HYBRID LATTICE REDUCTION ALGORITHM AND ITS IMPLEMENTATION ON AN SDR BASEBAND PROCESSOR FOR LTE

Ubaid Ahmad, Min Li, Sofie Pollin, Amir Amin, Liesbet Van der Perre, Rudy Lauwereins

Interuniversity Micro-Electronics Center (IMEC) vzw
Kapeldreef-75, Leuven, B-3001, Belgium
email: {ubaid, limin, pollins, aminamir, vdperre, lauwerei}@imec.be

ABSTRACT

Lattice Reduction (LR) is a promising technique to improve the performance of linear MIMO detectors. This paper proposes a Hybrid LR algorithm (HLR), which is a scalable LR algorithm. HLR is specifically designed and optimised to exploit ILP and DLP features offered by parallel programmable baseband architectures. Abundant vector-parallelism in HLR is enabled with highly-regular and deterministic data-flow. Hence, HLR can be easily parallelized and efficiently mapped on Software Defined Radio (SDR) baseband architectures. HLR can be adapted to operate in two different modes to achieve the best performance/cycle trade-off, which is highly desirable for SDR baseband processing. The proposed algorithm has been evaluated in the context of 3GPP-LTE and implemented on ADRES which is a Coarse Grain Reconfigurable Array (CGRA) processor. Most of the previously reported implementations of LR algorithms are for ASIC or FPGA. However, to the best of author's knowledge, this is the first reported LR algorithm explicitly designed and optimized, to have a scalable and adaptive implementation for a CGRA processor like ADRES. The reported implementation of HLR can achieve gains of up to 12 dB compared to ZF for MIMO detection.

1. INTRODUCTION

The optimal solution to the MIMO detection problem is the Maximum Likelihood (ML) detector. A brute force ML detector implementation requires exhaustive search over all the possible transmitted symbols so its complexity increases exponentially with the number of antennas and the signal constellation. The challenge is to have MIMO detectors that can achieve performance comparable to the ML detector while having a lower complexity. Linear Multiple Input Multiple Output (MIMO) detectors, such as Zero Forcing (ZF) or Minimum Mean Square Error (MMSE), are attractive choices for MIMO detection due to their low computational cost. However, they cannot efficiently remove the inter-stream interference and suffer from noise amplification. Lattice Reduction (LR) has been proposed [1] to improve the performance of a sub-optimal detector for MIMO systems. Linear transformations on the MIMO channel matrix are performed to make it more orthogonal. As a result, for a given MIMO detector, the multiple received streams can be correctly detected with a higher probability. LR-based linear ZF/MMSE detectors have been proposed in [1] [2]. A well known technique to compute the reduced lattice basis is the LLL algorithm [3]. Complex LLL (CLLL) algorithm has been proposed [4] as a variant of LLL algorithm for MIMO processing.

In the context of implementation, majority of the existing LR algorithms are not designed for efficient mapping on parallel programmable architectures. Implementations of these algorithms reported for ASIC [5] and FPGA [6] [7] [8] are essentially sequential and non-deterministic. These algorithms are not suited for parallelism because of irregular data-flow, non-deterministic control flow, and extensive memory-shuffling. These drawbacks will result in very low resource-utilization of parallel programmable architectures. Besides this, they can not be adapted to achieve various performance complexity trade-offs, which is required to utilize the potential offered by an SDR. Majority of the exiting work on lattice reduction algorithms aim at improving performance while sacrificing computational complexity and vice versa. However, an adaptive LR algorithm is required for SDR.

The main contribution of this paper is a Hybrid LR (HLR) algorithm. In the proposed algorithm we introduce enhancements to our previously reported work [9] to make it adaptive for implementation on an SDR baseband processor so that performance/cycle trade-offs can be made in an efficient manner. The algorithm is designed such that computationally expensive parts of LR can be executed simultaneously. Hence, LR can be performed on a block of sub-carriers in parallel. This improves the processing throughput significantly while making the algorithm suitable for a parallel implementation. On the other hand, deterministic data-flow can account for DLP. In addition, HLR offers both design-time as well as run-time scalability. At the design time, HLR can be initialized for DLP offered by a parallel programmable architecture while run-time scalability provides performance/complexity trade-off [9]. The algorithm is optimised to have an implementation that can adapt between two different modes to provide the best possible performance while consuming minimum cycles. Thus, our algorithm can operate in different scalability modes in an adaptive fashion. For performance evaluation, HLR is implemented on ADRES [10] which is a CGRA processor for LTE baseband processing.

The remainder of this paper is organized as follows: remaining of this section describes the system model and LR-aided MIMO detection. The HLR algorithm is proposed in Section 2, while Section 3 details the implementation of our proposed algorithm on ADRES. Experimental results are reported in Section 4. Afterwards, conclusions are drawn in Section 5.

1.1 Lattice Reduction-aided MIMO Detection

Consider a spatially multiplexed MIMO system with M transmit and N receive antennas denoted as $M \times N$. The vec-

tor of received symbols $y \in \mathbb{C}^{N\times 1}$ is given as

$$y = \mathbf{H}x+n, \tag{1}$$

where $x \in \mathbb{C}^{M\times 1}$ denotes the vector of transmitted symbols taken independently from a Quadrature Amplitude Modulation (QAM) constellation with $\mathrm{E}\left[xx^H\right] = 1$, and $n \in \mathbb{C}^{N\times 1}$ is the vector of independent complex Gaussian noise samples where $n_i \sim N(0,\sigma^2)$ for $1 \le i \le M$. $\mathbf{H} \in \mathbb{C}^{N\times M}$ denotes the MIMO channel matrix and is considered to be perfectly known at the receiver.

$\mathbf{H}$ is assumed to be column full-rank then the columns $h_i, 1 \le i \le M$, of the channel matrix $\mathbf{H}$, can be seen as generator basis matrix for an M dimensional complex lattice $\mathbf{L}(\mathbf{H}) \in \mathbb{C}^{N\times 1}$. $\mathbb{C}^{N\times 1}$ consists of all integer linear combinations of the set of linearly independent basis column vectors, h_i i.e.

$$\mathbf{L}(\mathbf{H}) = \left\{ \mathbf{H}\alpha = \sum_{i=1}^{M} h_i\alpha_i. \mid \alpha_i \in \mathbb{C}\mathbb{Z} \text{ for } 1 \le i \le M \right\}$$

The possible set of basis vectors of lattice $\mathbf{L}$ can be infinite. The main idea behind LR is to obtain a reduced (i.e., orthogonal basis) generator basis $\mathbf{H}$ for the same lattice $\mathbf{L}$ in order to improve the performance of a linear equalizer [1]. $\mathbf{H}$ and $\widetilde{\mathbf{H}}$ generate the same lattice i.e. $\mathbf{L}(\mathbf{H}) = \mathbf{L}(\widetilde{\mathbf{H}})$, if $\widetilde{\mathbf{H}} = \mathbf{HT}$ where $\mathbf{T} \in \mathbb{C}^{M\times M}$ is a uni-modular matrix with $det(\mathbf{T}) = \pm 1$.

A well known approach to generate the transformation matrix $\mathbf{T}$ is the LLL algorithm [3]. The basis matrix $\mathbf{H}$ can be represented as a product of an $M \times N$ matrix $\mathbf{Q}$ and an $M \times M$ matrix $\mathbf{R}$ with $\mathbf{H} = \mathbf{QR}$, where $\mathbf{Q}$ is unitary and $\mathbf{R}$ is upper triangular matrix. With respect to the QR decomposition, h_k is almost orthogonal to the sub space spanned by $h_1,...,h_{k-1}$, if $\left|R_{1,k}\right|,...,\left|R_{k-1,k}\right|$ are small compared to $R_{k,k}$, where $2 \le k \le M$, [2].

Definition(Lenstra-Lenstra-Lovasz-Reduced) : A basis $\widetilde{\mathbf{H}}$ with QR decomposition $\widetilde{\mathbf{H}} = \widetilde{\mathbf{Q}}\widetilde{\mathbf{R}}$ is called LLL-reduced with parameter δ -see [2] [4], if

$$\left|\widetilde{\mathbf{R}}_{l,k}\right| \le \frac{1}{2}\left|\widetilde{\mathbf{R}}_{l,l}\right| \ \forall l \text{ where } 1 \le l < k \le M \tag{2}$$

and

$$\delta\left|\widetilde{\mathbf{R}}_{k-1,k-1}\right|^2 \le \left|\widetilde{\mathbf{R}}_{k,k}\right|^2 + \left|\widetilde{\mathbf{R}}_{k-1,k}\right|^2 \ \forall k \text{ where } 2 \le k \le M \tag{3}$$

2. HYBRID LATTICE REDUCTION ALGORITHM

OFDM in LTE converts a frequency selective channel into a set of independent and parallel flat fading channels. A significant improvement in processing throughput can be achieved if LR can be performed on a set of sub-carriers simultaneously. This requires a LR algorithm that has a predetermined run time and a deterministic execution path. A typical implementation of the CLLL algorithm suffers from the fact that the complexity and run time is determined by the position and number of column swaps. Although the CLLL algorithm can be terminated at intermediate iterations to bound the execution time, the pre-processing of each channel matrix would then still follow a different execution path for each iteration. As a result, a group of channel matrices cannot be reduced in parallel and an efficient implementation on architectures supporting vector (parallel) operations is not possible. Our proposed HLR algorithm, shown in Table 1, introduces necessary modifications in the CLLL algorithm while combining the best features offered by the well known fixed complexity LLL [11] and Effective LLL [12] algorithm, to overcome above mentioned shortcomings in a scalable parallel implementation.

2.1 Modified Effective LLL Algorithm

Effective LLL algorithm was introduced in [12]. A basis is called effective LLL reduced when,

$$\left|\widetilde{\mathbf{R}}_{l,k}\right| \le \frac{1}{2}\left|\widetilde{\mathbf{R}}_{l,l}\right| \forall k \text{ where } 2 \le k \le M, l = k-1 \tag{4}$$

and the *Lovasz* condition (3). In Effective LLL algorithm, a basis is size reduced against only the previous one before *Lovasz* condition checking, i.e. h_k is size reduced against h_{k-1} (lines (9-13)) Table 1. Size reducing basis h_k against $h_{k-2}...h_1$ for $k = 3,..,M$ is performed at the end of the algorithm (lines (45-51)). Moving of intermediate size reduction operations to the end of the algorithm significantly reduces the complexity compared to CLLL [12]. The final size reduction (lines (45-51)) does not contain any conditional statements so they can be very efficiently pipelined on a parallel programmable processor for better resource utilization.

The swap condition for CLLL, known as the *Lovasz* condition is given by (3). A direct implementation of the *Lovasz* condition requires significant computational resources. This can be avoided by using Siegel condition [6] instead of *Lovasz* condition. Dividing both sides of (3) by $R_{k-1,k-1}$ and using the fact (4) is already true before *Lovasz* test, (5) can be used for condition checking. The parameter $\delta = [(1/2,1]]$ in *Lovasz* test gives the parameter $\alpha = [2,4]$ in Siegel condition [13],

$$\left|\widetilde{\mathbf{R}}_{k-1,k-1}\right|^2 \le \alpha\left|\widetilde{\mathbf{R}}_{k,k}\right|^2 \ \forall k \text{ where } 2 \le k \le M \tag{5}$$

In the current implementation we set $\alpha = 2$. By doing this Siegel condition can be evaluated by simple bit shift and a comparison in fixed point implementation which reduces computational complexity. Since, there are significant number of column swaps in the first few iterations of CLLL, we modify HLR for SIMD architectures in such a way that column swaps and Givens rotations are always performed. Later, Siegel condition is evaluated (line (29)) Table 1, and if satisfied matrices $\mathbf{Q}^T$, $\mathbf{R}$, $\mathbf{T}$ are updated (lines (31-33)). This is required for parallel implementation of Givens rotations. By this modification LR can be performed on a block of adjacent sub-carriers in parallel. As described above, the CLLL algorithm has an un-deterministic data-flow caused by the column swaps. The number and sequence of column swaps depends on the input $\widetilde{R}$. To avoid this, HLR uses the approach proposed in [11]. A pre-determined sequence of k values (i.e. sequence of column swaps) is traversed (line(8-36)). As a result of this pre-determined fixed execution order, LR iterations can be performed on a number of sub-carriers simultaneously. The total number of iterations are bounded above by N_{max} (line(4)). This feature is required for implementation on a pipelined parallel hardware.

Table 1: Hybrid Lattice Reduction Algorithm

INPUT: $\mathbf{Q}^T$, $\mathbf{R}$, $\mathbf{T}$ OUTPUT: $\widetilde{\mathbf{Q}}^T, \widetilde{\mathbf{R}}, \widetilde{\mathbf{T}}$

1: Initialization: $\widetilde{\mathbf{Q}}^T := \mathbf{Q}^T, \widetilde{\mathbf{R}} := \mathbf{R}, \widetilde{\mathbf{T}} := \mathbf{T}$
2: Switch := Mode1 or Mode2
3: Niter := 0, exit := FALSE, exit := FALSE
4: **while** Niter $\leq$ Nmax AND exit = FALSE
5: **for** $i = 1$ **to** 8
6: flag := 0
7: $k := 2$
8: **while** $k \leq M$
9: $l = k-1$
10: $\mu = \lceil \tilde{R}_{l,k}/\tilde{R}_{l,l} \rfloor$
11: $\tilde{\mathbf{R}}(1:l,k) := \tilde{\mathbf{R}}(1:l,k) - \mu\tilde{\mathbf{R}}(1:l,l)$
12: $\mathbf{T}(:,k) := \mathbf{T}(:,k) - \mu\mathbf{T}(:,l)$
13: **if** Switch:=Mode1
14: **for** $l = k-2$ **to** 1 **step** -1
15: $\mu = \lceil \tilde{R}_{l,k}/\tilde{R}_{l,l} \rfloor$
16: $\tilde{\mathbf{R}}(1:l,k) := \tilde{\mathbf{R}}(1:l,k) - \mu\tilde{\mathbf{R}}(1:l,l)$
17: $\mathbf{T}(:,k) := \mathbf{T}(:,k) - \mu\mathbf{T}(:,l)$
18: **end**
19: **end**
20: $\widetilde{\mathbf{R}}_c(k-1:k,k-1) \leftarrow \widetilde{\mathbf{R}}(k-1:k,k)$
21: $\widetilde{\mathbf{R}}_c(k-1:k,k) \leftarrow \widetilde{\mathbf{R}}(k-1:k,k-1)$
22: $\widetilde{\mathbf{R}}_c(k-1:k,k+1:M) \leftarrow \widetilde{\mathbf{R}}(k-1:k,k+1:M)$
23: $\widetilde{\mathbf{T}}_c(:,k-1) \leftarrow \widetilde{\mathbf{T}}(:,k)$
24: $\widetilde{\mathbf{T}}_c(:,k) \leftarrow \widetilde{\mathbf{T}}(:,k-1)$
25: $\widetilde{\mathbf{Q}}_c^T(k-1:k,:) \leftarrow \widetilde{\mathbf{Q}}^T(k-1:k,:)$
26: $\mathbf{\Theta} = \begin{bmatrix} \overline{\alpha} & \overline{\beta} \\ -\beta & \alpha \end{bmatrix}$ with $\alpha = \frac{\tilde{R}_c(k-1,k-1)}{\left\|\tilde{R}_c(k-1:k,k-1)\right\|}$, $\beta = \frac{\tilde{R}_c(k,k-1)}{\left\|\tilde{R}_c(k-1:k,k-1)\right\|}$
27: $\widetilde{\mathbf{R}}_c(k-1:k,k-1:M) := \mathbf{\Theta}\widetilde{\mathbf{R}}_c(k-1:k,k-1:M)$
28: $\widetilde{\mathbf{Q}}_c^T(k-1:k,:) := \mathbf{\Theta}\widetilde{\mathbf{Q}}_c^T(k-1:k,:)$
29: **if** $\left|\widetilde{\mathbf{R}}(k-1,k-1)\right|^2 > 2\left|\widetilde{\mathbf{R}}(k,k)\right|^2$
30: flag := 1
31: $\widetilde{\mathbf{R}}(k-1:k,k-1:M) \leftarrow \widetilde{\mathbf{R}}_c(k-1:k,k-1:M)$
32: $\widetilde{\mathbf{T}}(:,k-1:k) \leftarrow \widetilde{\mathbf{T}}_c(:,k-1:k)$
33: $\widetilde{\mathbf{Q}}^T(k-1:k,:) \leftarrow \widetilde{\mathbf{Q}}_c^T(k-1:k,:)$
34: **end**
35: $k := k+1$
36: **end**
37: V$(1,i)$:= flag
38: **end**
39: Niter := Niter + 1
40: **if** sum(V$(1,:) \leq NLT$
41: exit := TRUE
42: **end**
43: **end**
44: **if** Switch:=Mode2
45: **for** $k = 3$ **to** M **step** 1
46: **for** $l = k-2$ **to** 1 **step** -1
47: $\mu = \lceil \tilde{R}_{l,k}/\tilde{R}_{l,l} \rfloor$
48: $\tilde{\mathbf{R}}(1:l,k) := \tilde{\mathbf{R}}(1:l,k) - \mu\tilde{\mathbf{R}}(1:l,l)$
49: $\mathbf{T}(:,k) := \mathbf{T}(:,k) - \mu\mathbf{T}(:,l)$
50: **end**
51: **end**
52: **end**

OUTPUT: $\mathbf{T}$ satisfying $\widetilde{\mathbf{H}} = \mathbf{HT}$

2.2 Adaptive Hybrid Approach

A combination of the above mentioned features, offered by Effective LLL [12] and the Fixed Complexity LLL [11], is required for an efficient implementation on a parallel baseband processor. To utilise the potential of SDR, HLR can adapt between these algorithms by the switch (line (13 and 44)) Table 1. Later on, we show, with experimental results, that operating in one Mode always cannot achieve the best performance/cycle trade-off. The performance requirements of an SDR are always changing so this feature provides run-time adaptability.

2.3 Run-Time Scalability

In OFDM, especially in LTE, adjacent sub-carriers are spaced very closely together and hence within the coherence bandwidth. Assume that the block size, or the number of channel matrices, to be processed in parallel is chosen to be small or within the coherence time. Then, these channel matrices are likely to have a similar execution time with a similar position and number of column swaps. Based on this, a simplified exit criterion i.e. Number of Lattice Thresholding(NLT) [9] is defined for a block of sub-carriers. When NLT is set to 4 (line (40)) Table 1 HLR terminates as soon as half of the $\widetilde{R}$ in a block gets LR reduced according to the criteria given by (4) and (5) before the bounded run-time N_{max} is reached. NLT value of 1 means that all the $\widetilde{R}$ gets reduced if an unbounded run-time is allowed.

3. IMPLEMENTATION ON ADRES

The presented work is based on the ADRES ASIP template [10]. As shown in Fig.1, the parameterizable template consists of an array of densely interconnected FUs that have local Register Files (RF) and configuration loop buffer. Using the ADRES template, we can design an ASIP with extensive parallelism by combining ILP and DLP. By changing the size of the array (number of FUs), we can tune the amount of supported ILP. By changing the number of SIMD slots in each FU, we can tune the amount of supported DLP. In our work, ADRES ASIP template is instantiated for 8-way SIMD and a block size of 8 sub-carriers is chosen for HLR to be lattice reduced simultaneously. Division and inverse square-root are computationally intensive operations and both are required in majority of LR algorithms. Division is present in the computation of $\mu = \lceil \tilde{R}_{l,k}/\tilde{R}_{l,l} \rfloor$ (line (10)). Division is followed by rounding so it can be replaced by comparisons. In the current implementation, we restrict the values of $\mu = [0,1,2,3,4]$, simulation results show that restricting the values of μ has negligible effect on the BER performance. μ can be calculated as,

$$\left|\tilde{\mathbf{R}}_{l,k})\right| \geq \beta \left|\tilde{\mathbf{R}}_{l,l}\right|, \text{ where } \beta = [0.5, 1.5, 2.5, 3.5]$$

$$\mu = \beta + 0.5$$

This eliminates the need of a divider unit and μ can be calculated by bit shifts, additions and a comparison. This has significant savings in terms of CPU cycles. By this approach, μ can be calculated for different sub-carriers in parallel. The inverse of square-root is required in the computation of the Givens Rotation (line (26)). A well known method for computing inverse of square root to arbitrary precision involves Newton-Raphson iterations. For example, to compute

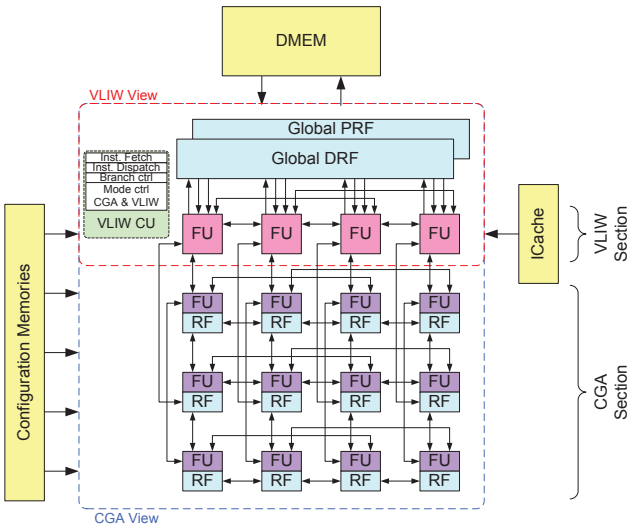

Figure 1: ADRES

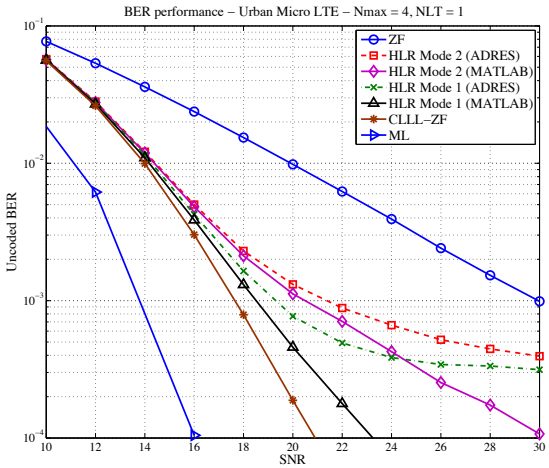

Figure 2: Uncoded BER of a 4x4 16-QAM MIMO system, Urban micro LTE channel at Speed = 3 km/h LR algorithms with ZF

$x = 1/sqrt(b)$ for $b > 0$. Initial estimate x_0 is calculated by linear approximation using a Look-up-table LUT, x_0 is then further refined iteratively using the following Newton Raphson equation:

$$x_{i+1} = 0.5x_i(3 - bx_i)$$

A 16 byte LUT for initial approximation and 2 iterations of the above equation are used for finding the inverse square-root up to a precision of 1 LSB in the current implementation.

4. EXPERIMENTAL RESULTS

In this section, we provide performance comparison of HLR to CLLL when used with ZF. A 4x4 MIMO system, with the LTE Urban micro channel at a user mobility of 3km/h is considered. Simulations are carried out for block size of 8 sub-carriers and the ADRES ASIP template is instantiated for 8 way SIMD. The effect of choosing the scalability parameters N_{max}, NLT and switching between Mode 1 and Mode 2 is also demonstrated.

4.1 BER Performance

In Fig.2 and Fig.3, BER performance of ZF using different LR algorithms is plotted. $N_{max} = 4$ and $N_{max} = 8$ is considered for HLR in Fig.2 and Fig.3, respectively. BER performance of both the MATLAB and fixed-point ADRES implementation of HLR are reported. In the current implementation 16-bit saturation quantization is used. The simulations show that, when HLR in Mode 2 is used with $N_{max} = 4$ and $NLT = 1$, gain from ZF is 8dB at a BER of 10^{-3} Fig.2. When HLR is switched from Mode 2 to Mode 1, the difference between CLLL and HLR decreases further. Fig.2 shows that the HLR is less than 2dB away from CLLL at a BER of 10^{-3} in Mode 1. As N_{max} is increased to 8 and $NLT = 1$, HLR in both Modes is less than 1dB away from CLLL, Fig.3, at a BER of 10^{-3}. The scalability parameters N_{max} and NLT both can be adjusted at run-time to achieve the required performance.

4.2 Implementation

The proposed implementation provides both design-time and run-time scalability. At design-time, any block size can be chosen depending on DLP offered by the available architecture and at run-time both N_{max} and NLT can be chosen to provide the desired performance/complexity tradeoff. Besides HLR can switch between two different Modes to provide the best performance/complexity trade-off. Fig. 4 shows the

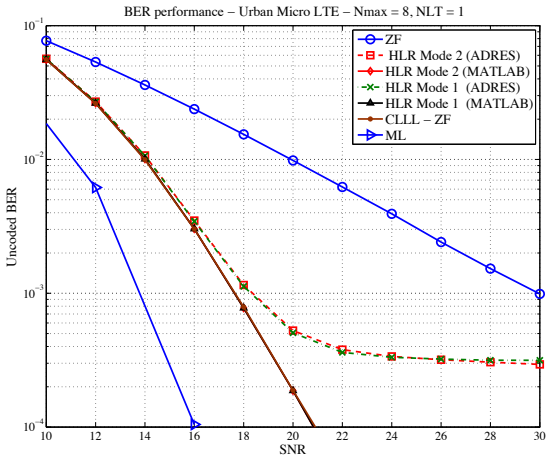

Figure 3: Uncoded BER of a 4x4 16-QAM MIMO system, Urban micro LTE channel at Speed = 3 km/h LR algorithms with ZF

average number of cycles per sub-carrier required by HLR (Mode1 and Mode 2) and the degradation in terms of BER from CLLL for different values of N_{max} and NLT. In Fig. 4, values of $N_{max} = [4,6,8]$ are shown by the three points on each individual curve (left to right). This clearly shows that the run-time adaptation to operate at various implementation points is required for efficient performance. The scalable parameters N_{max} ,NLT; and the adaptive switching

between Mode1 and Mode2 can be used for the best performance/complexity trade-off shown by optimal operation points in Fig. 4. Implementation results are summarized

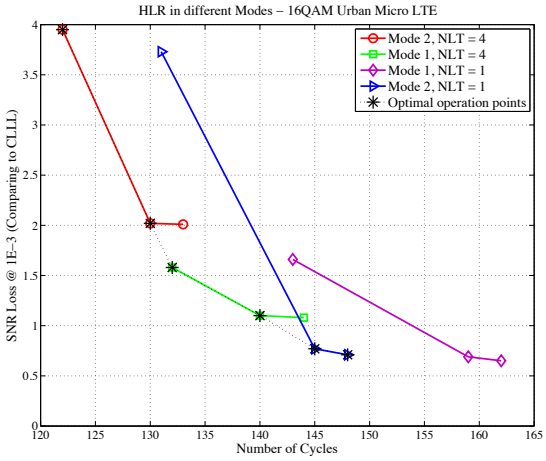

Figure 4: 4x4 16-QAM MIMO system, Performance - Cycles Tradeoff curves

in Table 2 in comparison with the previously reported designs [8] [6] [5] and our previous implementation [9]. Our LR implementation shows a clear advantage in terms of average clock cycles per LR when compared to the non-scalable FPGA [8] [6] as well as the scalable ASIC [5] implementations of LR algorithms, respectively.

	LLL [8]	CA [6]	I-SCNT-LR [5]	SBP-LR [9]	HLR
Platform	Virtex-4	Virtex-II Pro	0.65nm ASIC	ADRES	ADRES
Clock freq.(MHz)	140	100	400	600	600
Avg. Cycles per LR	130	420	713	104	130
Avg. Time per LR (μs)	0.93	4.20	1.78	0.17	0.21

Table 2: Comparison of HLR to different implementations

5. CONCLUSION

In this work, an algorithm architecture co-design approach is followed to propose a Hybrid LR (HLR) algorithm. Deterministic data-flow and predetermined execution time of HLR allows for its efficient parallelized mapping on any parallel programmable SDR baseband processor (ADRES in our implementation). Simulation results for 3GPP-LTE show that the HLR can be configured to operate in different modes to achieve efficient performance/cycles trade-off. HLR-ZF can provide SNR gains of up to 12dB when compared to linear-ZF. Additionally, improved execution times are attainable compared to the previously reported FPGA and ASIC implementations of lattice reduction algorithms by enabling parallel processing.

REFERENCES

[1] H. Yao and G. Wornell, "Lattice-reduction-aided detectors for mimo communication systems," in *Global Telecommunications Conference, 2002. GLOBECOM '02. IEEE*, vol. 1, 17-21 2002, pp. 424 – 428 vol.1.

[2] D. Wubben, R. Bohnke, V. Kuhn, and K.-D. Kammeyer, "Mmse-based lattice-reduction for near-ml detection of mimo systems," in *Smart Antennas, 2004. ITG Workshop on*, 18-19 2004, pp. 106 – 113.

[3] A. K. Lenstra, H. W. Lenstra, and J. L. Lovasz, "Factoring polynomials with rational coefficients," in *Math. Ann.*, vol. 261, 1982, p. 515534.

[4] Y. H. Gan, C. Ling, and W. H. Mow, "Complex lattice reduction algorithm for low-complexity full-diversity mimo detection," *Signal Processing, IEEE Transactions on*, vol. 57, no. 7, pp. 2701 –2710, july 2009.

[5] D. Wu, J. Eilert, and D. Liu, "A programmable lattice-reduction aided detector for mimo-ofdma," in *Circuits and Systems for Communications, 2008. ICCSC 2008. 4th IEEE International Conference on*, 26-28 2008, pp. 293 –297.

[6] L. Barbero, D. Milliner, T. Ratnarajah, J. Barry, and C. Cowan, "Rapid prototyping of clarkson's lattice reduction for mimo detection," in *Communications, 2009. ICC '09. IEEE International Conference on*, 14-18 2009, pp. 1 –5.

[7] B. Gestner, W. Zhang, X. Ma, and D. Anderson, "Vlsi implementation of an effective lattice reduction algorithm with fixed-point considerations," in *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, 19-24 2009, pp. 577 –580.

[8] ——, "Vlsi implementation of a lattice reduction algorithm for low-complexity equalization," in *Circuits and Systems for Communications, 2008. ICCSC 2008. 4th IEEE International Conference on*, 26-28 2008, pp. 643 –647.

[9] U. Ahmad, A. Amin, M. Li, S. Pollin, L. Van der Perre, and F. Catthoor, "Scalable block-based parallel lattice reduction algorithm for an sdr baseband processor," in *Communications, 2011. ICC '11. to appear in IEEE International Conference on*, 2011.

[10] B. Mei, A. Lambrechts, J.-Y. Mignolet, D. Verkest, and R. Lauwereins, "Architecture exploration for a reconfigurable architecture template," *Design Test of Computers, IEEE*, vol. 22, no. 2, pp. 90 – 101, mar. 2005.

[11] H. Vetter, V. Ponnampalam, M. Sandell, and P. Hoeher, "Fixed complexity lll algorithm," *Signal Processing, IEEE Transactions on*, vol. 57, no. 4, pp. 1634 – 1637, april 2009.

[12] C. Ling and N. Howgrave-Graham, "Effective lll reduction for lattice decoding," in *Information Theory, 2007. ISIT 2007. IEEE International Symposium on*, 2007, pp. 196 –200.

[13] B. Gestner, W. Zhang, X. Ma, and D. V. Anderson, "Lattice reduction for mimo detection: From theoretical analysis to hardware realization," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 2010.