Decoding the Golden Code: A VLSI Design

Barbara Cerato, Guido Masera, and Emanuele Viterbo

***Abstract*—The recently proposed Golden code is an optimal space-time block code for 2×2 multiple-input–multiple-output (MIMO) systems. The aim of this work is the design of a VLSI decoder for a MIMO system coded with the Golden code. The architecture is based on a rearrangement of the sphere decoding algorithm that achieves maximum-likelihood (ML) decoding performance. Compared to other approaches, the proposed solution exhibits an inherent flexibility in terms of QAM modulation size and this makes our architecture particularly suitable for adaptive modulation schemes. Relying on the flexibility of this approach two different architectures are proposed: a parametric one able to achieve high decoding throughputs ($>$ 165 Mb/s) while keeping low overall decoder complexity (45 KGates), a flexible implementation able to dynamically adapt to the modulation scheme (4-,16-,64-QAM) retaining the low complexity and high throughput features.**

***Index Terms*—Digital architectures, Golden code, multiple-input–multiple-output (MIMO), sphere decoding, VLSI.**

I. Introduction

The hardware implementation of high data rate and high reliability wireless communication systems has raised engineering and research challenges for many years. Higher transmission reliability demands for higher levels of processing complexity in the mobile terminal, while faster data rates require increased throughput: both evolutionary trends are strong driving forces for the search of novel efficient architectures implementing the most critical baseband processing functions. In particular new standards proposed to regulate Wireless Local Area Networks (WLAN) and Metropolitan Area Networks (MAN) are significant examples of very challenging applications from the implementation point of view.

There are two main objectives on which research is actually focused. The first goal is to make wireless communication data rate comparable to that of wired communications, the second one is to improve reliability, by combating multipath, noise and interference effects. The recourse to multiple-input–multiple-output (MIMO) systems seems to be one of the most promising solutions to reach both these results. In particular new space-time codes over MIMO channels are now able to reach the best tradeoff between data rate and diversity gain, although they require more sophisticated detection schemes at the receiver [1]–[3].

The main contribution of this work is in the hardware design of a decoder for this kind of codes, in particular for the decoding of a 2×2 MIMO signal coded with the *Golden code* [3]. Golden code is a recently proposed full-rate and full-diversity space-time block code, chosen for its good energy efficiency. The maximum-likelihood (ML) decoding algorithm for the Golden code is based on the *sphere decoder* [4], which has already been widely addressed in the literature also from a hardware implementation point of view [5]–[7].

Several architectures have been proposed for the efficient implementation of the sphere decoding architecture. In [6, ASIC-I], in order to reach high throughput dedicated multipliers and parallel computations are used adopting a so called "one node per cycle" architecture. Other architectures instead take advantage of suboptimal algorithms: good examples of this approach are given in [6, ASIC-II], where the L_∞-norm is implemented as an alternative to the more expensive L_2-norm, and in [7], where the K-Best algorithm allows for performance-complexity trade-offs. These choices lead to fully optimized architectures, achieving high throughput; however, they are not ML (the loss is about 1.4 dB in the case of [6, ASIC-II]), and most of all the cited implementatios have been proposed for specific modulation and transmission schemes and do not have reconfigurability features, although in [6] the possibility to adapt the proposed solution to different modulations is also mentioned. In this work, we overcome these limitations, proposing two novel architectures designed with VHDL as a reusable intellectual property (IP) macrocell: the first one is parametrized with respect to the fixed-point representation of data and to the addressed modulation scheme. The second architecture is flexible, meaning that it can be dynamically configured to cover multiple modulation schemes. We note that both these hardware implementations can be equivalently used in a 4×4 uncoded MIMO system [8].

Section II is dedicated to reviewing the sphere decoding algorithm. The detailed descriptions of the two hardware implementations are then carried out in III and IV. In the last two sections, results and conclusions are presented.

II. Sphere Decoding Algorithm

The system model when Golden code has been used [9] can be expressed as

$$\boldsymbol{y} = \boldsymbol{H}x + \boldsymbol{z} = \boldsymbol{H}Gs + \boldsymbol{z} \tag{1}$$

where $\boldsymbol{y}$ is the 8×1 received real vector and $\boldsymbol{z}$ is a 8-D independent and identically distributed (i.i.d.) zero mean Gaussian noise real vector.

Decoding the Golden code is equivalent to decoding an 8-D lattice with generator matrix $\boldsymbol{M} = \boldsymbol{H}G$. Provided that $\boldsymbol{H}$ is perfectly known at the receiver, the optimal detector for a MIMO channel, is the maximum-likelihood (ML) detector. It solves the following equation:

$$\hat{\boldsymbol{s}} = \arg \min_{\boldsymbol{s} \in Q^n} \|\boldsymbol{y} - \boldsymbol{M}s\|^2 \tag{2}$$

where Q^n is the cardinality of the search space and $n = 8$.

Sphere decoding algorithms denote a family of algorithms, which aim at lowering the complexity of the minimization (2) by analyzing only a subset of the solution space [10]. They look at the set of possible solutions as points of a lattice and try to find the closest point to the received vector. In particular, a hypersphere is constructed around the received vector and only points inside it are taken into account, since the others are actually too far. This constraint can be written as

$$\|\boldsymbol{y} - \boldsymbol{M}s\|^2 \leq C_0 \tag{3}$$

where C_0 is the square radius of the hypersphere [4], [11].

1) Tree Construction: With a linear transformation of the matrix $\boldsymbol{M}$, such as QR or Cholesky decomposition, it is possible to rewrite $\boldsymbol{M}$ as a product of two matrices, one of which upper triangular [10]. In this work, QR decomposition has been employed so that, given $\boldsymbol{M} = \boldsymbol{Q}R$, (2) can be rewritten as

$$\arg \min_{\boldsymbol{s} \in Q^n} \|\boldsymbol{y} - \boldsymbol{Q}Rs\|^2 = \arg \min_{\boldsymbol{s} \in Q^n} \|\boldsymbol{Q}^{\boldsymbol{T}}\boldsymbol{y} - \boldsymbol{R}s\|^2 = \arg \min_{\boldsymbol{s} \in Q^n} \|\tilde{\boldsymbol{y}} - \boldsymbol{R}s\|^2 \tag{4}$$

Manuscript received July 25, 2007. First published November 18, 2008; current version published December 17, 2008. This work was supported in part by the STREP Project IST-026905 (MASCOT) within the sixth framework program of the European Commission, by the MEADOW (MEsh ADaptive hOme Wireless nets) Project, founded by the Italian government, and by the IEEE.

B. Cerato and E. Viterbo are with DEIS, Universitá della Calabria, 87036 Calabria, Italy.

G. Masera is with CERCOM, Politecnico di Torino, 10129 Torino, Italy.

Digital Object Identifier 10.1109/TVLSI.2008.2003163

where we have exploited the orthogonality of $\boldsymbol{Q}$ and $\tilde{\boldsymbol{y}} = \boldsymbol{Q}^T\mathbf{y}$. The upper triangular structure of the factored matrix enables to take every component separately into account for the computation of the distance between the two points. The distance $d^2(\boldsymbol{s}) = \|\tilde{\boldsymbol{y}} - \boldsymbol{R}s\|^2$ can also be computed recursively as follows. Let us define the partial metric as in [6]

$$T^{(l)}(\boldsymbol{s}^{(l)}) = \begin{cases} 0, & \text{if } l = n+1 \\ T^{(l+1)}(\boldsymbol{s}^{(l+1)}) + |\tilde{y}_l - \sum_{j=l}^{n} R_{lj}s_j|^2 \\ = T^{(l+1)}(\boldsymbol{s}^{(l+1)}) + |\tilde{y}_l - \displaystyle\sum_{j=l+1}^{n} R_{lj}s_j - R_{ll}s_l|2 \\ = T^{(l+1)}(\boldsymbol{s}^{(l+1)}) + |\psi_l^{(l+1)} - R_{ll}s_l|^2, \\ \text{if } l = 1, \ldots, n \end{cases} \tag{5}$$

where $\boldsymbol{s}^{(l)} = [s_l \;\; s_{l+1} \;\; \cdots \;\; s_n]$, and

$$\psi_l^{(l+1)} = \tilde{y}_l - \sum_{j=l+1}^{n} R_{lj}s_j \tag{6}$$

with $\psi_n^{(n+1)} = \tilde{y}_n$. Then we can write $T^{(1)}(\boldsymbol{s}) = d^2(\boldsymbol{s})$.

One of the most interesting consequences of this interpretation is that the exploration of the lattice can be thought as a tree traversal. This tree has n levels and every node at each level has Q sons. At every level the radius constraint (3) must be verified and satisfied, otherwise the branch is pruned.

2) Tree Exploration: Several algorithms have been studied in order to make the tree traversal efficient. An efficient algorithm has been proposed by Schnorr and Euchner (SE) [12]. The SE algorithm has intrinsically variable throughput and this makes it not very suitable for hardware implementation. The key to make this algorithm efficient or, at least, with predictable throughput, is to make an effective pruning.

III. First Hardware Implementation: Parametrizable Solution

In the developed architecture, the datapath width, the size of the search tree and the modulation scheme are tunable parameters that can be statically configured to make the detector adaptable to different systems. The system is described with reference to the special case of the Golden code. The key elements of the developed architecture are described in the following paragraphs.

A. Flexible Hardware Solution

The key processing task in the tree exploration algorithm is given by (5), where we recall that $\psi_l^{(l+1)} = \tilde{y}_l - \sum_{j=l+1}^{n} R_{lj}s_j$, is the lth entry of an n elements vector $\boldsymbol{\psi}^{(l+1)}$, and $l+1$ is the tree level we are referring to. At level l, the generic ith entry of this vector can be decomposed in a recursive manner through the following expression:

$$\psi_i^{(l)} = \begin{cases} \tilde{y}_i, & \text{if } l = n+1 \\ \psi_i^{(l+1)} - R_{il}s_l, & \text{if } l = n, \ldots, 1 \end{cases} \tag{7}$$

where i is in the range $1, \ldots, n$ while the level l decreases from $n+1$ to 1.

The whole $\boldsymbol{\psi}^{(l)}$ can therefore be updated by means of

$$\boldsymbol{\psi}^{(l)} = \boldsymbol{\psi}^{(l+1)} - \boldsymbol{R}_l s_l, \quad l = n, \ldots, 1 \tag{8}$$

where $\boldsymbol{R}_l$ is the lth column of $\boldsymbol{R}$ and the initial value is given by $\boldsymbol{\psi}^{(n+1)} = \tilde{\boldsymbol{y}}$.

In order to minimize the final metric $d^2(\boldsymbol{s})$ at each level of the tree the minimum $\psi_l^{(l+1)} - R_{ll}s_l$ value between all sons must be selected. More precisely, at each tree node, placed at level l, the following three main operations have to be accomplished:

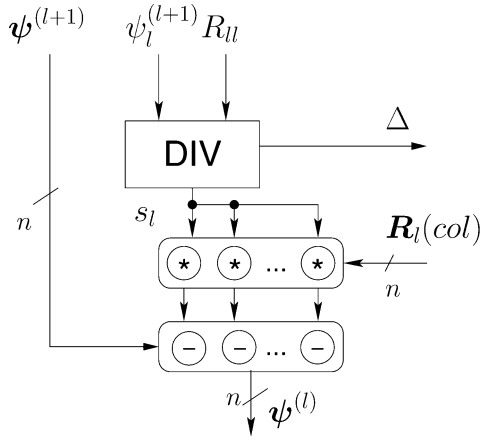

Fig. 1. **U_psi** unit datapath.

1) s_l that minimizes the difference $|\psi_l^{(l+1)} - R_{ll}s_l|$ is selected;
2) partial metric $T^l(\boldsymbol{s}^{(l)})$ is calculated according to (5);
3) for each $i = 1, \ldots, n$, $\psi_i^{(l)}$ is evaluated for the selected s_l value, according to (7).

Thus, the straightforward minimization of partial metrics $T^l(\boldsymbol{s}^{(l)})$ requires the difference computation for all the possible values of s_l. This technique becomes increasingly expensive with high order modulations, due to the large number of required operations.

In the proposed architecture, the minimization of $T^l(\boldsymbol{s}^{(l)})$ is rearranged in two steps. In the first processing step, the value of s_l that minimizes the difference $|\psi_l^{(l+1)} - R_{ll}s_l|$ is directly selected by means of a division; the obtained s_l is then used to generate $\psi_i^{(l)}$ amounts in (7), for all $i = 1, \ldots, n$. At the second step, (5) is finally evaluated to obtain the actual metric value $T^{(l)}$ for the selected son. Two functional blocks, **U_psi** and **Metric_compute** units, are allocated to perform the indicated processing steps.

In order to find the value of s_l able to minimize $|\psi_l^{(l+1)} - R_{ll}s_l|$, **U_psi** unit (shown in Fig. 1) receives as inputs the $\psi_l^{(l+1)}$ derived at the upper tree level, together with the lth diagonal element of matrix $\boldsymbol{R}$. The result of the division $\psi_l^{(l+1)}/R_{ll}$ is approximated to the closest odd integer. This approximation is equivalent to the selection of the closest point in a Q-PAM constellation.

The resulting value directly provides the desired s_l for the analyzed node. The new $\psi_i^{(l)}$ values are then evaluated in parallel to be used at the lower tree level. Vector $\boldsymbol{\psi}^{(l)}$ is stored in a dedicated memory, which will be later referred to as *Psi memory* in the global architecture given in Fig. 4.

The Δ output in Fig. 1 is defined as

$$\Delta = s_l - \frac{\psi_l^{(l+1)}}{R_{ll}}$$

and it represents the correction term to be applied to the division result in order to take the closest point in the equivalent PAM constellation. The use of Δ will be described later in this section.

The **Metric_compute** unit realizes the second processing step, evaluating the new metric $T^{(l)}$ for the selected son. It receives from the upper tree level $T^{(l+1)}$ as input together with the $\psi_l^{(l)}$ value generated by **U_psi** unit; the obtained $T^{(l)}$ is propagated to the lower tree level.

The described approach, and particularly the use of a division to obtain the optimal s_l, allows avoiding multiple metric computations; thus it offers low complexity and, at the same time, flexibility in terms of supported modulation schemes.

It is worth noting that, although the described technique introduces the division $\psi_l^{(l+1)}/R_{ll}$, only a few values of this ratio are of interest

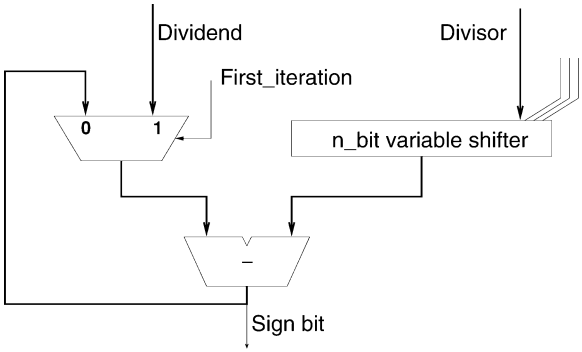

Fig. 2. Block diagram of the divisor.

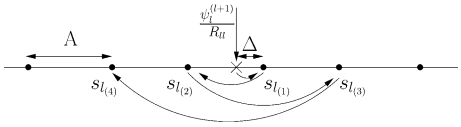

Fig. 3. Method used to select alternative nodes in U_psi_step unit.

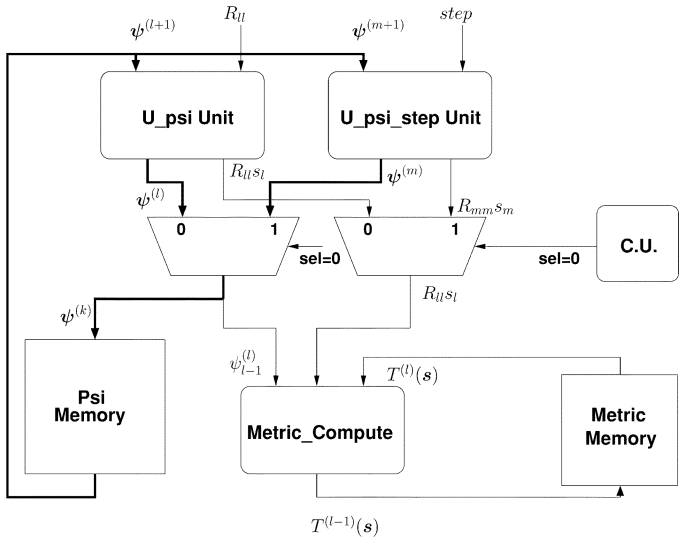

Fig. 4. *Sphere decoder* block scheme (case of a node expanded in the depth-first mode, with no pruning).

for the algorithm, those that correspond to the equivalent PAM constellation points $\pm 1, \pm 3, \ldots$. As a consequence, a general purpose hardware divisor is not necessary and the required operation can be executed by means of a simplified component able only to find the closest integer solution of this division and to determine if the approximation is by defect or by excess: the first $\log_2 Q$ steps of a successive subtraction divider [13] can be employed to this purpose, where Q^2 is the number of signals in the QAM constellation. This divider has a very simple architecture that uses only shifts and subtractions and employs a dichotomous process to find the requested value after $\log_2 Q$ steps. In the block diagram of Fig. 2, the multiplexer selects the dividend at the first step and the subtraction result in the following ones; the n-bit variable shifter is used to shift the divider by a number of positions that changes from the initial value of $\log_2 Q - 1$ down to 0. The subtractor returns the result one bit per iteration, starting from the most significant one.

B. Pipelined Tree Exploration

In order to ensure that a new node is expanded at each clock cycle, a new, alternative metric must be available also after a pruning operation has taken place. As a consequence, when the metrics of a given father node are evaluated, two "candidate" nodes are concurrently computed: the first one is a direct son of the current node and it is processed by the **U_psi** unit, while the alternative node, placed at a higher level in the tree, is concurrently computed by the **U_psi_step** sub-circuit. Both of them generate novel $\boldsymbol{\psi}^{(l)}$ values for the next step in the tree traversal.

U_psi and **U_psi_step** units share a very similar architecture, however the latter does not need to perform the division, as the second best choice for s_l (and thus for the alternative node) can be easily derived as follows. When **U_psi** unit computes the division, the result is approximated either by defect or by excess to the nearest PAM constellation point: the best choice for s_l is given by (see Fig. 3)

$$s_{l_{(1)}} = \frac{\psi_l^{(l+1)}}{R_{ll}} + \Delta \tag{9}$$

where Δ is the correction term provided as output by the **U_psi** unit (see Fig. 1).

The sign of Δ is used by **U_psi_step** unit to take the second (and following) nearest point in the PAM constellation, according to the following rule:

$$s_{l_{(k)}} = s_{l_{(k-1)}} - (-1)^k \operatorname{sign}(\Delta)(k-1)A \tag{10}$$

where A is the distance between two consecutive points and the initial value, $s_{l_{(1)}}$, is the closest point given in (9).

Fig. 3 shows the sequence of alternative nodes selected at a given tree level, after the occurrence of pruning. Depending on the values assumed by the father node metric, the algorithm descends along the tree, reaching the son node, or it moves to the alternative node on the same level. It is worth noting that the computations of the $\boldsymbol{\psi}^{(l)}$ values for both son and alternative nodes are performed concurrently with the elaboration of the T_l metric for the father node. This approach provides a significant speedup to the inherently serial SE Sphere Decoding Algorithm and has a limited impact on complexity.

C. Global Architecture

The block scheme of the SE tree-traversal circuit is finally depicted in Fig. 4. The following four fundamental processing blocks can be identified in this architecture:

- **U_psi** unit, which selects the most probable son of the current node and computes updated $\boldsymbol{\psi}^{(l)}$ through expression (8) (see also Fig. 1);
- **U_psi_step** unit, which selects the alternative node to be expanded and computes for this node the same amount;
- **Metric_compute** unit, which computes metric of the current node $T^{(l)} = T^{(l+1)}(\boldsymbol{s}^{(l+1)}) + |\psi_l^{(l+1)} - R_{ll}s_l|^2$, as in (5);
- **C.U.**, control unit devoted to the proper selection of the tree search direction.

The **C.U.** constitutes the core of the tree traversal algorithm and it must also carry-out two further tasks: to verify the pruning condition and, on the basis of this verification, to properly dispatch data between the other units. Symbols given in Fig. 4 are related to the case of a node expanded in the depth-first mode, with no pruning: inputs of the **Metric_compute** unit are fed with outputs provided by **U_psi** block. When a pruning occurs, multiplexers are switched and metrics related to the alternative node are selected. Finally, **Psi Memory** stores $\boldsymbol{\psi}^{(l)}$ vectors.

TABLE I
SYNTHESIS RESULTS AND COMPARISONS (16 BITS)

				This work		
Reference	ASIC-I [6]	ASIC-II [6]	[7]	PARAMETRIZABLE IMP.		FLEXIBLE IMPL.
Antennas	4×4			2×2 per two channel uses		
Modulation	16-QAM	16-QAM	16-QAM	16-QAM		4,16,64-QAM
Detector	depth-first sphere	sphere	K-best sphere	depth-first sphere		
BER Perf.	ML	Close to ML	Close to ML	ML		
Tech. [μm]	0.25	0.25	0.35	0.25	0.13	0.13
Core Area [GE]	117K +preproc.	50K +preproc.	91K +preproc.	56K +preproc.	45K +preproc.	55K +preproc.
Max. Clock	51 MHz	71 MHz	100 MHz	109 MHz	250 MHz	217 MHz
Throughput	73 Mbps @SNR=20 dB	169Mbps @SNR=20 dB	52 Mbps	73 Mbps @SNR=20 dB	167 Mbps @SNR=20 dB	146 Mbps @SNR=20 dB

IV. SECOND HARDWARE IMPLEMENTATION: FLEXIBLE MODULATION SOLUTION

The capability of managing more than one modulation scheme in order to adaptively select the most efficient one according to user needs and channel conditions, is one of the most important requirements of modern wireless communications systems. The Golden code, thanks to the non-vanishing determinant property, is very well suited for such application since it achieves the best performance independently of the QAM size [3]. In order to take full advantage of this Golden code feature, an enhanced implementation has been realized to allow run-time choice of the modulation scheme.

This implementation relies on the same architecture described in the previous section, with an additional parameter that allows the run-time selection of the constellation and that impacts mainly on the control logic.

At each level of the tree, the C.U., besides the pruning condition verification, also carries out a second verification task, related to the *mapping constraint*: it verifies if a certain value of s_l still belongs to the specified constellation and uses this information to drive the processing.

This *mapping constraint* must also be taken into account in the division $\psi_l^{(l+1)}/R_{ll}$. As the number of acceptable values for this operation depends on the adopted modulation, the constellation parameter is used to dynamically drive the iterations of the dichotomic division algorithm.

Although the architecture deals with the implementation of the Golden Code where $n = 8$, it is also scalable in terms of n. Increasing the number of transmitting and receiving antennas: a larger value of the n parameter can be set in the VHDL code to synthesize detectors for larger STBcodes. Of course a larger n implies a more expensive architecture: particularly the value of n mainly affects the number of ψ values to be evaluated in parallel in figures and, the depth of the tree and the size of ψ memory.

The complexity of **U_psi** and **U_psi_step** units grows almost linearly with n; the memory size increases as n^2, because n values of $\psi^{(l)}$ have to be stored for n tree levels. Finally, the throughput is expected to decrease with n, since the number of visited nodes grows, but this effect is strongly dependent also on the supported code.

V. SYNTHESIS RESULTS

The first proposed architecture, tailored to process the 16-QAM case, has been synthesized on both 0.13- and 0.25-μm CMOS standard cell technologies, using the Synopsys Version Z-2007.03-SP1; synthesis on 0.13-μm technology has been performed for the second flexible architecture. A commercial low-power library has been chosen.

A wide range of simulations have been carried out in order to determine the effects of different fixed-point representations on the performance for both 16 and 64-QAM modulation schemes, they show that a total of 12 bits lead to performance very close to the floating-point case for 16-QAM modulation, while 14 bits are necessary in the detection of 64-QAM signals. In order to enable the direct comparison with existing hardware realizations [6, I and II ASIC], [7], a 16 bit datapath has been chosen and for the first implementation a 16 QAM has been adopted. The overall decoder has also been simulated with the uncoded 4×4 MIMO system and throughput figures reported in Table I refer to this configuration.

The comparison of the described architectures to existing implementations tend to be quite difficult to carry out, because different approaches have been adopted: particularly, our solution implements the ML detection algorithm by means of a serial architecture, while the first ASIC in [6] maps the same algorithm onto a parallel structure and the second ASIC in [6] makes use of a serial scheme to realize a close to ML algorithm. These differences must be carefully evaluated while reading results in Table I.

Comparing the parameterizable architecture to parallel implementations in Table I, the solution described in [7] and the first ASIC presented in [6], it can be observed that a single metric computation is performed at each cycle, instead of multiple parallel metric computations. This characteristic justifies both the reduced complexity and the inherent flexibility of the proposed architecture. At the same time, thanks to the adopted pipelined architecture, a remarkable average decoding throughput is achieved without any highly specialized structure.

Implementation cost is slightly higher than for the second ASIC proposed in [6], where a serial approach is also adopted, in conjunction with a close to ML algorithmic approach.

On the other hand, the flexible implementation in the last column of Table I prove the limited complexity and performance overhead associated to the capability of dynamically adapting to different modulations (4-, 16-, and 64-QAM).

Finally, the finite precision analysis of the decoding algorithm proved that a negligible performance degradation is introduced in the case of 64-QAM modulation when a total of 14 bits are used: post-synthesis results show a complexity reduction of 8 Kgates (14% of core area) and a throughput increase to 150 Mb/s.

VI. Conclusion

A novel approach has been presented for the hardware implementation of a sphere decoder detector: the proposed solution uses a single metric computation per cycle and pipelining, breaking the sequential nature of SD algorithm.

The main element of novelty of the described approach is in its inherent flexibility that makes it suitable for adaptive modulation schemes. Two different hardware architectures have been designed: the first implementation is a parametrizable one, while the second is able to adapt on the fly to different modulation schemes.

Final synthesis results of the proposed architectures show a significant complexity reduction (approximately 50% for 16 QAM modulation) with respect to parallel structures. This is mainly due to the single metric computation per cycle. A remarkable average decoding throughput can be achieved with both implementations, thanks to the pipelining technique, even if the hardware was not tailored on a single modulation scheme as all previously proposed solutions.

References

[1] M. Damen, A. Tewfik, and J.-C. Belfiore, "A construction of a space-time code based on number theory," *IEEE Trans. Inf. Theory*, vol. 48, no. 3, pp. 753–760, Mar. 2002.

[2] B. A. Sethuraman, B. S. Rajan, and V. Shashidhar, "Full-diversity, highrate space-time block codes from division algebras," *IEEE Trans. Inf. Theory*, vol. 49, no. 10, pp. 2596–2616, Oct. 2003.

[3] H. Yao and G. Wornell, "Achieving the full MIMO diversity-multiplexing frontier with rotation-based space-time codes," presented at the Allerton Conf. Commun., Control Comput., Monticello, IL, Oct. 2003.

[4] E. Viterbo and J. Boutros, "A universal lattice code decoder for fading channels," *IEEE Trans. Inf. Theory*, vol. 45, no. 5, pp. 1639–1642, Jul. 1999.

[5] K. Wong, C. Tsui, R. S. Cheng, and W. Mow, "A VLSI architecture of a K-best lattice decoding algorithm for MIMO channels," in *Proc. IEEE ISCAS*, 2002, pp. 273–276.

[6] A. Burg, M. Borgmann, M. Wenk, M. Zellwegger, W. Fichtner, and H. Bölcskei, "VLSI implementation of MIMO detection using the sphere decoding algorithm," *IEEE J. Solid-State Circuits*, vol. 40, no. 7, pp. 1566–1577, Jul. 2005.

[7] Z. Guo and P. Nilsson, "A VLSI architecture for the Schnorr-Euchner decoder for MIMO systems," in *Proc. IEEE CAS Symp. Emerging Technol.*, Jun. 2004, pp. 65–68.

[8] P. W. Wolniansky, G. J. Foschini, G. D. Golden, and R. A. Valenzuela, "V-BLAST: An architecture for realizing very high data rates over the rich-scattering wireless channel," in *Proc. Int. Symp. Signals, Syst., Electron.*, Oct. 1998, pp. 295–300.

[9] E. Viterbo, "The golden code homepage," 2006. [Online]. Available: http://www1.tlc.polito.it/~viterbo/perfect_codes/Golden_Code.html

[10] M. O. Damen, H. El Gamal, and G. Caire, "On maximum-likelihood detection and the search of the closest lattice point," *IEEE Trans. Inf. Theory*, vol. 49, no. 10, pp. 2389–2402, Oct. 2003.

[11] U. Fincke and M. Pohst, "Improved methods for calculating vectors of short length in a lattice, including a complexity analysis," *Math. Computat.*, vol. 44, no. 170, pp. 463–471, Apr. 1985.

[12] C. Schnorr and M. Euchner, "Lattice basis reduction: Improved practical algorithms and solving subset sum problems," *Math. Program.*, vol. 66, no. 2, pp. 181–191, Sep. 1994.

[13] B. Parhami, *Computer Arithmetic. Algorithms and Hardware Designs*. Oxford, U.K.: Oxford Univ. Press, 2000.