This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE ICC 2009 proceedings

Rapid Prototyping of Clarkson's Lattice Reduction for MIMO Detection

Luis G. Barbero*, David L. Milliner†, T. Ratnarajah*, John R. Barry† and Colin Cowan*

*ECIT Institute
Queen's University, Belfast, UK
Email: l.barbero@ecit.qub.ac.uk

†School of Electrical and Computer Engineering
Georgia Institute of Technology, Atlanta, USA
Email: dlm@ece.gatech.edu

***Abstract*—This paper presents the field-programmable gate array (FPGA) implementation of a variant of the Lenstra-Lenstra-Lovász (LLL) lattice reduction (LR) algorithm, known as the Clarkson's Algorithm (CA), and its application to uncoded multiple input-multiple output (MIMO) detection. The CA provides practically the same performance as the LLL algorithm while having a considerably lower complexity, especially for MIMO systems with a large number of transmit and receive antennas. The algorithm has been implemented in real-time using a rapid prototyping methodology, greatly reducing its development time. Implementation results indicate that the variable complexity and the sequential nature of LR algorithms, like the CA, remain their main drawbacks from an implementation point of view.**

***Index Terms*—MIMO, lattice reduction, rapid prototyping.**

I. Introduction

Lattice reduction (LR) has been proposed in the context of uncoded detection of spatially-multiplexed multiple input-multiple output (MIMO) systems as a means of improving the performance of sub-optimal detectors [1]. This is achieved by preprocessing the MIMO channel by an LR algorithm, which transforms the channel into a *more orthogonal* equivalent one, and applying a linear or successive interference cancellation (SIC) detector to that new channel [2]. In particular, these LR-aided detectors have been shown to achieve the same diversity as the maximum likelihood detector (MLD) [3]. Although a number of LR methods exist in the literature with different levels of performance and complexity [4], [5], only recently have those algorithms been studied from an implementation point of view [6] - [8].

This paper presents a field-programmable gate array (FPGA) implementation of the Clarkson's Algorithm (CA) presented in [9], a variant of the popular Lenstra-Lenstra-Lovász (LLL) LR algorithm. The LLL algorithm approximates the optimal performance of the Korkine-Zolotareff (KZ) algorithm while having a polynomial average complexity in the dimension of the lattice in uncorrelated Rayleigh fading environments [10]. On the other hand, the CA provides practically the same performance but reduces the complexity of the LLL algorithm by modifying the reduction criterion and eliminating the intermediate size reduction steps [9]. Although the prototyping results of the CA do not match those of the optimized very large-scale integration (VLSI) implementation of the LLL algorithm in [8], the lower complexity of the CA makes it the most promising LR algorithm from a practical point of view.

A. Lattice Reduction-Aided Detection

We consider a spatially-multiplexed MIMO system with M transmit and N receive antennas, denoted as $M \times N$. The vector of received symbols $\mathbf{y} \in \mathbb{C}^{N\times 1}$ can be modelled as

$$\mathbf{y} = \mathbf{H}\mathbf{s} + \mathbf{v}\,, \tag{1}$$

where $\mathbf{s} \in \mathbb{C}^{M\times 1}$ denotes the vector of transmitted symbols taken independently from a quadrature amplitude modulation (QAM) constellation $\mathcal{O}$ of P points with $\mathrm{E}\left[|s_i|^2\right] = 1/M$, for $1 \leq i \leq M$, and where $\mathbf{v} \in \mathbb{C}^{N\times 1}$ is the vector of independent complex Gaussian noise samples $v_i \sim \mathcal{CN}(0, \sigma^2)$, for $1 \leq i \leq M$. The channel matrix $\mathbf{H} \in \mathbb{C}^{N\times M}$ has independent elements $h_{j,i} \sim \mathcal{CN}(0,1)$, for $1 \leq j \leq N$ and $1 \leq i \leq M$, representing a wireless propagation environment with uncorrelated Rayleigh fading. We assume that the channel is perfectly known at the receiver and that $N \geq M$.

The columns of the channel matrix $\mathbf{H}$ in (1), $\mathbf{h}_i$ for $1 \leq i \leq M$, can be seen as a generator basis of an M-dimensional complex lattice $\mathcal{L}(\mathbf{H}) \in \mathbb{C}^{N\times 1}$, where the lattice is defined as all complex integer combinations of the generator basis, i.e.

$$\mathcal{L}(\mathbf{H}) \triangleq \left\{ \mathbf{H}\mathbf{z} = \sum_{i=1}^{M} \mathbf{h}_i z_i \mid z_i \in \mathbb{C}\mathbb{Z} \text{ for } 1 \leq i \leq M \right\}.$$

The main idea behind LR-aided detectors is to obtain a reduced (i.e. *more orthogonal*) generator basis $\tilde{\mathbf{H}}$ for the same lattice $\mathcal{L}$ in order to improve the performance of sub-optimal detectors [1]. Two matrices $\mathbf{H}$ and $\tilde{\mathbf{H}}$ generate the same lattice, $\mathcal{L}(\mathbf{H}) = \mathcal{L}(\tilde{\mathbf{H}})$, if they can be written as $\tilde{\mathbf{H}} = \mathbf{H}\mathbf{T}$, where $\mathbf{T} \in \mathbb{C}\mathbb{Z}^{M\times M}$ is a unimodular matrix with determinant $\det(\mathbf{T}) = \pm 1$ [4]. The system model in (1) can then be rewritten as $\mathbf{y} = \tilde{\mathbf{H}}\mathbf{x} + \mathbf{v}$, where $\mathbf{x} = \mathbf{T}^{-1}\mathbf{s}$. Thus, sub-optimal detectors can be applied to this system model in order to obtain an estimate of $\mathbf{x}$, denoted $\hat{\mathbf{x}}$, before calculating an estimate of the transmitted vector $\mathbf{s}$, denoted $\hat{\mathbf{s}}$, using the relationship $\hat{\mathbf{s}} = \mathbf{T}\hat{\mathbf{x}}$. A detailed description of the operation of LR-aided detectors can be found in [11].

II. Clarkson's Algorithm

The CA has been proposed in [9] as a means of reducing the complexity of the LLL algorithm while providing a similar

978-1-4244-3435-0/09/$25.00 ©2009 IEEE

Algorithm 1: $(\tilde{\mathbf{Q}}, \tilde{\mathbf{R}}, \mathbf{T}) = \text{Clarkson}(\mathbf{Q}, \mathbf{R})$

1 $\tilde{\mathbf{Q}} = \mathbf{Q}$; $\tilde{\mathbf{R}} = \mathbf{R}$; $\mathbf{T} = \mathbf{I}_{M\times M}$;
2 $k = 2$;
3 **while** $k \leq M$
4 **if** $|\tilde{\mathbf{R}}(k-1,k-1)|^2 > 2|\tilde{\mathbf{R}}(k,k)|^2$
5 $\mu = \lceil \tilde{\mathbf{R}}(k-1,k)/\tilde{\mathbf{R}}(k-1,k-1) \rfloor$;
6 **if** $\mu \sim= 0$
7 $\tilde{\mathbf{R}}(1\!:\!k\!-\!1,k) = \tilde{\mathbf{R}}(1\!:\!k\!-\!1,k) - \mu\tilde{\mathbf{R}}(1\!:\!k\!-\!1,k\!-\!1)$;
8 $\mathbf{T}(:,k) = \mathbf{T}(:,k) - \mu\mathbf{T}(:,k-1)$;
9 **end**
10 Swap $k-1$th and kth columns of $\tilde{\mathbf{R}}$ and $\mathbf{T}$;
11 $\boldsymbol{\Theta} = \begin{matrix} \alpha^* & \beta \\ -\beta & \alpha \end{matrix}$ where $\alpha = \frac{\tilde{\mathbf{R}}(k-1,k-1)}{\|\tilde{\mathbf{R}}(k-1:k,k-1)\|}$; $\beta = \frac{\tilde{\mathbf{R}}(k,k-1)}{\|\tilde{\mathbf{R}}(k-1:k,k-1)\|}$;
12 $\tilde{\mathbf{R}}(k-1:k,k-1:M) = \boldsymbol{\Theta}\tilde{\mathbf{R}}(k-1:k,k-1:M)$;
13 $\tilde{\mathbf{Q}}(:,k-1:k) = \tilde{\mathbf{Q}}(:,k-1:k)\boldsymbol{\Theta}^{\mathrm{H}}$;
14 $k = \max(k-1,2)$;
15 **else**
16 $k = k+1$;
17 **end**
18 **end**
19 **for** $k = 2:M$
20 **for** $l = k-1:-1:1$
21 $\mu = \lceil \tilde{\mathbf{R}}(l,k)/\tilde{\mathbf{R}}(l,l) \rfloor$;
22 **if** $\mu \sim= 0$
23 $\tilde{\mathbf{R}}(1:l,k) = \tilde{\mathbf{R}}(1:l,k) - \mu\tilde{\mathbf{R}}(1:l,l)$;
24 $\mathbf{T}(:,k) = \mathbf{T}(:,k) - \mu\mathbf{T}(:,l)$;
25 **end**
26 **end**
27 **end**

performance. They both transform an input basis $\mathbf{H} = \mathbf{QR}$ into a *more orthogonal* reduced basis $\tilde{\mathbf{H}} = \tilde{\mathbf{Q}}\tilde{\mathbf{R}}$ with the only two algorithmic differences being [9]:

1) The use of the simpler Siegel reduction condition, i.e. $|\tilde{r}_{k-1,k-1}|^2 \leq 2|\tilde{r}_{k,k}|^2$, for $2 \leq k \leq M$, as opposed to the Lovász reduction condition of the LLL algorithm, i.e. $\delta|\tilde{r}_{k-1,k-1}|^2 \leq |\tilde{r}_{k,k}|^2 + |\tilde{r}_{k-1,k}|^2$, for $2 \leq k \leq M$ with e.g. $\delta = 3/4$.
2) The elimination of the intermediate size reduction operations that take place in the LLL algorithm. The size reduction operations in the CA are performed at the end of the algorithm, outside the main loop, except for a single size reduction operation performed before column exchange and Givens rotation.

The CA is reproduced in Algorithm 1, using MATLAB notation, where the two aforementioned differences can be observed. In order to compare their performances, Fig. 1 shows the cumulative distribution function (CDF) of the natural logarithm of the condition number of $\tilde{\mathbf{H}}$, defined as $\ln(\kappa(\tilde{\mathbf{H}})) = \ln(\|\tilde{\mathbf{H}}^{-1}\|\|\tilde{\mathbf{H}}\|)$ where $\|\cdot\|$ denotes the Euclidean norm operator, for each one of the algorithms. For a 4×4 system $\ln(\kappa(\tilde{\mathbf{H}}))$ is the same for both algorithms, while a very small degradation appears for the CA in an 8×8 system.

In addition, Table I shows the average number of times the more computationally intensive sections of the algorithms are executed in a 4×4 and an 8×8 system if the QR and

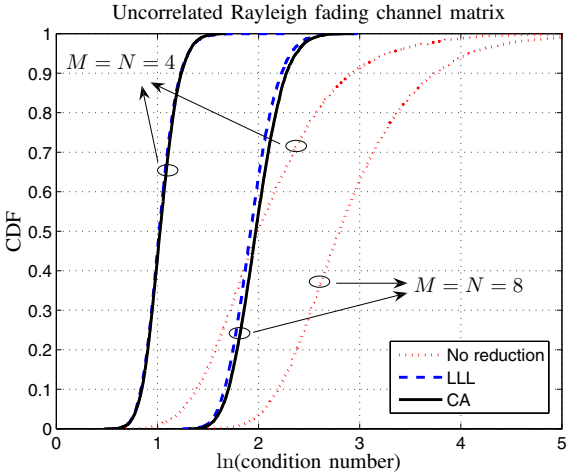

Fig. 1. CDF of $\ln(\kappa(\tilde{\mathbf{H}}))$ for different LR methods ($\delta = 3/4$ has been used for the LLL algorithm).

TABLE I
AVERAGE NUMBER OF EXECUTIONS OF THE DIFFERENT ALGORITHM SECTIONS.

$M = N = 4$		crit.	coef.	red.	rot.
QR	LLL	9.29	17.01	6.76	3.72
	CA	8.32	9.12	5.64	3.12
SQR	LLL	5.67	10.78	5.82	1.55
	CA	5.01	7.16	5.39	1.16
$M = N = 8$					
QR	LLL	30.36	118.11	29.29	12.13
	CA	22.97	36.22	19.06	8.22
SQR	LLL	14.28	59.55	23.57	3.70
	CA	11.66	30.35	19.41	2.35

the Sorted QR (SQR) decompositions are used. The following sections have been identified (with the corresponding lines in Algorithm 1):

- The reduction criterion check (crit.) in line 4.
- The computation of μ (coef.) in lines 5 and 21.
- The size reduction (red.) in lines 7, 8, 23 and 24.
- The Givens rotation (rot.) in lines 11-13.

The definition of these sections allows for the two algorithms to be easily compared from a hardware complexity point of view, even though each section has a different effect on the overall complexity of the algorithms[1]. All sections are present in both algorithms in exactly the same form (except for the reduction criterion check, which is simpler in the CA). Additionally, each section can be mapped to a specific hardware block so that the overall hardware complexity of both algorithms is almost equivalent[2]. Finally, the number of executions gives a rough estimate of the LR speed for each of the algorithms. Based on the above and looking at the results in Table I, it can be seen that the CA has a lower

[1]The crit. section is the least complex while the rot. section is the most.
[2]It should be noted that the red. and rot. sections depend on the value of k but, in hardware, the same blocks could be used irrespective of k.

complexity than the LLL algorithm in all cases, especially for the 8×8 system. This reduced complexity translates into a higher LR speed, which in turn leads to a more optimized LR implementation. This same trend has been observed when looking at the standard deviation of the number of executions (not shown). From Table I it can also be observed how the use of the SQR, which iteratively minimizes the diagonal elements of $\mathbf{R}$ as they are computed [11], can reduce the overall complexity of both algorithms (only the red. section of the CA in an 8×8 system increases marginally when the SQR is used).

III. FPGA Implementation

A. Platform and Methodology

An FPGA-based rapid prototyping system has been used for the implementation of the CA, providing the flexibility required to move quickly from a computer-based simulation to its hardware implementation. Development has been performed for Digilent's XUP-V2P board hosting a Xilinx Virtex-II-Pro FPGA (XC2VP30). The rapid prototyping methodology selected has been based on The Mathwork's MATLAB and Simulink and Xilinx's DSP System Generator. Initially, MATLAB is used to implement a complete MIMO system including transmitter, channel and receiver. The CA is then implemented on the FPGA using the DSP System Generator. The development of the FPGA model is embedded in a Simulink testbench that facilitates the debugging of the algorithm during the development stage, with the possibility of monitoring every signal in the FPGA model. The CA design is then synthesized for the FPGA using Xilinx's synthesis tools. The main advantage of this rapid prototyping platform and methodology is that it allows us to perform real-time hardware-in-the-loop testing of the algorithm without requiring any knowledge of hardware description languages (HDLs).

B. FPGA Architecture

The CA has been implemented for a 4×4 system in order to test its real-time suitability for LR-aided MIMO detection. Fig. 2 shows a simplified top-level block diagram of the CA FPGA architecture. The diagram contains all the main blocks of the design except the input and output memories used to synchronize the communication between the FPGA board and Simulink running on the host computer. The functions of the different blocks of the design are the following:

- Input and Output interfaces: adapt the numerical format and structure of the matrices between the CA and the input and output memories, synchronizing the communication between them.
- State Machine: controls the state transitions of the CA engine, generating the appropriate control signals for the rest of the blocks.
- Internal Memory: saves the intermediate values of the $\tilde{\mathbf{Q}}$, $\tilde{\mathbf{R}}$ and $\mathbf{T}$ matrices during the operation of the CA.
- Subset Selection: depending on the stage of the CA (determined by the values of k and l in Algorihm 1), this blocks selects the subsets of $\tilde{\mathbf{Q}}$, $\tilde{\mathbf{R}}$ and $\mathbf{T}$ required for the current iteration. In particular, during the main loop, the following subsets are selected: $\tilde{q}_{:,k-1:k}$, $\tilde{r}_{1:k,k-1:k}$, $\tilde{r}_{k-1:k,k+1:M}$ and $t_{1:k,k-1:k}$. During the final size reduction iterations, the following subsets are selected: $\tilde{r}_{1:l,k}$, $\tilde{r}_{1:l,l}$, $t_{:,l}$ and $t_{:,k}$.
- Reduction Criterion: checks the Siegel reduction condition to determine whether size reduction and Givens rotation are required. This block corresponds to the crit. algorithm section in Section II.
- Size Reduction: computes the coefficient μ and performs the size reduction operations if $\mu \neq 0$. This section combines the coef. and red. algorithm sections.
- Givens Rotation: performs the Givens rotation on $\tilde{\mathbf{Q}}$ and $\tilde{\mathbf{R}}$ together with the column exchange of $\tilde{\mathbf{R}}$ and $\mathbf{T}$. It corresponds to the rot. algorithm section.
- Subset Mux: this multiplexer, depending on the stage of the algorithm and the result of the reduction criterion, selects the appropriate subsets to be integrated back into the full matrices.
- Subset Merge: merges the modified subsets of $\tilde{\mathbf{Q}}$, $\tilde{\mathbf{R}}$ and $\mathbf{T}$ back into the full matrices to be stored by the Internal Memory block.

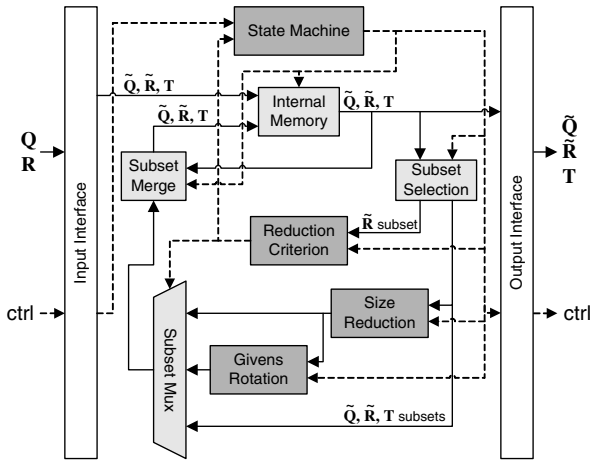

Fig. 2. Simplified top-level block diagram of the CA architecture with solid lines indicating data paths and dashed lines indicating control paths.

In general, the white and light-grey blocks in Fig. 2 play a fundamental storage role, making extensive use of the FPGA flip-flops (FFs). On the other hand, the dark-grey blocks are more computationally intensive. The only exception is the State Machine block, whose complexity comes from a functional point of view, to generate the valid control signals depending on the algorithm state, rather than from a hardware resources point of view (most of the signals in that block are booleans or small integers). In addition, the complexity of the Reduction Criterion block is lower than that of the same block in a LLL implementation, thanks to the simpler Siegel reduction condition.

Thus, the most computationally intensive blocks are the Size reduction and the Givens rotation ones. Even though the main

aim of this work was to obtain a proof of concept implementation of the CA using a rapid prototyping methodolgy, a number of measures have been taken to reduce the complexity of those two blocks, without compromising the development methodology. In particular, it can be seen that both blocks require divider architectures. In order to reduce the effect the dividers would have in the overall resource use and the latency of the design, only one divider has been used in the entire design, being reused by both blocks. In addition, the division operations have been transformed into multiplications by the inverse, to reduce the number of divisions performed. On the other hand, an off-the-shelf Xilinx divider block has been used to reduce the development time. The single square root operation required in the Givens Rotation block is performed also by an off-the-shelf block.

Another critical aspect of the Size Reduction and Givens Rotation blocks is the reuse of multipliers in order to reduce the complexity of the design. In the Size Reduction block, one complex multiplier is used for all the size reduction operations on $\tilde{\mathbf{R}}$ while another complex multiplier is used for the operations on $\mathbf{T}$. This setup represents a trade-off between the use of multipliers and the latency of the design[3]. In the Givens Rotation block, two complex multipliers and two real-complex multipliers are running in parallel to perform the Givens Rotation operations on both $\tilde{\mathbf{Q}}$ and $\tilde{\mathbf{R}}$ sequentially. Due to the different fixed-point precision used for the elements of $\tilde{\mathbf{Q}}$ and $\tilde{\mathbf{R}}$, the values at the input/output of the multipliers had to be converted to/from integer values with no fractional bits so that the same multipliers could be used for both matrices. Overall, the steps described above have helped reduce the resource use of the CA implementation while exclusively using the graphical interface provided by the DSP System Generator.

IV. Results

This section shows the FPGA implementation results of the CA in terms of hardware resource use, bit error rate (BER) performance of CA-aided detectors and LR speed. The resource use of the CA implementation is summarized in Table II. Initially, it can be observed how the multipliers are the least used resource due to the multiplier reuse described in the previous section. On the other hand, the slices are the most used resource, although this result can be modified by changing the settings of the place and route tools. Each slice contains two FFs and two look-up tables (LUTs) and the use of those resources is below 40%, indicating that a considerable percentage of slices are only partially used. The random access memory (RAM) blocks are used mostly by the input and output buffers required to perform a number of LRs in every hardware co-simulation run. The results in Table II have been obtained considering a wordlength of 14 bits for the elements of $\tilde{\mathbf{Q}}$ and $\tilde{\mathbf{R}}$ and 9 bits for the elements of $\mathbf{T}$.

Fig. 3 shows the BER performance of CA-aided MIMO detectors in a 4×4 system using 4-, 16- and 64-QAM modulations as a function of the signal to noise ratio (SNR) per bit, $E_b/N_0 = \log_2^{-1}(P)/\sigma^2$. The FPGA results have been obtained by running the MIMO system in MATLAB and performing the LRs in real-time on the prototyping platform. The fixed-point precision of the design has been adjusted so that the FPGA performance practically matches the floating-point MATLAB performance if 4-QAM modulation is used. This resulted in the following fixed-point formats: $\tilde{\mathbf{Q}}$(1.13), $\tilde{\mathbf{R}}$(5.9) and $\mathbf{T}$(9.0), where the first and second numbers indicate the number of integer and fractional bits, respectively. Lastly, both a zero-forcing (ZF) linear detector and a SIC detector have been simulated. It can be seen how the detectors using the FPGA CA match the performance of those using a MATLAB CA in the 4-QAM case. However, a performance degradation appears as we increase the modulation order, an aspect often not reported in the literature. This is due to the constellation points getting closer to each other as the modulation order increases.

In order to measure the LR speed of the CA, Fig. 4 shows the CDF of the number of cycles required per CA LR using both the QR and the SQR decompositions. As shown in Section II, the use of the SQR reduces the number of cycles required per LR compared to using the QR, with a 37% reduction in the average number of cycles. Similar reductions have been reported for the LLL algorithm in [8]. The number of cycles has been obtained targeting the implementation to match the 100MHz clock frequency limitation of the prototyping board. Different clock frequency/number of cycles trade-offs could be achieved by adjusting the internal pipeline stages

[3]The two size reduction operations can be performed in parallel, reducing the overall latency of the block.

TABLE II
FPGA resource use of the CA in a 4×4 system.

XC2VP30	Use	Percentage
Number of slices	7,349 / 13,696	53%
Number of FFs	9,051 / 27,392	33%
Number of 4-input LUTs	10,254 / 27,392	37%
Number of 18×18 multipliers	24 / 136	17%
Number of 18Kb RAM blocks	69 / 136	51%

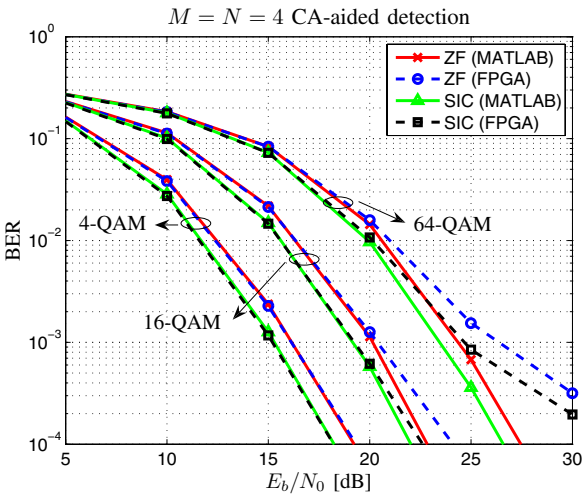

Fig. 3. BER performance of CA-aided detectors as a function of the SNR per bit in a 4×4 system.

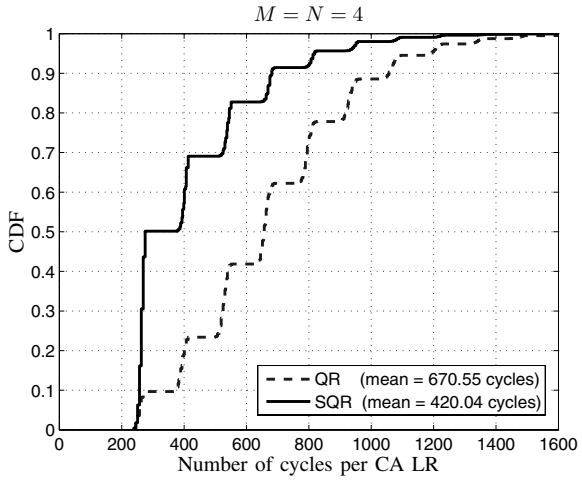

Fig. 4. CDF of the number of cycles per CA LR using both QR and SQR decompositions.

TABLE III
COMPARISON OF REAL-TIME IMPLEMENTATIONS OF THE CA AND THE LLL ALGORITHM IN A 4×4 SYSTEM.

	LLL [8]	CA
Plaftorm	Virtex-4	Virtex-II Pro
Number of slices	3,617	7,349
Number of multipliers	10	24
Clock freq. (MHz)	140	100
Average cycles per LR	130	420
Average time per LR (μs)	0.93	4.20

of the blocks in the design in order to find the optimal trade-off point. However, those designs could not be tested in real-time on the hardware platform.

Finally the CA performance is compared to that of the LLL presented in [8] in Table III. The LLL implementation is better than that of the CA due primarily to the following two reasons. Firstly, the LLL has been implemented using an HDL allowing a greater flexibility but a longer development time compared to the rapid prototyping methodology used for the CA. Secondly, a number of optimizations have been implemented for the square root and division operations in [8], as opposed to the off-the-shelf blocks used here. This has an effect on both the slice use and the latency of the design. In particular, the latency of the Size Reduction block is of 37 cycles, with 29 of them used for the division operation while the latency of the Givens rotation block is of 88 cycles, with 73 cycles devoted to the square root and division operations. The number of multipliers of the CA could also be reduced to match that of the LLL by reusing the multipliers between the Size Reduction and the Givens Rotation blocks. It should also be noted that the CA has been implemented on a comparatively older hardware platform which can have a non-negligible effect on the implementation results. In any case, the complexity results of the CA indicate that its implementation using the methodology and platform of [8] would improve the results of the LLL algorithm, making the CA a promising LR algorithm for MIMO detection.

V. CONCLUSION AND FUTURE WORK

This paper has presented an FPGA implementation of the CA applied to LR-aided detectors using a rapid prototyping approach. It has been shown how the CA should be considered as a low-complexity alternative to existing LR methods. In particular, we believe that a VLSI implementation of the CA would provide a better performance than the existing LLL one. Finally, the optimization of the square root and division operations in the CA and the prototyping of other LR algorithms, like the Seysen's Algorithm [5], are the main subjects of ongoing work.

ACKNOWLEDGEMENTS

The authors would like to thank Dominik Seethaler, Christoph Studer and Andreas Burg for many insightful discussions. The work of Luis G. Barbero was supported by the UK Engineering and Physical Sciences Research Council under grant number EP/G026092/1. The work of David L. Milliner and John R. Barry was supported by Texas Instruments, Inc.

REFERENCES

[1] H. Yao and G. W. Wornell, "Lattice-reduction-aided detectors for MIMO communication systems," in *Proc. IEEE Global Telecommunications Conference (GLOBECOM '02)*, vol. 1, Taipei, Taiwan, Nov. 2002, pp. 424–428.

[2] C. Windpassinger and R. F. H. Fischer, "Low-complexity near-maximum-likelihood detection and precoding for MIMO systems using lattice reduction," in *Proc. IEEE Information Theory Workshop (ITW '03)*, vol. 1, Paris, France, Apr. 2003, pp. 345–348.

[3] M. Taherzadeh, A. Mobasher, and A. K. Khandani, "LLL lattice-basis reduction achieves the maximum diversity in MIMO systems," in *Proc. IEEE Iternational Symposium on Information Theory (ISIT '05)*, vol. 1, Adelaide, Australia, Sep. 2005, pp. 1300–1304.

[4] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger, "Closest point search in lattices," *IEEE Trans. Inf. Theory*, vol. 48, no. 8, pp. 2201–2214, Aug. 2002.

[5] D. Seethaler, G. Matz, and F. Hlawatsch, "Low-complexity MIMO data detection using Seysen's lattice reduction algorithm," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '07)*, vol. 3, Honolulu, HI, USA, Apr. 2007, pp. 53–56.

[6] A. Burg, D. Seethaler, and G. Matz, "VLSI implementation of a lattice-reduction algorithm for multi-antenna broadcast precoding," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS '07)*, vol. 1, New Orleans, LA, USA, May 2007, pp. 673–676.

[7] D. Wu, J. Eilert, and D. Liu, "A programmable lattice-reduction aided detector for MIMO-OFDMA," in *Proc. IEEE International Conference on Circuits and Systems for Communications (ICCSC '08)*, vol. 1, Shanghai, China, May 2008, pp. 293–297.

[8] B. Gestner, W. Zhang, X. Ma, and D. V. Anderson, "VLSI implementation of a lattice reduction algorithm for low-complexity equalization," in *Proc. IEEE International Conference on Circuits and Systems for Communications (ICCSC '08)*, vol. 1, Shanghai, China, May 2008, pp. 643–647.

[9] I. V. L. Clarkson, *Approximation of Linear Forms by Lattice Points with Applications to Signal Processing*. Australian National Univesity PhD Thesis, Jan. 1997.

[10] J. Jaldén, D. Seethaler, and G. Matz, "Worst- and average-case complexity of LLL lattice reduction in MIMO wireless systems," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '08)*, vol. 1, Las Vegas, NV, USA, Apr. 2008, pp. 2685–2688.

[11] D. Wübben, R. Böhnke, V. Kühn, and K.-D. Kammeyer, "MMSE-based lattice-reduction for near-ML detection of MIMO systems," in *Proc. ITG Workshop on Smart Antennas*, vol. 1, Munich, Germany, Mar. 2004, pp. 106–113.