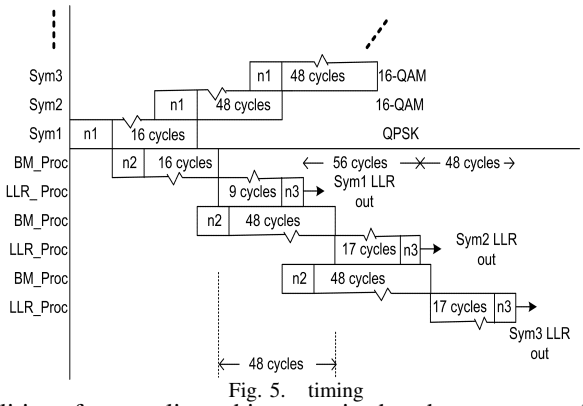

Fig. 5. timing

qualities of a systolic architecture, in that the processors have mostly local connections, has continuous flow even while switching between modulation schemes, and data is read from memory only at the start of the detection process for a received vector.

Recall that every vector symbol is composed of $4log_2\eta$ bits, and it takes 4η cycles to generate LLRs in steady state per vector symbol. Hence, the throughput is given by $(log_2(\eta)/\eta)f$, where f is the clock frequency.

Fig.5 shows the timing diagram of the detector. It is straightforward to see that the LLRs out of the LLR processor are independent of the number of pipeline stages n1, n2, n3. While switching between QPSK to 16-QAM it takes 56 cycles to compute LLRs for 16-QAM, this is because we are using a serial LLR processor. However, in steady state it takes 4η =48 cycles to generate LLRs for 16-QAM.

IV. Results and Discussion

We have made use of l^1 norm [10] to the avoid use of multipliers. Use of this norm leads to a loss of about 0.3-0.4dB w.r.t to the optimal l^2 norm (Fig.6). The SNR gains due to metric-recycling vary from 0.9dB in case of 64-QAM to about 1.3dB in case of QPSK.

An approximate estimate of hardware resources of the detector is shown in Table.I. Synthesis was done using Synopsys Design-Vision and mapped on 45nm Nangate tech. library. The input data (matrix data) is 11 bits fixed point (for both, real and imaginary parts) [5]. The internal precision is maintained, which leads to word-length of 22bits for the EDs. The maximum achievable clock frequency is limited by the loop in the cell in bitmetric manager (containing 22 bit cmpsel, M1), this loop delay comes to about 2.3ns. Hence, it can be clocked at about 434MHz. Assuming that the node processor array needs 55+55+77+99=286 bits every four cycles (recall that new matrix data is loaded every η cycles, and η=4 for QPSK) the memory bandwidth is upper bounded by about 9bytes/cycle. Assuming a cache of size:2048 bytes, line size:9 bytes, associativity:1, and no. of banks:1, we get access time of 0.3ns using the HP-CACTI tool [14] which is far less than 2.3ns. Hence, throughput of 217Mbps for QPSK, 108.5Mbps for 16-QAM, and 40.6Mbps for 64-QAM can be achieved . The clock frequency can be improved further by reducing the worldlength of the EDs by using "clipping" [11,13]. However, the impact of clipping on the PER needs to be carefully studied for various MCS and channel conditions. Moreover, overall throughput can be scaled simply by letting multiple detector cores operate on different carrier tones in a multi-carrier system such as MIMO-OFDM. Dynamic configurability is especially useful in an OFDMA system (the tones in OFDMA is shared between multiple users transmitting at different MCS), where the detector has to switch between different modes (and users) as quickly as possible to avoid switching latency.

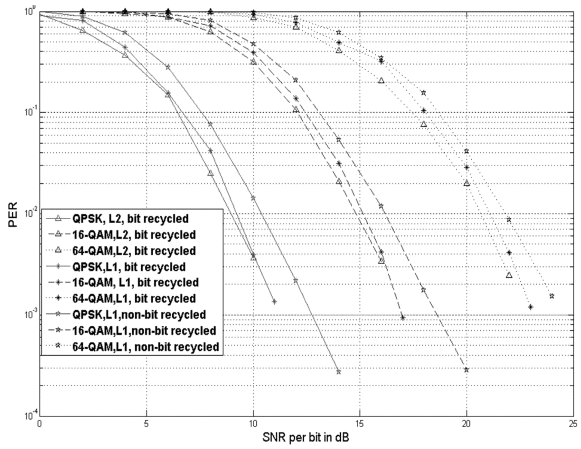

Fig. 6. PER Performance of LORD Algorithm

TABLE I
SYNTHESIS RESULTS

	Gate Count	Storage FFs
Node Procs	10.3KGE	572
BitMetric Proc	6.5KGE	584
LLR Proc	1.1KGE	574

V. Conclusion

A novel configurable MIMO detector architecture is presented in this paper. The detector is dynamically configurable for QPSK, 16-QAM and 64-QAM modulation schemes for a 4×4 MIMO system. The proposed architecture is highly suitable for the next generation wireless standards because of its flexibility, support for soft output and higher throughput.

References

[1] Hochwald,B. M., TenBrink,S., *"Achieving Near-Capacity on a Multiple-Antenna Channel"*, IEEE Trans. on Commun., 51:389399, Mar. 2003.

[2] Shariat-Yazdi, R.Kwasniewski, T., *"Challenges in the Design of Next Generation WLAN Terminals"*, Canadian Conference on Electrical and Computer Engineering(CCECE), pp. 1483-1486,April.2007.

[3] Bhagawat,P., Dash,R., Choi, G., *"Architecture for Reconfigurable MIMO detector and its FPGA Implementation"*, 15th IEEE International Conference on Electronics, Circuits, and Systems, ICECS 2008.

[4] Wang,H., et al., *" Managing dynamic reconfiguration on MIMO Decoder"*, Parallel and Distributed Processing Symposium,26-30 March 2007.

[5] Bhagawat,P., Dash,R., Choi, G., *"Dynamically Reconfigurable Soft Output MIMO Detector"*, XXVI IEEE Conference on Computer Design, ICCD, Oct.2008.

[6] Wu, D., Eilert, J., Liu, D.,*"Implementation of a High-Speed MIMO Soft-Output Symbol Detector for Software Defined Radio"*, Journal of Signal Processing Systems, May, 2009.

[7] Siti, M., Fitz, M.P., *"A Novel Soft-Output Layered Orthogonal Lattice Detector for Multiple Antenna Communications"*, IEEE International Conference Communications, 2006. ICC '06.

[8] Michalke,C., Zimmermann,E., Fettweis, G., *"Linear Mimo Receivers vs. Tree Search Detection: A Performance Comparison Overview"*, IEEE 17th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC),pp.1-7, Sept. 2006.

[9] Tomasoni, A.; Siti, M.; Ferrari, M.; Bellini, S., *"Turbo-LORD: A MAP-Approaching Soft-Input Soft-Output Detector for Iterative MIMO Receivers,"* ,Global Telecommunications Conference, 2007. GLOBECOM '07. IEEE , vol., no., pp.3504-3508, 26-30 Nov. 2007

[10] Burg, A.,et al., *"VLSI implementation of MIMO detection using the sphere decoding algorithm"*, IEEE Journal Solid State Circuits, vol.40, pp 1566-1577, July 2005.

[11] Studer, C.; Burg, A.; Bolcskei, H., *"Soft-output sphere decoding: algorithms and VLSI implementation,"*, Selected Areas in Communications, IEEE Journal on , vol.26, no.2, pp.290-300, February 2008.

[12] Wang, R., Giannakis, G., *"Approaching MIMO channel capacity with reduced-complexity soft sphere decoding,"*, in Proc. of IEEE Wireless Communications and Networking Conf. (WCNC), vol. 3, Mar. 2004, pp.16201625.

[13] Bhagawat, P., Dash, R., Gwan Choi, *"Systolic like soft-detection architecture for 4x4 64-QAM MIMO system,"*, Design, Automation and Test in Europe Conference and Exhibition, 2009. DATE '09.

[14] http://quid.hpl.hp.com:9081/cacti/