VLSI IMPLEMENTATION OF AN EFFECTIVE LATTICE REDUCTION ALGORITHM WITH FIXED-POINT CONSIDERATIONS

Brian Gestner, Wei Zhang, Xiaoli Ma and *David V. Anderson*

School of Electrical and Computer Engineering
Georgia Institute of Technology, Atlanta, GA 30332

ABSTRACT

Lattice reduction-aided equalization techniques have emerged as a low-complexity method to achieve the same diversity as maximum likelihood detectors. We address the VLSI implementation of these LR-aided equalizers by modifying the CLLL algorithm from a fixed-point hardware perspective. We then apply the modified algorithm together with additional micro-architecture and operation scheduling enhancements to create an updated CLLL processor. Finally, through BER simulations and FPGA synthesis results we demonstrate the suitability of our CLLL processor for integration into a 64-QAM MIMO system.

Index Terms— MIMO, Lattice Reduction, CORDIC

1. INTRODUCTION

Lattice reduction (LR) techniques have been incorporated into the equalization process to improve the performance. Currently, the Lenstra, Lenstra, Lovász, (LLL) algorithm [5] has been considered almost exclusively to perform LR and has been extended to the complex field to reduce the complexity further [2, 7]. Both numerical simulations [2] and theoretical proof [7] show that complex LLL (CLLL)-aided low-complexity equalizers achieve the same diversity as (near-) maximum likelihood equalizers (MLE) for i.i.d. channels. Compared to near-MLEs such as the sphere decoding (SD) method, the major advantages of LR-aided equalizers are the following: (i) Both the average complexity and complexity variance of LR-aided equalizers are much lower; (ii) The complexity of LR-aided equalizers does not depend on the SNR or constellation size but exclusively on the channel matrices, while the complexity of the SD method increases dramatically as SNR decreases and the constellation size increases; (iii) When the channel is invariant over several transmission blocks, e.g., slow-fading environment, LR-aided equalizers require the LR channel processing once, but the SD method requires a tree-search for each new received signal vector.

However, the iterative nature of the CLLL algorithm makes the complexity random and thus renders the economic real-time VLSI implementation challenging. Therefore, there are doubts if LR techniques are feasible in real-time communication systems. Currently, the only LR dedicated VLSI implementation reported in literature is the Brun's algorithm-based channel precoder described in [1]. Brun's algorithm achieves lower average complexity than the CLLL algorithm as the result of simplifications such as eigenvector approximations, which lead to reduced performance [8] (5dB loss on coding gain compared with CLLL).

In this paper we continue our work toward a resource-efficient, low-latency CLLL processor [3] by proposing, analyzing, and implementing fixed-point hardware-driven modifications to the CLLL algorithm.

2. CLLL-AIDED MIMO DECODING

Consider a V-BLAST multi-antenna system with N_t transmit-antennas and N_r receive-antennas. The data stream is divided into N_t sub-streams and transmitted through N_t antennas. Let $\boldsymbol{s} \in \mathcal{S}^{N_t}$ represent the $N_t \times 1$ transmitted symbol vector at one time slot, where $\mathcal{S}$ is the signal constellation, and $\boldsymbol{H}$ be the channel matrix that consists of $N_r \times N_t$ independent identically distributed (i.i.d.) complex Gaussian random variables with zero mean and unit variance. The $N_r \times 1$ received signal $\boldsymbol{y}$ can be expressed as $\boldsymbol{y} = \boldsymbol{H}\boldsymbol{s} + \boldsymbol{w}$, where $\boldsymbol{w}$ is the white Gaussian noise vector observed at the N_r receive-antennas.

The pseudo code of the CLLL algorithm is given in Table 1. Basically, the CLLL algorithm operates on the QR decomposition of matrix $\boldsymbol{H}$ to generate the reduced matrix $\tilde{\boldsymbol{H}} = \boldsymbol{H}\boldsymbol{T}$, where $\boldsymbol{T}$ is a unimodular matrix as shown in [3,6]. The QR decomposition $\tilde{\boldsymbol{H}} = \tilde{\boldsymbol{Q}}\tilde{\boldsymbol{R}}$ satisfies the following two conditions:

$$|\Re[\tilde{R}_{i,k}]| \leq \frac{1}{2}|\tilde{R}_{i,i}|, |\Im[\tilde{R}_{i,k}]| \leq \frac{1}{2}|\tilde{R}_{i,i}| \ , \ \forall\, i < k, \qquad (1)$$

$$\delta|\tilde{R}_{k-1,k-1}|^2 \leq |\tilde{R}_{k,k}|^2 + |\tilde{R}_{k-1,k}|^2 \ , \ \forall\, k \in [2, N_t], \qquad (2)$$

where the parameter δ is fixed as $\frac{3}{4}$ in this paper but can be arbitrarily chosen from $(\frac{1}{2}, 1)$ [7]. Eq. (1) is usually referred to as the size reduction condition, while Eq. (2) is called the δ condition. The CLLL algorithm functions by alternately forcing the size condition true (Lines 3-7 in Table 1) and the δ condition true (Lines 9-13) for progressively larger $k \times k$ upper-left sub-matrices of the $\tilde{\boldsymbol{R}}$ matrix and completing basis updates on the $\tilde{\boldsymbol{R}}$ (Line 11) and $\tilde{\boldsymbol{Q}}$ matrix (Line 12) accordingly.

Before modifying the CLLL algorithm from a fixed-point hardware perspective, we give the following two lemmas:

Lemma 1 *For the matrix $\boldsymbol{H}$ with i.i.d. complex Gaussian entries with zero mean and unit variance having QR decomposition $\boldsymbol{H} = \boldsymbol{Q}\boldsymbol{R}$, the magnitude of elements of $\boldsymbol{R}$ is bounded by B when saturation quantization is adopted. Specifically, when $\boldsymbol{H}$ is a 4×4 matrix, $B = 2^{2.84}$.*

Proof: Since each entry of $\boldsymbol{H}$ is complex Gaussian distributed with zero mean and unit variance, we know $2\|\boldsymbol{h}_n\|^2$ is Chi-square distributed with degrees of freedom $2N_r$, where $\boldsymbol{h}_n$ is the n^{th} column of $\boldsymbol{H}$. Thus, the upper bound B can be determined according to the target overflow probability, i.e., the probability of the column norm $\|\boldsymbol{h}_n\|$ exceeding a bound B corresponds to one overflow event every 190 years for an IEEE 802.11n system that requires the processing of 128 MIMO channel matrices every 4 μs [4]. Assuming that we adopt saturation quantization at the receiver, B safely upper bounds the elements of $\boldsymbol{R}$. Specifically, when $N_r = N_t = 4$, we can find $B = 2^{2.84}$. ■

Table 1. The Complex LLL Algorithm

(1) $[\tilde{\boldsymbol{Q}}, \tilde{\boldsymbol{R}}, \boldsymbol{T}]$ = sorted QR $(\boldsymbol{H})$; $\delta = \frac{3}{4}$; $k = 2$;
(2) while $k \leq m$
(3) for $n = k-1 : -1 : 1$
(4) $u = \text{round}(\tilde{R}_{n,k}/\tilde{R}_{n,n})$
(5) $\tilde{R}_{1:n,k} = \tilde{R}_{1:n,k} - u \cdot \tilde{R}_{1:n,n}$
(6) $T_{:,k} = T_{:,k} - u \cdot T_{:,n}$
(7) end
(8) if $\delta|\tilde{R}_{k-1,k-1}|^2 > |\tilde{R}_{k,k}|^2 + |\tilde{R}_{k-1,k}|^2$
(9) Swap $(k-1)^{\text{th}}$ and k^{th} columns in $\tilde{\boldsymbol{R}}$ and $\boldsymbol{T}$
(10) $\boldsymbol{\Theta} = \frac{1}{\|\tilde{R}_{k-1:k,k-1}\|}\begin{bmatrix}\tilde{R}^*_{k-1,k-1} & \tilde{R}_{k,k-1} \\ -\tilde{R}_{k,k-1} & \tilde{R}_{k-1,k-1}\end{bmatrix}$
(11) $\tilde{R}_{k-1:k,k-1:m} = \boldsymbol{\Theta}\tilde{R}_{k-1:k,k-1:m}$
(12) $\tilde{Q}_{:,k-1:k} = \tilde{Q}_{:,k-1:k}\boldsymbol{\Theta}^{\mathcal{H}}$
(13) $k = \max(k-1, 2)$;
(14) else
(15) $k = k+1$
(16) end
(17) end

With the following lemma, it is clear that B also upper bounds the diagonal elements of $\tilde{\boldsymbol{R}}$ during the CLLL processing. As a result, the real and imaginary parts of the off-diagonal entries of $\tilde{\boldsymbol{R}}$ are bounded by $\frac{1}{2}B$ after size reduction.

Lemma 2 *During the CLLL processing, the maximum magnitude of the diagonal elements of $\tilde{\boldsymbol{R}}$ does not increase.*

Proof: Notice that the CLLL algorithm only updates the diagonal elements of $\tilde{\boldsymbol{R}}$ when the δ condition (2) fails (Line 8), which means $|\tilde{R}_{k-1,k-1}|^2 > |\tilde{R}_{k,k}|^2 + |\tilde{R}_{k-1,k}|^2$. After basis updating, the squared magnitude of the $(k-1)^{\text{th}}$ and k^{th} diagonal elements are

$$|\tilde{R}_{k-1,k-1}|^2 \leftarrow |\tilde{R}_{k-1,k}|^2 + |\tilde{R}_{k,k}|^2,$$
$$|\tilde{R}_{k,k}|^2 \leftarrow \frac{|\tilde{R}_{k,k}|^2}{|\tilde{R}_{k-1,k}|^2 + |\tilde{R}_{k,k}|^2}|\tilde{R}_{k-1,k-1}|^2,$$

both of which are less than the original $|\tilde{R}_{k-1,k-1}|^2$. Thus, the maximum magnitude of the diagonal entries of $\tilde{\boldsymbol{R}}$ does not increase. ■

3. MODIFYING EFFECTIVE CLLL

The effective LLL algorithm was introduced in [6] for concatenation with a successive interference cancelation (SIC) detector. Essentially, only the first inner-loop (Lines 3-7 in Table 1) is executed such that at the end of the effective LLL algorithm the $\tilde{\boldsymbol{R}}$ matrix only satisfies the size condition in (1) for $i = k-1$. Although the effective LLL algorithm greatly reduces the CLLL algorithm complexity, the $\tilde{\boldsymbol{R}}$ entries that are not size-reduced are allowed to increase uncontrollably, which is unacceptable in a fixed-point hardware implementation. In this section, we propose to relax the size reduction condition and guarantee an upper-bound on the elements of matrix $\tilde{\boldsymbol{R}}$.

We first examine how the entries of matrix $\tilde{\boldsymbol{R}}$ can increase during a relaxed size reduction process. Define lines 3 to 7 in Table 1 as the inner-loop and lines 2 to 17 as the outer-loop. For the outer-loop on the k^{th} column, if we let $\tilde{R}'_{n,k}$ represent the intermediate value of $\tilde{R}_{n,k}$ after the first $(k-n-1)$ inner-loop iterations but before the size reduction on the n^{th} row, and $u_{l,k}$ denote the u value in line 4 at the $(k-l)^{\text{th}}$ inner-loop iteration, then

$$\tilde{R}'_{n,k} = \tilde{R}_{n,k} - \sum_{l=n+1}^{k-1} u_{l,k}\tilde{R}_{n,l} \tag{3}$$

The summation on the righthand side involves $\tilde{R}_{n,l}$, which are the results of size reduction operations in previous outer-loops (when the CLLL algorithm was operating on upper-left square matrices smaller than $k \times k$). We can relax the size condition in (1) on these elements by associating a $\phi_{n,l} \geq \frac{1}{2}$ with each $\tilde{R}_{n,l}$ and only requiring that

$$\left|\Re[\tilde{R}_{n,l}]\right|, \left|\Im[\tilde{R}_{n,l}]\right| < \phi_{n,l}\left|\tilde{R}_{n,n}\right|. \tag{4}$$

Noting that in the original CLLL algorithm, all the ϕ's are $\frac{1}{2}$. This relaxed size condition reduces the complexity of the size reduction operation because now each inner-loop need only be executed if the above condition is not true.

Therefore, we can upper bound the magnitude of the real component of $\tilde{R}'_{n,k}$ in (3) by

$$\left|\Re[\tilde{R}_{n,k}]\right| + \sum_{l=n+1}^{k-1}\left(|\Re[u_{l,k}]| + |\Im[u_{l,k}]|\right)\phi_{n,l}\left|\tilde{R}_{n,n}\right|. \tag{5}$$

We can also relax the size condition on the elements of the k^{th} column that are updated during the previous inner-loop iterations of the current outer-loop iteration to obtain

$$\left|\Re[\tilde{R}'_{l,k}] - \Re[u_{l,k}]\tilde{R}_{l,l}\right| < \phi_{l,k}\left|\tilde{R}_{l,l}\right| \tag{6}$$

By applying the upper bound B to the diagonal element $|\tilde{R}_{l,l}|$, we can rewrite (6) as

$$|\Re[u_{l,k}]|\left|\tilde{R}_{l,l}\right| < \phi_{l,k}B + \left|\Re[\tilde{R}'_{l,k}]\right| \tag{7}$$

Finally, as shown in [7, Appendix B], we have $\left|\tilde{R}_{n,n}\right| < (\delta - \frac{1}{2})^{\frac{n-l}{2}}\left|\tilde{R}_{l,l}\right|$. We substitute this result with $\delta = \frac{3}{4}$, and (7) (repeating for the imaginary parts) into (5) to obtain

$$\left|\Re[\tilde{R}'_{n,k}]\right| < \left|\Re[\tilde{R}_{n,k}]\right| + \sum_{l=n+1}^{k-1} 2^{l-n}\phi_{n,l}\left(2\phi_{l,k}B + \left|\Re[\tilde{R}'_{l,k}]\right| + \left|\Im[\tilde{R}'_{l,k}]\right|\right).$$

For a particular k outer-loop iteration, we can begin with this expression for the $n = k-1$ inner-loop iteration and recursively substitute. Assuming the ϕ's are symmetric with respect to the real and imaginary components and do not change during the process of the algorithm, we obtain

$$\left|\Re[\tilde{R}'_{n,k}]\right| < \gamma_{n,k}B + \left|\Re[\tilde{R}_{n,k}]\right| + \sum_{p=n+1}^{k-1}\alpha_{p,k}\left(\left|\Re[\tilde{R}_{p,k}]\right| + \left|\Im[\tilde{R}_{p,k}]\right|\right), \tag{8}$$

where the α's and γ's are functions of the ϕ's.

Note that at the end of an outer-loop for a particular k the $\left|\Re[\tilde{R}_{p,k}]\right|$ and $\left|\Im[\tilde{R}_{p,k}]\right|$ terms are upper bounded by $\phi_{p,k}B$ for $p \neq k-1$ due to the relaxed size condition, and by B for $p = k-1$ due to the δ condition. If we additionally enforce an absolute $\frac{1}{2}B$ upper bound on the $p \neq k-1$ elements (executing the appropriate inner-loop iteration as needed), then the maximum energy of the sub-vector consisting of the first to the $(k-1)^{\text{th}}$ elements of the k^{th} column is

$$\sum_{p=1}^{k-1}|\tilde{R}_{p,k}|^2 \leq B^2\left(1 + \frac{k-2}{2}\right) \tag{9}$$

This upper bound is significant because it is the maximum energy that could be re-distributed among the sub-vector elements due to subsequent basis updates (as δ conditions fail and the CLLL algorithm operates on smaller matrix sizes). To maximize the rightside in (8), we assume that subsequent basis updates distribute the energy among the sub-vector elements to maximize this upper bound. Solving this constrained maximum problem, we obtain

$$\left|\Re[\tilde{R}'_{n,k}]\right| < B\left(\gamma_{n,k} + \sqrt{k\left(\frac{1}{2} + \sum_{p=n+1}^{k-1} \alpha_{p,k}^2\right)}\right) \tag{10}$$

and reach a similar upper bound for the imaginary components. By designing hardware around these upper bounds, we can safely utilize variants of the effective CLLL algorithm in fixed-point implementations.

4. ARCHITECTURE IMPROVEMENTS

The CLLL processor in our initial work [3] consists of a main datapath that handles the size reduction and basis update operations and a secondary datapath that checks the δ condition and computes the $\boldsymbol{\Theta}$ matrix. To improve the area and throughput metrics of the CLLL processor, we alter the architecture to implement a modified effective CLLL algorithm, optimize the datapath width using the upper bounds in Section 3, and implement additional sub-module changes. Throughout our discussion we assume that the target system is the MIMO system described in Section 2.

4.1. Effective CLLL Scheme

To use a modified effective CLLL algorithm we must first decide what extent to weaken the size condition in our hardware implementation. As described in [6] to maintain the Bit-Error-Rate (BER) performance of LLL-aided equalizers, the size condition cannot be relaxed for the $\tilde{R}_{k-1,k}$ elements. Therefore $\phi_{k-1,k} = \frac{1}{2}$, and we must always execute the first iteration of the inner-loop. Simulations of the example 4×4 MIMO system from Section 2, however, reveal that by increasing all the other ϕ's from $\frac{1}{2}$ to $\frac{3}{2}$ and additionally forcing a size reduction operation when the $\frac{1}{2}B$ absolute upper bound is not satisfied, the average number of inner-loop iterations for $n < k-1$ is reduced by a factor of 4.

4.2. Main Datapath

We can implement the chosen effective CLLL scheme with only a small amount of additional hardware, and then reduce the required integer bits in the main datapath modules using the upper bounds derived in (10). The main datapath consists of an integer-rounded divider module, complex multiplier pipeline, and an accumulator/retire buffer to buffer the evolving $\tilde{R}_{1:n,k}$ matrix entries (Line 5). The divider in the CLLL processor consists of a single Newton-Raphson (NR) iteration-based reciprocation module and a multiplication pipeline. To compute the integer-rounded quotient in Line 4, the integer-rounded divider computes a reduced-precision reciprocal of the $\tilde{R}_{n,n}$ matrix element, multiplies this reciprocal by the $\tilde{R}_{n,k}$ complex element, and appropriately integer-rounds the real and imaginary parts of the quotient. The divider also contains logic to detect the $|\Re[u]|, |\Im[u]| = 0, 1$ cases and bypasses the multiplication pipeline appropriately. Part of this logic includes a comparator to detect $|\Re[\tilde{R}_{n,k}]|, |\Im[\tilde{R}_{n,k}]| < \frac{3}{2}|\tilde{R}_{n,n}|$, which is the same logic required to check the relaxed size condition of the chosen effective CLLL scheme. Therefore the only additional control overhead required to implement the chosen effective CLLL scheme is a comparator to check the absolute bound, $|\Re[\tilde{R}_{n,k}]|, |\Im[\tilde{R}_{n,k}]| < \frac{1}{2}B$.

We are now able to simplify various parts of the main datapath:

- Since the diagonal elements of the $\tilde{\boldsymbol{R}}$ matrix are upper bounded by $B = 2^{2.84}$, the NR-based reciprocation unit only requires 3 integer bits.
- Since the dividend ($\tilde{R}_{n,k}$) can increase during the size reduction process (Lines 3-7), it requires a larger number of integer bits. The dividends are the $\tilde{R}'_{n,k}$ elements defined in Section 3. Evaluation of the upper-bound in (10) reveal that given our chosen effective CLLL scheme, all the dividends are bounded above by $2^{7.70}$. Hence 8 dividend integer bits are sufficient.
- After the size reduction operation of a particular k outer-loop iteration, the off-diagonal elements are upper bounded by $\frac{1}{2}B$, but subsequent basis updates could increase the magnitude of these matrix elements. In the worst case the maximum energy of $2^{2(3.34)}$ ((9) evaluated for $k = 4$) could be redistributed to the real or imaginary component of a single matrix element. Therefore the $\tilde{\boldsymbol{R}}$ matrix storage requires 4 integer bits.

4.3. Secondary Datapath

The secondary datapath consists of a Householder CORDIC module that computes the square root of the righthand side of line 8 and then subsequently computes the parameters to form the $\boldsymbol{\Theta}$ matrix if a basis update is required. This is accomplished by grouping the righthand side components into a real vector $\boldsymbol{v} = \left[\tilde{R}_{k,k}, \Re[\tilde{R}_{k-1,k}], \Im[\tilde{R}_{k-1,k}]\right]^T$. Then a sequence of J Householder vectoring operations can be used to compute the norm of this 3D vector to a certain precision within a constant CORDIC gain factor, $C = \prod_{i=1}^{J} c_i$:

$$C\left(\|\boldsymbol{v}\| + \epsilon\right)\boldsymbol{e}_1 = \boldsymbol{A}_J \cdots \boldsymbol{A}_1 \boldsymbol{v}, \tag{11}$$

where multiplication by $\boldsymbol{A}_i$ can be implemented with shift and addition operations, $\boldsymbol{A}_i \boldsymbol{A}_i^T = c_i^2 \boldsymbol{I}$, $e_1 = [1, 0, 0]^T$, and ϵ is the error term introduced by the finite number of vectoring operations. In the hardware implementation, the righthand side in (11) is iteratively computed with a barrel shifter and adder tree, and then multiplication by $1/C$ compensates for the CORDIC gain factor. The $\boldsymbol{\Theta}$ matrix parameters are contained in $\boldsymbol{v}/\|\boldsymbol{v}\|$. These can be computed by applying the transpose of the $\boldsymbol{A}_i$ matrices in the opposite order to the vector $(1/C)\boldsymbol{e}_1$:

$$\boldsymbol{A}_1^T \cdots \boldsymbol{A}_J^T \left(\frac{1}{C}\boldsymbol{e}_1\right) = \frac{\boldsymbol{v}}{\|\boldsymbol{v}\| + \epsilon}, \tag{12}$$

Given this understanding, we can again apply the upper bounds derived earlier. The $\boldsymbol{v}$ vector consists of a diagonal element, which is upper-bounded by B and two off-diagonal elements, which after the size reduction operation, are upper bounded by $\frac{1}{2}B$. Therefore the righthand side in (11) is upper bounded by $D = C\left(B\sqrt{\frac{3}{2}} + \epsilon\right)$. For a reasonable choice of ϵ, we find that $D = 2^{3.95}$. Therefore the internal Householder CORDIC datapath requires 4 integer bits.

Given the low number of integer bits required, we consider an additional architecture change to accelerate the computation of the $\boldsymbol{\Theta}$ matrix. Due to the reversed order that the $\boldsymbol{A}_i$ matrices are applied in (12), the normalized $\boldsymbol{v}$ vector computation must begin after

Table 2. VLSI Implementation Results

	xc4vlx80-12	xc5vlx110-3
Real Multipliers	6/96	6/64
Gate Equivalents	79,308	64,693
Slices	3,322/35,840	1,369/17,280
Clock frequency	164 MHz	205 MHz
Average cycles per matrix	102	102

these matrices are determined. Therefore the $\boldsymbol{\Theta}$ matrix computation requires $2J$ cycles. Slight manipulation of (11) reveals that

$$\frac{v_i}{\|\boldsymbol{v}\| + \epsilon} = \boldsymbol{e}_1^T \boldsymbol{A}_J \cdots \boldsymbol{A}_1 \left(\frac{1}{C} \boldsymbol{e}_i \right), \tag{13}$$

where v_i is the i^{th} element of $\boldsymbol{v}$ and $\boldsymbol{e}_i$ is the standard Euclidean basis vector. This formulation suggests the utility of a Householder CORDIC architecture that has been unrolled by a factor of four such that a vectoring operation and three rotation operations can be computed simultaneously. In this architecture wire shifts reduce the complexity of the barrel shifters. Especially in an FPGA where multiplexing is resource-intensive, this hardware reduction partially offsets the quadrupled adder complexity. The net result is an architecture only requiring double the FPGA resources. In addition the unrolling allows more effective hardware re-timing, resulting in a reduced critical path. Given that this architecture requires only $J + 3$ cycles (compared to $2J$ cycles) to compute the $\boldsymbol{\Theta}$ matrix, we adopt this architecture for the updated CLLL processor.

5. RESULTS

To verify our integer bits choices and to determine the required number of fraction bits we conduct simulations of a 64-QAM 4×4 V-BLAST transmission with LR-aided successive interference cancelation (LRSIC) being employed at the receiver. We assume that the quantized channel state information and quantized received signal are passed through three modules to obtain the estimate of the transmitted symbol vector: a sorted QR decomposition (SQRD), a CLLL processer, and a SIC module. A fixed-point CLLL processor model mimics the exact behavior of the hardware implementation. The BER results shown in Figure 1 demonstrate that if we adopt a $[5.13]$ $\tilde{\boldsymbol{R}}$ matrix number representation, $[2.13]$ $\tilde{\boldsymbol{Q}}$ matrix number representation, and the $\boldsymbol{T}$ and u representations from [3], then the simulation results are nearly identical to a floating-point implementation. We also note that no overflow events occurred during processing of any of the 2 million channel instances, as expected.

We then realized the CLLL processor on an FPGA by using Synplify Pro for synthesis and Xilinx ISE 9.1 for P/R. These results are shown in Table 2. As a result of the reduced integer bits requirements, the number of multipliers is reduced from 10 [3] to 6 and the other area metrics are reduced as well. The updated CLLL processor on average can process over 2 million channel matrices / second. We expect at least a doubling of the clock frequency when this design is realized in a custom 65 nm silicon implementation.

The random cycle count nature of the CLLL processor is a consequence of the random complexity of the CLLL algorithm, which is also seen in sphere decoding algorithms. With additional buffering for the original $\boldsymbol{R}$ and $\boldsymbol{Q}$ matrices (before the lattice reduction), the CLLL processor could be integrated into a hybrid equalizer that uses the original $\boldsymbol{R}$ and $\boldsymbol{Q}$ matrices for the equalization process until the $\tilde{\boldsymbol{R}}$ and $\tilde{\boldsymbol{Q}}$ matrices are ready.

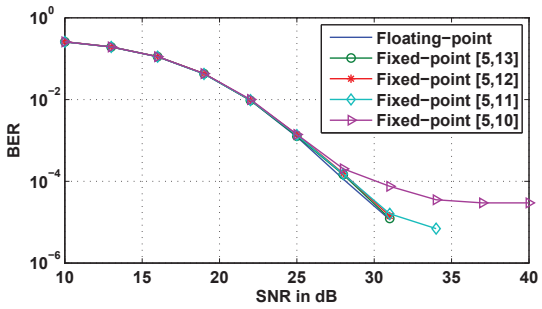

Fig. 1. 64-QAM, LR-aided Successive Interference Cancelation (SIC) equalizer fixed-point simulation results for a variety of $\tilde{\boldsymbol{R}}$ matrix bit precisions.

6. CONCLUSION

In this paper we rigorously analyzed the effect of relaxing the size condition in the CLLL algorithm, resulting in a modified Effective CLLL algorithm that reduces the complexity of the size reduction operation and requires negligible additional hardware resources. By applying this analysis to our original CLLL architecture with other optimizations, we were able to improve greatly the area and throughput metrics. Given the general treatment of the analysis, other modified effective CLLL schemes can be explored, and our analysis can be applied to arbitrary square antenna array sizes.

7. REFERENCES

[1] A. Burg, D. Seethaler, and G. Matz, "VLSI implementation of a lattice-reduction algorithm for multi-antenna broadcast precoding," *Proc. of IEEE International Symposium on Circuits and Systems*, May 27-30, 2007, pp. 673 - 676.

[2] Y. H. Gan and W. H. Mow, "Complex lattice reduction algorithms for low-complexity MIMO detection," *Proc. of IEEE Global Telecommunications Conf.*, vol. 5, St. Louis, USA, Nov. 28-Dec. 2, 2005, pp. 2953 – 2957.

[3] B. Gestner, W. Zhang, X. Ma, and D. V. Anderson,"VLSI Implementation of a Lattice Reduction Algorithm for Low-Complexity Equalization, *Proc. of IEEE International Conference on Circuits and Systems for Communications*, May 26-28, 2008.

[4] IEEE TGn Working Group, "Joint proposal: High throughput extension to the 802.11 Standard".

[5] A. K. Lenstra, H. W. Lenstra, and L. Lovász, "Factoring polynomials with rational coefficients," *Math. Ann*, vol. 261, pp. 515–534, 1982.

[6] C. Ling and N. Howgrave-Graham, "Effective LLL reduction for lattice decoding," in *Proc. IEEE International Symposium on Information Theory*, Nice, France, June 24-29, 2007, pp. 196-200.

[7] X. Ma and W. Zhang, "Performance analysis for MIMO systems with linear equalization," *IEEE Trans. Communications*, vol. 56, no. 2, pp. 309–318, Feb. 2008.

[8] D. Seethaler and G. Matz, "Efficient vector perturbation in multi-antenna multi-user systems based on approximate integer realtions," *Proc. of European Signal Proc. Conf.*, Florence, Italy, Sept. 4-8, 2006, pp. 673-676.