TOWARDS APPROACHING NEAR-OPTIMAL MIMO DETECTION PERFORMANCE ON A C-PROGRAMMABLE BASEBAND PROCESSOR

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

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

ABSTRACT

Lattice Reduction aided softoutput MIMO detectors have been demonstrated to offer a promising gain. However, computing Log-Likelihood ratios (LLR) for near-optimal MIMO detection, still poses a significant challenge for practical implementations. In this work, we present counter-ML bit-flipping algorithm for LLR generation. The proposed LLR generation algorithm has been designed to take advantage of the previously reported list generation algorithm, Multi-Tree Selective Spanning (MTSS), by maximizing the reuse of computations. Afterwards, a C-programmable MIMO detector architecture providing both data level parallelism (DLP) and instruction level parallelism (ILP), is designed for implementation. The proposed solution supports multiple MIMO detection modes, with both hard and softoutput. Performance of the proposed solution can be tuned ranging from SIC to near-ML to near-MAP, by adjusting a single parameter. In case of 4×4 QAM-64, it achieves peak-throughputs of 2.43Gbps and 629Mbps in case of hard and softoutput MIMO detection, with only 66.37mW and 76.14mW respective power consumption.

1. INTRODUCTION

The upcoming wireless standards, 3GPP LTE-Advanced (LTE-A) and 802.11ac, have high throughput requirements and multi-mode operations. To meet the high-throughput requirements, baseband processors supporting parallelism, such as those combining Instruction Level Parallelism (ILP) and Data Level Parallelism (DLP), are required. In order to support multi-mode operation and allow performance/complexity trade-offs, programmable baseband solutions are required [1]. Furthermore, [2] [3] demonstrate that high performance software-defined solutions with cost comparable to dedicated architectures can be designed. However, a programmable MIMO detector implementation, that can provide near-optimal performance is still a significant challenge for practical systems.

Multi-Tree Selective Spanning [4] has been demonstrated to be suitable for programmable architectures and allows to conveniently tune detection performance. In this work, we further extend MTSS to support softoutput MIMO detection. In majority of tree-search based MIMO detection algorithms such as SD [5], FCSD [6] and SSFE [7], a significant reason of performance degradation is the missing counter-ML hypothesis. The missing counter-ML hypothesis causes a loss in BER performance in softoutput MIMO detection. LLR generation techniques that employ bit-flipping [8] [9], improve detection performance at low additional computational complexity. In this paper a new technique for LLR generation, counter-ML bit-flipping, is proposed. Counter-ML bit-flipping takes advantage of the MTSS algorithm to maximize the reuse of computations. Afterwards, a C-programmable baseband processor, providing both DLP and ILP, is designed for implementation. The proposed implementation supports multiple MIMO detection modes with both hard and softoutput, by employing MTSS and counter-ML bit-flipping, respectively. An algorithm/architecture co-design approach is used to provide a scalable solution, with tunable performance/energy trade-offs. The baseband processor can be operated in multiple modes with performance ranging from SIC to near-ML to near-MAP.

2. BACKGROUND AND SYSTEM MODEL

Consider a spatially multiplexed MIMO system with M_t transmit and N_t receive antennas. The vector of received symbols $y \in \mathbb{C}^{N_t \times 1}$ is given as,

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

where $s \in \mathbb{C}^{M_t \times 1}$ is the vector of transmitted symbols taken independently from a M-QAM constellation, and $n \in \mathbb{C}^{N_t \times 1}$ is the vector of complex Gaussian noise samples ($n_i \sim N(0, \sigma^2)$) and $\mathbf{H} \in \mathbb{C}^{N_t \times M_t}$ denotes the MIMO channel matrix. Lattice Reduction-aided linear MIMO detection has been proposed in [10] [11] [12]. To perform LR-aided MIMO detection first a reduced lattice basis is obtained as $\widetilde{\mathbf{H}} = \mathbf{HP}$, where $\mathbf{P}$ is generated by using a LR algorithm [13]. The system equation (1) can be rewritten as,

$$y = \mathbf{HPP}^{-1}s + n, \tag{2}$$

978-1-4799-2893-4/14/$31.00 ©2014 IEEE

Then the Moore-Penrose pseudo inverse of the transformed channel matrix, $\widetilde{\mathbf{H}}^{\dagger}$ is applied to obtain

$$\hat{z} = \mathbf{P}^{-1}s + w, \tag{3}$$

where w is colored noise with zero mean and variance $C_w = \sigma^2(\widetilde{\mathbf{H}}^{\mathbf{H}}\widetilde{\mathbf{H}})^{-1}$. A low complexity tree-searching method has been proposed in [14], by using the approximation $\widetilde{\mathbf{H}}^{\mathbf{H}}\widetilde{\mathbf{H}} \approx \mathbf{I}$ and performing QR-decomposition of $\mathbf{P}^{-1}$ as $\mathbf{P}^{-1} = \mathbf{QR}$, to obtain

$$\left\|\hat{z} - \mathbf{P}^{-1}s\right\|^2 = \|\hat{y} - \mathbf{R}s\|^2 \tag{4}$$

from (3), where $\hat{y} = \mathbf{Q}^{\mathbf{H}}\hat{z}$, $\mathbf{Q}$ is a unitary matrix and $\mathbf{R}$ is an upper-triangle matrix. This transforms (4) into a tree-search problem.

3. COUNTER-ML BIT-FLIPPING FOR SOFT OUTPUT MIMO DETECTION

Soft-output MIMO detection usually consists of two parts: (1) A list generator that gives a list of candidate symbol vectors, denoted by $\mathcal{L} \subseteq \Omega^M$, where Ω^M is the set containing all the possibilities of $M_t \times 1$ vector symbol s; (2) A Log-likelihood-ratio (LLR) generator that approximates the *a posteriori probabilities* (APP). Using the Max-Log Approximation (MAP), the $LLR(j,b)$ value corresponding to the b^{th} bit of the j^{th} scalar symbol in s can be calculated as,

$$\frac{1}{2\sigma^2}\left(\min_{s\in\mathcal{L}\cap\chi^0_{j,b}} \|\hat{y} - \mathbf{R}s\|^2 - \min_{s\in\mathcal{L}\cap\chi^1_{j,b}} \|\hat{y} - \mathbf{R}s\|^2\right) \tag{5}$$

where $\chi^0_{j,b}$ and $\chi^1_{j,b}$ are the disjoint sets of symbol vectors that have the b^{th} bit of the j^{th} scalar symbol in s equal to 0 and 1, respectively. Soft-output MIMO detection consists of two minima search problems, as shown in (5), which can be re-written as,

$$LLR(j,b) = \left\{ \begin{array}{l} \mathbf{T}^{ML} - \mathbf{T}^{\overline{ML}}_{j,b},\ s^{ML}_{j,b} = 0 \\ \mathbf{T}^{\overline{ML}}_{j,b} - \mathbf{T}^{ML},\ s^{ML}_{j,b} = 1 \end{array} \right\} \tag{6}$$

where $\mathbf{T}^{ML} = \left\|\hat{y} - \mathbf{R}s^{ML}\right\|^2$ and s^{ML} is obtained from the candidate list as,

$$s^{ML} = \arg\min_{s\in\mathcal{L}} \|\hat{y} - \mathbf{R}s\|^2, \tag{7}$$

In (6) $s^{ML}_{j,b} = 0$ denotes the b^{th} bit of the j^{th} scalar symbol of s^{ML} is 0, and vice versa. So $LLR(j,b)$ can be computed by first finding the ML solution (7) and then finding the counter-ML hypothesis given as,

$$\mathbf{T}^{\overline{ML}}_{j,b} = \min_{s^{\overline{ML}}\in\chi^{\overline{ML}}_{j,b}} \left\|\hat{y} - \mathbf{R}s^{\overline{ML}}\right\|^2, \tag{8}$$

where $\chi^{\overline{ML}}_{j,b}$ is obtained by flipping $s_{j,b} \in \mathcal{L}$ to $s^{\overline{ML}}_{j,b}$. In contrast to [9] where the corresponding bit of symbol vectors in $\mathcal{L}$ is flipped to both 0 and 1 to obtain two new sets $\chi^0_{j,b}$ and $\chi^1_{j,b}$, in this work $\mathcal{L}$ is flipped to obtain only the set $\chi^{\overline{ML}}_{j,b}$. If $s^{ML}_{j,b} = 0$ then $s^{\overline{ML}}_{j,b} = 1$. In this paper, MTSS [4] is considered for list generation, because of its low computational complexity, regular data flow and near-optimal performance. It has been demonstrated to achieve near-ML performance [4]. However, as in majority of other tree search algorithms, there are two major problems when using MTSS for softoutput MIMO detection. 1). MTSS cannot guarantee that counter ML hypothesis always exists i.e. it might happen that $\mathcal{L}\cap\chi^{\overline{ML}}_{j,b} = \phi$. This causes a degradation in BER performance due to the missing counter ML hypothesis. 2). Also, for an existing counter ML hypothesis $\chi^{\overline{ML}}_{j,b}$, MTSS does not guarantee the minimization of (8), as it solves (7) by using a local minimization criteria, i.e. spanning fixed number of nodes at each layer, for details see [4] and [7]. In order to tackle these two problems we propose the counter-ML bit-flipping strategy for LLR generation.

Starting from Level $i = N_t$, the PED (Partial Euclidean Distance) of a partial symbol vector $s^i = [s_i, s_{i+1}, .., s_{Nt}]$ is $T_i(s^i) = T_{i+1}(s^{i+1}) + \left\|e_i(s^i)\right\|^2$, where the PED-increment is $e_i(s^i) = \hat{y}_i - \sum_{k=i}^{Nt} R_{ik}s_k$. To generate the counter-ML hypothesis, the b^{th} bit of the j^{th} scalar symbol in $s^i = [s_i, .., s_j, .., s_{Nt}]$ is flipped to the counter-ML bit to obtain $s^{\overline{ML}}_{j,b} = [s_i, .., s_j + \Delta^{\overline{ML}}_{j,b}, .., s_{Nt}]$. The PED-increment for the bit-flipped symbol is given as,

$$\begin{aligned} e_i(s^{\overline{ML}}_{j,b}) &= \hat{y}_i - \sum_{k=i}^{j-1} R_{ik}s_k - R_{ij}(s_j + \Delta^{\overline{ML}}_{j,b}) - \sum_{k=j+1}^{Nt} R_{ik}s_k \\ &= e_i(s^i) - R_{ij}\Delta^{\overline{ML}}_{j,b} \end{aligned} \tag{9}$$

This shows that when flipping the b^{th} bit of the j^{th} scalar symbol in s to obtain $s^{\overline{ML}}_{j,b}$, only $e_i(s^i)$ with $i = [1 \text{ to } j]$ need to be updated, while $e_i(s^i)$ with $i = [j+1 \text{ to } N_t]$ remains unchanged. Note that, $e_i(s^i)$ with $i = [j+1 \text{ to } N_t]$ has already been computed during the list generation (MTSS), which can be reused. So the PED for a counter-ML bit-flipped symbol $s^{\overline{ML}}_{j,b}$, at a level i can be updated as,

$$T_i(s^{\overline{ML}}_{j,b}) = \left\{ \begin{array}{ll} T_{i+1}(s^{\overline{ML}}_{j,b}) + ||e_i(s^{\overline{ML}}_{j,b})||^2, & i = 1 \text{ to } j \\ T_{i+1}(s_{i+1}) & , i = j+1 \text{ to } N_t \end{array} \right\} \tag{10}$$

So flipping the b^{th} bit of j^{th} scalar symbol only requires updating $T_i(s^{\overline{ML}}_{j,b})$ for $i = [1 \text{ to } j]$, while $T_i(s_i)$ for $i = j + 1 \text{ to } N_t$ is reused, which is generated during the list generation.

Assume that, $s^{\overline{ML}}_{j,b_0} = 0$, which means that bit at location b_0 in $s^{\overline{ML}}_j$ equals 0. So the bit b_0 in s_j has to be flipped to 0, in order to obtain $s^{\overline{ML}}_{j,b_0}$. For example, if the $s_j = 3$ having the corresponding bit $[b_1 = 0, b_0 = 1]$ (Fig.1) and

$s_{j,b_0}^{\overline{ML}} = 0$, then $\Delta_{j,b_0}^{\overline{ML}} = -2$ as shown in Table 1. The counter-ML bit-flipping can be performed by a simple addition, $s_j + \Delta_{j,b_0}^{\overline{ML}} = s_{j,b_0}^{\overline{ML}}$. In this work, a LUT is used to store the bit-flipping values, shown in Table 1. In case of QAM-16, LUT with 8 entries is required and is stored using $8 \times 4bit$ values. LUT is shown only for $s_j = [-3, +3]$, in a similar way $s_j = [-1, +1]$ LUT entries can be constructed.

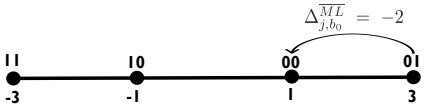


Fig. 1. Gray-coded QAM-16 points (real-axis).

	b_1		b_0	
	$s_{j,b_1}^{\overline{ML}} = 0$	$s_{j,b_1}^{\overline{ML}} = 1$	$s_{j,b_0}^{\overline{ML}} = 0$	$s_{j,b_0}^{\overline{ML}} = 1$
$s_j = +3$	0	-6	-2	0
$s_j = -3$	6	0	+2	0

Table 1. LUT for generating $\Delta_{j,b}^{\overline{ML}}$

4. ARCHITECTURE DESIGN

The C-programmable baseband processor shown in Fig.2, is designed using the TARGET tool suite [15]. The designed ASIP supports both hardoutput and softoutput MIMO detection, by implementing MTSS [4] and the counter-ML bit-flipping, respectively. The baseband processor is configured for 4×4 and QAM-64 MIMO detection. However, it can be configured to support other modes as well i.e. $2 \times 2, 8 \times 8$ and QAM-4, QAM-16 by changing the firmware only, as it is C-programmable. DLP is enabled across each Functional Unit (FU) by 8-way SIMD operations, while all the FUs can also perform independent execution, hence providing ILP. The processor has two distinct data paths 16-bit and 6-bit (Fig.2). 16-bit data is stored in the Register Files VWRF0 and VWRF1, while 6-bit data is stored in VRF. 16-bit operations and 6-bit operations are performed on FUs VWFU0/1 and VFU0/1, respectively. The shared register file VWR between the VWRF0/1, acts as a scratchpad memory and is accessed by the MOV FUs. $\hat{y}$ and $\mathbf{R}$ are stored in the 16-bit memory DMEM(VW)0/1, respectively and are loaded to VWRF0/1 by the load/store FUs (LD/ST). While, the generated candidate list $\mathcal{L}$ is stored in VRF and is written to DMEM0. Each of the FU in 16-bit and 6-bit data path performs 8×16 bit and 8×6 bit operations, respectively. VWR and VRF has 64 entries of 8×16 bit and 8×6 bit, respectively. While, each of VWRF0/1 has 32 entries of 8×16 bit. The motivation behind using a 6-bit data path is that, both $\mathcal{L}$ (which are basically constellation points Fig.1) and $\Delta_{j,b}^{\overline{ML}}$ (Table 1), can be stored using 6-bits. Note that the candidate list $\mathcal{L}$ has to be big, in order to achieve near-optimal performance, so using a 6-bit register file (VRF) saves both power and area. By using same bit widths for both $\mathcal{L}$ and $\Delta_{j,b}^{\overline{ML}}$, same instructions can be used for performing the common operations in MTSS [4] and counter-ML bit-flipping. This reduces the instruction set of the processor, as explained below:

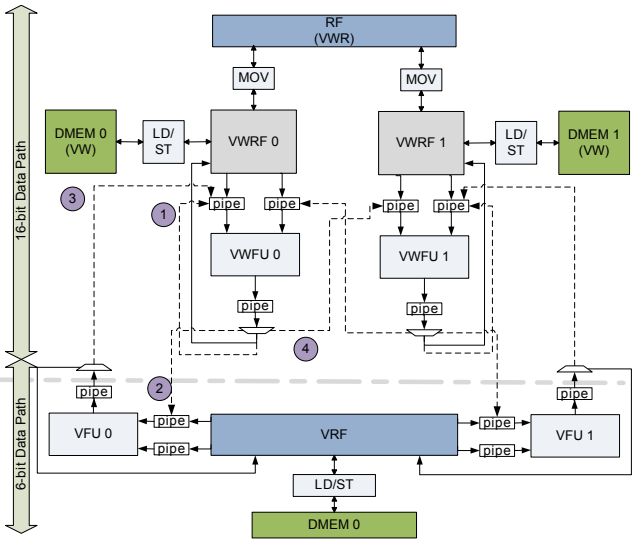


Fig. 2. C-programmable baseband processor supporting both Hardoutput and Soft-output MIMO detection.

1) Slicing and Enumeration: In list generation algorithms slicing and enumeration is required. Both slicing and enumeration are supported on VFU0/1. A MAC operation $(\hat{y}_i - R_{ik}s_k)$, is always performed after slicing/enumeration. By-pass path shown by the dashed line 3 in Fig.2, directly feeds the constellation point s (obtained by slicing/enumeration) to VWFU0 for a MAC operation. While $\hat{y}_i$ and R_{ik} are read from VWRF0. This by-passing decreases register accesses to VRF. Details of slicing and enumeration are not explained here due to space limitations, for details see MTSS [4].

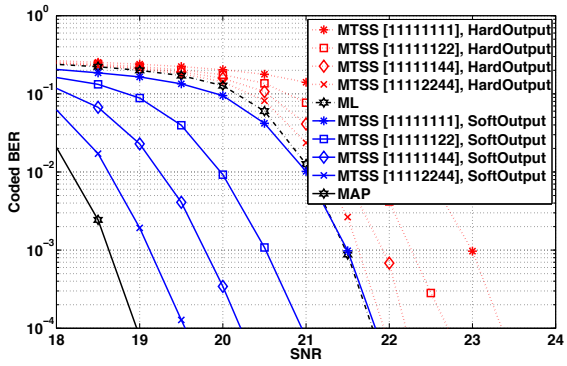
2) MAC operation: In both MTSS and LLR generation MAC operations are performed extensively. In list generation $\hat{y}_i - R_{ik}s_k$ and in LLR generation (9) are the MAC operations. Both of these operations are supported on VWFU0/1. By using 6-bit for both s_k and $\Delta_{j,b}^{\overline{ML}}$, they are performed by the same instruction.

3) $\Delta_{j,b}^{\overline{ML}}$ Generation: $\Delta_{j,b}^{\overline{ML}}$ is generated by VFU0/1. The output of the VFU0/1 can be directly read into the VWRF0/1 to perform the MAC operation (9). This is enabled by the by-pass path shown by the dashed line 3 in Fig.2. This avoids writing the $\Delta_{j,b}^{\overline{ML}}$ to VRF, hence decreases access to VRF.

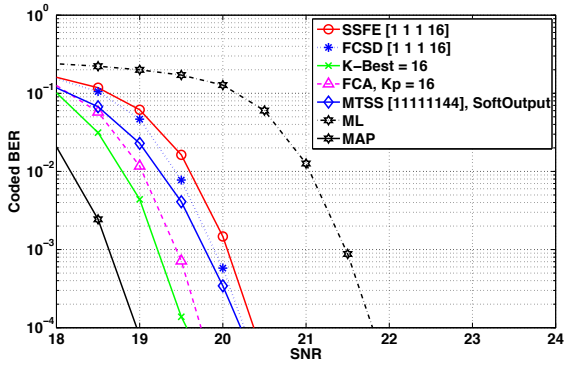
4) PED update operation: Updating the PED values is also required in both MTSS and LLR generation, as shown in (10). PED update operations are supported on both VWFU0/1. Note that the PED update is always performed after a MAC operations. In order to avoid storing the result of a MAC operation (9) in the VWRF0/1, by-pass paths are enabled (shown by the dashed line 4 in Fig.2). This by-pass directly outputs the results of a MAC operation performed on VWFU0 to the input of VWFU1, so the PED update can be performed in the next cycle. Hence, minimizing register file access.

5. RESULTS

A 4×4 MIMO system using QAM-64, with the 3GPP-LTE channel model and 1/2-rate Turbo coding is considered. DRP-LR [4] is used for LR. Coded BER performance of MTSS is shown in Fig.3(a). In case of hardoutput the candidate with the least PED given by (7), is demapped to the QAM constellation. While in case of softoutput, counter-ML bit-flipping is used for LLR generation. MTSS can be operated in different modes by choosing the spanning vector m [4]. Fig.3(a) shows that, MTSS with $m = [11112244]$ achieves near-ML and near-MAP performance in case of hard and softoutput detection, respectively. Fig.3(b) provides the coded BER performance comparison of MTSS to SSFE [7], Fixed complexity sphere detector (FCSD) [6], LR-aided Fixed candidates algorithm (FCA) [14] and the K-best algorithm. In order to have a fair comparison same number of candidates, i.e. the candidate list size $\mathcal{L} = 16$, is considered. Fig.3(b) shows that MTSS with counter-ML bit-flipping provides better BER performance than both SSFE and FCSD. The BER performance of K-best and FCA is better, as they are based on the K-best criteria that uses strict-sorting based on PED at each detection layer, while MTSS uses a local-search criteria. MTSS with $m = [11112244]$ achieves near-MAP performance, Fig.3(b). VHDL of the proposed design is generated using



(a)



(b)

Fig. 3. 4×4 QAM-64 (a) MTSS with both Hardoutput and Softoutput (b) MTSS compared to other MIMO detectors.

TARGET tool suite [15] and is synthesized using Cadence RTL compiler with a 40nm standard digital cell library. Operating at f_{clk}=813MHz, the processor consumes 66.37mW and 76.14mW power at 1.1-V supply voltage in hard and soft-output modes, respectively. Counting a 2-input NAND gate as one equivalent gate, the proposed design takes 584k gates equivalent (kGE). Table 2 shows the throughput and energy efficiency of the proposed designed in different modes. Table

	Hardoutput			Softoutput		
m	Cycles T_c	Throughput (Mbps)	Energy Efficiency (pJ/bit)	Cycles T_c	Throughput (Mbps)	Energy Efficiency (pJ/bit)
[11111111]	8	2439	27.21	31	629	120.97
[11111122]	16	1219	54.42	96	203	374.64
[11111144]	57	342	193.88	202	96	788.31

$Throughput = (f_{clk} \times nbps \times N_t)/T_c$

Table 2. 4×4, QAM-64 in different modes.

3 provides a comparison to recently reported MIMO detector implementations. The proposed design is almost $7\times$ better in terms of energy efficiency, than our previously reported work [4], which is basically due to the 6-bit data path and by adding dedicated by-passing. Although, the proposed design cannot compete with the ASIC design in terms of energy and area efficiency (Table 3). However, the processor is capable of supporting multiple performance modes ranging from SIC to near-ML to near-MAP. Besides, it can be configured to support other QAM modulations QPSK and QAM-16 by changing the firmware only, as it is C-programmable.

	[16]	[17]	[8]	[4]	This Work	
Process(nm)	130	130	65	40	40	
Algorithm	K-Best K = 10	K-Best K = 10	Imbalanced FSD	MTSS		
Decision Type	Hard	Soft	Hard	Hard	Hard	Soft
Gate Count (kG)	114	174	88.2	6364	584	
f_{clk} (MHz)	282	270	165	700	813	
Throughput (Mbps)	675	655	1980	730		
Normalized Throughput (Mbps) †	2193	2128	3217.5	730	2439	629
Power (mW)	135 @ 1.3 V	195 @ 1.3 V	102.7 @ 1.2 V	147.16 @ 1.1 V	66.37 @ 1.1 V	76.14
Normalized Power (mW) ‡	29.74	42.95	53.10	147.16		
Energy Efficiency (pJ/b)	13.55	20.18	16.50	201.58	27.21	120
Area Efficiency (Mbps/kG)	19.24	12.22	36.47	0.1147	4.17	1.07

† Technology scaling from process x to 40 nm $f_{clk} = (f_{clk}^x * x)/(40nm)$
‡ Power is normalized using $Power \times (1.1V/Volt)^2 \times (40nm/x)$

Table 3. 4×4 QAM-64 MIMO Detector Implementations.

6. CONCLUSION

In this work, an LLR generation algorithm called counter-ML bit-flipping is proposed. Afterwards, a C-programmable MIMO detector architecture supporting multiple MIMO detection modes with both hard and softoutput, is designed for implementation. Implementation results for 3GPP-LTE 4×4 QAM-64 show that, the proposed implementation delivers peak-throughputs of 2.43Gbps and 629Mbps in case of hard and softoutput MIMO detection, with only 66.37mW and 76.14mW respective power consumption. Moreover, the proposed implementation can be configured to operate in different modes, with performance ranging from SIC to near-ML to near-MAP, providing performance/energy trade-offs.

7. REFERENCES

[1] U. Ramacher, "Software-Defined Radio prospects for multistandard mobile phones," *Computer*, vol. 40, no. 10, pp. 62 –69, oct. 2007.

[2] Xuezheng Chu and J. McAllister, "Software-defined sphere decoding for FPGA-based MIMO detection," *Signal Processing, IEEE Transactions on*, vol. 60, no. 11, pp. 6017–6026, 2012.

[3] Chengwei Zheng, J. McAllister, and Yun Wu, "A kernel interleaved scheduling method for streaming applications on soft-core vector processors," in *Embedded Computer Systems (SAMOS), 2011 International Conference on*, 2011, pp. 278–285.

[4] U. Ahmad, M. Li, R. Appeltans, D. Nguyen, A. Amin, A. Dejonghe, L. Van der Perre, R. Lauwereins, and S. Pollin, "Exploration of lattice reduction aided soft-output mimo detection on a dlp/ilp baseband processor," *Signal Processing, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2013.

[5] M.O. Damen, H. El Gamal, and G. Caire, "On maximum-likelihood detection and the search for the closest lattice point," *Information Theory, IEEE Transactions on*, vol. 49, no. 10, pp. 2389 – 2402, oct. 2003.

[6] L.G. Barbero and J.S. Thompson, "Extending a fixed-complexity sphere decoder to obtain likelihood information for turbo-MIMO systems," *Vehicular Technology, IEEE Transactions on*, vol. 57, no. 5, pp. 2804–2814, 2008.

[7] Min Li, B. Bougard, E.E. Lopez, A. Bourdoux, D. Novo, L. Van Der Perre, and F. Catthoor, "Selective spanning with fast enumeration: A near maximum-likelihood MIMO detector designed for parallel programmable baseband architectures," in *Communications, 2008. ICC '08. IEEE International Conference on*, may 2008, pp. 737 –741.

[8] Liang Liu, J. Lofgren, and P. Nilsson, "Area-efficient configurable high-throughput signal detector supporting multiple MIMO modes," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 59, no. 9, pp. 2085 –2096, sept. 2012.

[9] Min Li, D. Novo, B. Bougard, F. Naessens, L. Van der Perre, and F. Catthoor, "An implementation friendly low complexity multiplierless llr generator for soft mimo sphere decoders," in *Signal Processing Systems, 2008. SiPS 2008. IEEE Workshop on*, 2008, pp. 118–123.

[10] 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.

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

[12] Ying Hung Gan, Cong Ling, and Wai Ho 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.

[13] D. Wubben, D. Seethaler, J. Jalden, and G. Matz, "Lattice reduction," *Signal Processing Magazine, IEEE*, vol. 28, no. 3, pp. 70–91, 2011.

[14] Wei Zhang and Xiaoli Ma, "Low-complexity soft-output decoding with lattice-reduction-aided detectors," *Communications, IEEE Transactions on*, vol. 58, no. 9, pp. 2621 –2629, september 2010.

[15] "Target tool suite website http://www.retarget.com," .

[16] M. Shabany and P.G. Gulak, "A 675 Mbps, 4x4 64-QAM K-Best MIMO detector in 0.13μm CMOS," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 20, no. 1, pp. 135 –147, jan. 2012.

[17] D. Patel, V. Smolyakov, M. Shabany, and P.G. Gulak, "VLSI implementation of a WiMAX/LTE compliant low-complexity high-throughput soft-output K-Best MIMO detector," in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, 30 2010-june 2 2010, pp. 593 –596.