An Adaptive Detector Implementation for MIMO-OFDM Downlink

Shahriar Shahabuddin, Janne Janhunen, Essi Suikkanen, *Heidi Steendam and Markku Juntti

Department of Communications Engineering and Centre for Wireless Communications, University of Oulu, Finland.

*DIGCOM research group, Department of Telecommunications and Information Processing, Ghent University, Belgium.

***Abstract*—Cognitive radio (CR) systems require flexible and adaptive implementations of signal processing algorithms. An adaptive symbol detector is needed in the baseband receiver chain to achieve the desired flexibility of a CR system. This paper presents a novel design of an adaptive detector as an application-specific instruction-set processor (ASIP). The ASIP template is based on transport triggered architecture (TTA). The processor architecture is designed in such a manner that it can be programmed to support different suboptimal multiple-input multiple-output (MIMO) detection algorithms in a single TTA processor. The linear minimum mean-square error (LMMSE) and three variants of the selective spanning for fast enumeration (SSFE) detection algorithms are considered. The detection algorithm can be switched between the LMMSE and SSFE according to the bit error rate (BER) performance requirement in the TTA processor. The design can be scaled for different antenna configurations and different modulations. Some of the algorithm architecture co-optimization techniques used here are also presented. Unlike most other detector ASIPs, high level language is used to program the processor to meet the time-to-market requirements. The adaptive detector delivers 4.88 - 49.48 Mbps throughput at a clock frequency of 200 MHz on 90 nm technology.**

I. Introduction

Cognitive radio (CR) systems have been proposed to efficiently use the available radio-frequency spectrum. The basic functions of the cognitive radio are the ability to sense the environment, the capacity to learn and the ability to adapt within any layer of the radio communication system [1]. The adaptivity becomes more challenging for the physical layer implementation in the mobile devices. The physical layer algorithms implemented for the mobile device are typically embedded in the printed circuit board (PCB) as fixed hardware accelerators. The fixed hardware implementations provide high data rate, use less logic gates and consume less power. The drawback of the fixed hardware implementation is that it operates on a fixed set of parameters only and it is very difficult to modify the design in the future. Therefore, the hardware accelerators are not the best choice for CR systems where flexibility is a key requirement. To solve this flexibility problem, it is possible to have different hardware accelerators to support different modes and different parameters, but the PCB size can become too large to accommodate all the accelerators. Another solution is to implement the adaptive algorithms as software to program the digital signal processor (DSP) chip of the mobile devices. At first sight, the software implementation is ideal for CR systems because they provide flexibility, where the parameters and the algorithms are changed by software. However, the problem in DSPs is the inability to achieve a high throughput, as their architectures are fixed and not tailored for any particular algorithm. The programmable architectures that are customized for a small set of algorithms can be the ideal choice for CR systems. The software-hardware codesign method, which is used for the programmable architectures. They provide not only the required flexibility for adaptive algorithms, but also a higher throughput than the pure software solutions. In this paper, we present an implementation of an adaptive detector as a customized programmable processor.

An adaptive detector for multiple-input multiple-output (MIMO) system selects the detection mode based on the channel estimation. The idea was proposed by Onggosanusi $et\ al.$ [2]. The adaptive detector contains different detectors with different complexities and bit error rate (BER) performances. The complex detectors are used for ill-conditioned channels and the simple ones for good channel conditions. The detection algorithm is changed based on a metric computed from the channel matrix. This metric can be the condition number of the channel or the distribution of the channel correlation [2]. The linear minimum mean-square error (LMMSE) and three variants of the selective spanning for fast enumeration (SSFE) detection algorithms are considered. The LMMSE filters can be used when the channel condition is good or low correlated channels. The SSFE forms a class of tree search algorithms that provides a feasible implementation complexity for moderately correlated channels [3].

The LMMSE and SSFE hardware implementations are at a mature stage and different implementations can be found in [4] and [5]. A unified hardware solution for both LMMSE and SSFE is difficult to implement. Typically, for an adaptive detector, the LMMSE and SSFE are implemented separately and used with a multiplexer to change between the algorithms when needed. We take different approach and design a unified programmable processor that supports both the LMMSE and SSFE realizations. The processor is based on the transport triggered architecture (TTA) paradigm. TTA is a processor design philosophy where the programmer can control the internal data transports between different function units of the processor [6]. Unlike the traditional processors, TTA exploits

the instruction level parallelism (ILP) by processing several instructions in a single clock cycle. The TTA based codesign environment (TCE) tool is used in this work to design the processor. TCE enables the designer to write an application with a high level language and design the target processor in a graphical user interface at the same time [7] [8]. The processor is programmed with C language to shorten the time-to-market. The processor achieves 4.88 - 49.48 Mbps throughput at a clock frequency of 200 MHz on 90 nm technology.

The rest of the paper is organized as follows: In Section II, the system model is explained. The simulation parameters and the error rate figures are also presented. In Section III, a brief discussion of LMMSE, SSFE and their usage in the adaptive detector is explained. Section IV presents the top level TTA processor architecture and programming techniques. In Section V, the performance of the implementation and the discussion is presented. The conclusions are drawn in Section VI.

II. System Model

We consider a MIMO system with orthogonal frequency-division multiplexing (OFDM) with N transmit antennas, which are sending data over the channel, and M receive antennas such that $N \geq M$. A layered space-time architecture with horizontal encoding is applied in the transmitter. Two streams of data bits are encoded separately according to the 3GPP Long Term Evolution (LTE) standard. The two streams of the encoded data bits are interleaved and multiplexed onto four antennas. The encoded bits are modulated to symbols with quadrature amplitude modulation (QAM). Bit interleaved coded modulation (BICM) is applied and data is transmitted over the channel via different antennas. A block diagram of the system model is presented in Fig. 1.

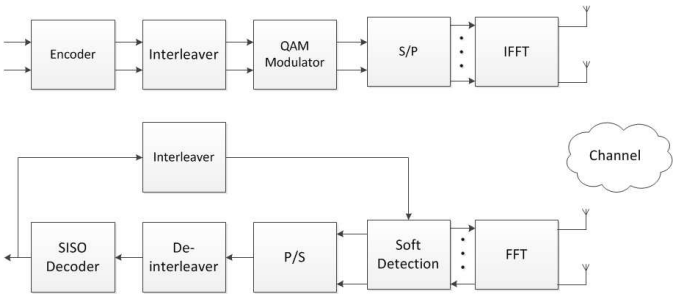

Fig. 1. The system model.

The received signal is composed of the multiplication of the complex channel matrix with the transmitted symbol vector distributed by additional white Gaussian noise caused by the channel. The received signal $\mathbf{y}$ can be represented as

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

where $\mathbf{y} \in \mathbb{C}^M$ is the received signal vector, $\mathbf{x} \in \mathbb{C}^N$ is the transmit symbol vector and $\mathbf{n} \in \mathbb{C}^M$ is the circularly symmetric complex white Gaussian noise vector with zero mean and variance σ^2. The (n, m)th component $h_{m,n}$ of the channel matrix $\mathbf{H} \in \mathbb{C}^{M \times N}$ is the channel coefficient from the nth transmit antenna to the mth receive antenna.

Soft detection is applied in the receiver to detect the transmitted signal. The LMMSE and SSFE algorithms with the spanning vectors [11111111], [11111222] and [11112223] are used for the detection. Those detection schemes are explained in Section III in more detail. Turbo coding is used as the forward error correction (FEC) scheme.

A similar scenario is simulated with a link level Matlab simulator. SSFE and LMMSE detectors are applied to a 4×4 system. The modulation method used for both detectors are 16-QAM and 64-QAM. A 5 MHz bandwidth corresponding to 512 OFDM subcarriers is considered. Each SNR point consists of 3360 OFDM symbols. One OFDM symbol consists of 512 subcarriers where 300 subcarriers are loaded with data and the rest are used as a guard interval. In the simulation, the mobile velocity is set to 3 kmph and the turbo decoder performs 6 iterations. The typical urban (TU) channel model with base station (BS) azimuth spread of 2 or 5 degrees is applied in the simulator. A list of the parameters for the simulations are presented in Table I.

TABLE I
Simulation and Channel Model Parameters

Number of subcarriers	512 (300) active
Channel coding	Turbo Coding
Coding Rate	1/2
Symbol duration	71.39 μ s
Symbol time	66.7 μ s
Cyclic prefix duration	4.69 μ s
Modulation	16 QAM and 64 QAM
user velocity	3 kmph
Channel model	TU
Number of paths	6
path delays	[0 ... 2510] ns
path power	[0 ... -20] dB
BS azimuth spread	2° / 5°
MS azimuth spread	35°

The BER results of the above mentioned detectors are shown in Figs. 2 - 4. The detectors are simulated in an uncorrelated channel with 64-QAM are considered in Fig. 2. It can be observed that the LMMSE and the SSFE with spanning vector [11111222] exhibit performance similar to each other. Therefore, the detector with the lowest complexity, i.e. LMMSE, can be used. The SSFE [11111111] can be used if the BER requirement is relaxed.

In Fig. 3, the detectors are simulated for a moderately correlated channel for 16-QAM and 64-QAM. For 64-QAM, LMMSE and SSFE [11111111] require a SNR beyond 30 dB which is impractical. Therefore, SSFE [11111222] and [11112223] have to be used for 64-QAM and a moderately correlated channel. The same statement is true for SSFE [11111222] and SSFE [11112223] for 16-QAM.

In Fig. 4, the detectors are simulated for a highly correlated channel for 16-QAM. For 64-QAM, all of the detectors work only beyond 30 dB. The LMMSE and SSFE [11111111] exhibit very high SNR requirements even for 16-QAM. Therefore, SSFE [11111222] and SSFE [11112223] can only be

used for highly correlated channel with 16-QAM. The BER result is helpful to determine the algorithm to be used for the adaptive detector for a specific set of parameters. More simulation results can be found in [9].

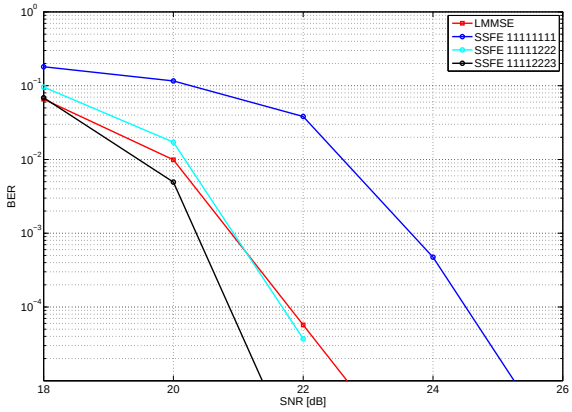

Fig. 2. Detector performance in an uncorrlated channel.

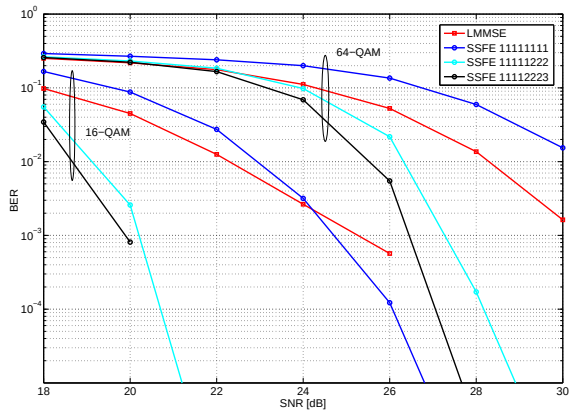

Fig. 3. Detector performance in a moderately correlated channel.

III. Detection Schemes

The function of the MIMO detector is to estimate the transmitted signal vector $\mathbf{x} \in \mathbb{C}^N$ and to feed the soft output to the decoder. Maximum likelihood (ML) detectors are optimal detectors. The ML detector calculates the Euclidean distances between received signal $\mathbf{y}$ and lattice points $\mathbf{Hx}$ and selects the closest lattice point. To find the closest lattice point, the ML detector selects that particular lattice point for which the Euclidean distance is minimum, i.e.,

$$\hat{\mathbf{x}}_{\mathrm{ML}} = \arg \min_{\mathbf{x} \in \mathbf{\Omega}^M} \| \mathbf{y} - \mathbf{Hx} \|_2^2 . \tag{2}$$

ML detection algorithm is complex to implement as hardware. Therefore, some suboptimal detectors like LMMSE or zero-forcing (ZF) are used instead of the optimal detector.

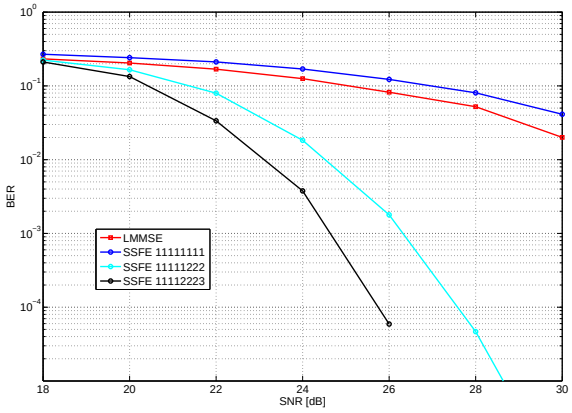

Fig. 4. Detector performance in a highly correlated channel.

However, generally, these linear detectors do not perform close to the optimal ML detectors. Equation (2) can also be solved by constructing a spanning-tree. The spanning tree consists of N+1 levels and in each level the several child nodes come out from the father node depending on the constellation size. Different classes of tree search algorithms are proposed in [3] [10] performing near to the optimal detectors but are more complex than the LMMSE or ZF detectors.

A. Linear Minimum Mean Square Error

The LMMSE detector minimizes the mean square error between the transmitted signal vector $\mathbf{x}$ and the soft output vector $\tilde{\mathbf{x}}$. The LMMSE detector can be calculated as

$$\mathbf{W} = (\mathbf{H}^{\mathbf{H}}\mathbf{H} + \sigma^2 \mathbf{I}_M)^{-1}\mathbf{H}^{\mathbf{H}}. \tag{3}$$

where $\mathbf{H} \in \mathbb{C}^{M \times N}$ denotes the channel matrix, σ^2 denotes noise variance and $\mathbf{I}_M$ is the $M \times M$ identity matrix. The output of the LMMSE detector can be calculated as

$$\tilde{\mathbf{x}} = \mathbf{Wy}. \tag{4}$$

The inversion required in the above equation becomes quite complex for higher number of antennas. Typically, the channel matrix $\mathbf{H}$ is QR decomposed into two parts as $\mathbf{H} = \mathbf{QR}$. Here $\mathbf{Q} \in \mathbb{C}^{(M \times M)}$ denotes a unitary matrix and $\mathbf{R} \in \mathbb{C}^{(M \times M)}$ denotes an upper triangular matrix. The QR decomposition is necessary for the tree search detection algorithms and the inversion of an upper triangular matrix is simpler than a dense matrix.

The QR decomposition is slightly modified for the LMMSE filter. The additive noise is taken into account by considering an extended channel matrix $\underline{\mathbf{H}}$ [11].

$$\underline{\mathbf{H}} = \begin{bmatrix} \mathbf{H} \\ \sigma \mathbf{I}_N \end{bmatrix} = \underline{\mathbf{QR}} = \begin{bmatrix} \mathbf{Q}_1 \\ \mathbf{Q}_2 \end{bmatrix} \underline{\mathbf{R}} \tag{5}$$

Here, $\underline{\mathbf{Q}} \in \mathbb{C}^{(N+M) \times N}$ matrix is composed of $\mathbf{Q}_1 \in \mathbb{C}^{(M \times N)}$ and $\mathbf{Q}_1 \in \mathbb{C}^{(M \times N)}$. $\underline{\mathbf{R}} \in \mathbb{C}^{N \times N}$ denotes an upper triangular matrix and invertible with less complexity.

The LMMSE detector is then obtained from,

$$\mathbf{W} = \underline{\mathbf{R}}^{-1}\underline{\mathbf{Q}}^{H}. \quad (6)$$

B. Selective Spanning with Fast Enumeration

SSFE is a breadth-first tree-search detection algorithm that has a regular and deterministic dataflow which makes it suitable for programmable architectures. In a traditional tree search algorithm, the number of child nodes that spans each level depends on the constellation size. The most likely candidate nodes are kept and the rest are deleted in each level. Therefore, the sorting and deleting process makes the traditional tree search algorithm complex. The SSFE can be characterized with a spanning vector $\mathbf{m} = [m_1, m_2, ..., m_N]$. The spanning vector indicates the number of child nodes that span from the parent node in each level. On the other hand the nodes are never deleted. Therefore, the traditional sorting and deletion is not present in the SSFE algorithm, which reduces the algorithm complexity.

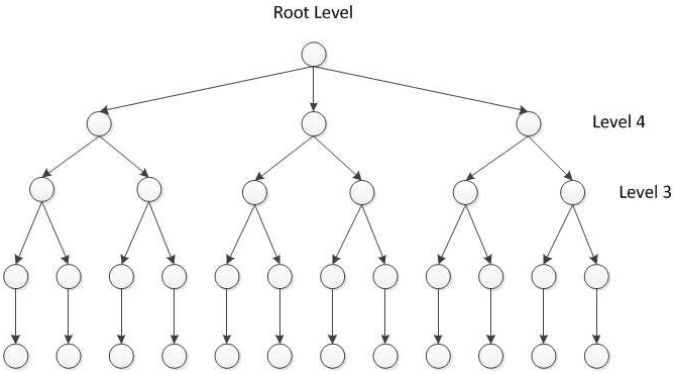

Fig. 5. Topology of the SSFE algorithm.

The topology of the SSFE search tree for a level update vector [1223] is shown in Fig. 5. The level update vector indicates that three nodes are coming out from the parent node at the root level. Therefore, level 4 has three candidates. The next value of the level update vector specifies that two child nodes are connected to each parent node from level 4. In the end, there are 12 candidate nodes. A compromise between the complexity and the BER performance can be achieved by carefully choosing the level update vector.

C. Adaptive Detector

The block diagram of the MIMO baseband receiver with the adaptive detector is presented in Fig. 6. The detector needs an estimate of the channel $\mathbf{H}$ which is obtained by the channel estimator. The output of the channel estimator is fed to the detector selection block, which computes either the condition number of the channel or the distribution of the channel correlations, and selects the suitable detection algorithm from these values. The detector selection is updated periodically and the detection algorithm is changed according to the channel condition. In Fig. 6, only the LMMSE and the SSFE detection algorithms are shown as we restricted our attention to these algorithms in this work, but other algorithms can also be employed.

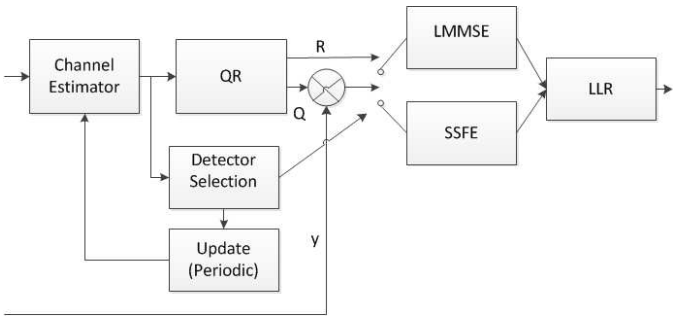

Fig. 6. Block diagram of a receiver with adaptive detector.

IV. TTA Processor for Adaptive Detection

A fixed point TTA processor is designed to support the LMMSE and the three variants of SSFE algorithm. The QR decomposition and the detector selection logic implementation is not considered in this work. A part of the processor designed is illustrated in Fig. 7. For readability, the whole processor is not given in the figure. The blocks in the upper part of the figure represent the function units and register files of the processor. The black horizontal straight lines represent the buses of the processor. The vertical rectangular blocks represent the sockets. The connection between function units and buses is illustrated by black spots in the sockets.

The processor includes load/store unit (LSU), arithmetic logic unit (ALU), global control unit (GCU) and register files. Based on the resource requirements in the high level language, more function units and register files are added.

The log-likelihood ratio (LLR) inputs are read from a first-in-first-out (FIFO) memory buffer by using the function unit called STREAM. The STREAM units can read every input sample in one clock cycle. Eight STREAM units are used to get the input LLRs simultaneously. The STREAM units are used to implement the sliding window technique that helps to decode the input block in smaller parts parallely. One STREAM unit is used to write the output LLRs in the memory buffer.

A single cycle special function unit (SFU) slicer is designed to accelerate the program execution. The slicer compares two values and returns constant values defined by the modulation order. The designed slicer SFU takes two inputs which are the value needed to be sliced and the number of nodes. The slicer has three outputs indicating that three best symbol candidates will be returned. In the real valued signal model, 16-QAM and 64-QAM have four and eight symbol candidates respectively, but the level update vector used in this work restricts this to a maximum of three candidates.

One LSU unit is used to support the memory accesses. The LSU units are used to read and write memory. The memory can be read in three clock cycles and can be written in a single cycle. The ALU unit is used to perform the basic

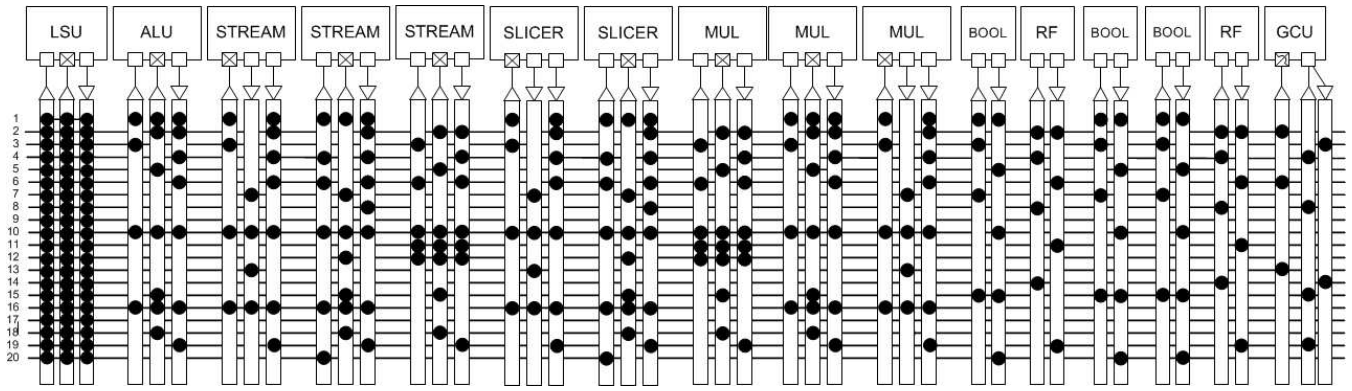

Fig. 7. Implemented processor with reduced number of function units.

arithmetic operations like addition, subtraction etc. Operations like shifting right or left are also included in the ALU.

Twenty five buses are used in the design. Several register files are used to save the intermediate results. In terms of power consumption, registers can be more expensive than the memory, but to meet the latency requirements register files are needed. A single Boolean register file is included in the processor design.

The processor is programmed with a high level language C. Macros are used to call the special function units. SSFE with m = [11111111] is written without any loop in C. In each level only one symbol candidate and the euclidian distance is calculated. The only difference between the SSFE variants is the amount of input data and multiplications increase when the tree goes to the next level. Therefore, the data needed for the next level is read beforehand in an earlier stage when the STREAM unit is idle. For example, The value of the $R(8,8)$ is needed in the first stage and the values of $R(7,7)$ and $R(7,8)$ are needed in the next stage. Therefore, after reading the $R(8,8)$ needed for the first stage, the next values are being read with the same STREAM unit while the other operations are being executed in parallel. In each level only one slicing operation is needed. Therefore, no parallel operation of slicer is needed in this mode.

In case of SSFE with m = [11111222] and m = [11112223] the input data is read in the same way. Several slicing operations are done in parallel with different SFUs using marcos. Some of the loops are unrolled to avoid unnecessary calculations.

It can be seen from (5) that LMMSE requires a matrix inversion and matrix multiplication. The inversion of the upper triangular matrix, $\mathbf{R}$ is written in C following the algorithm proposed in [12]. The nested loops are converted to single loops so that the compiler can easily parallelize the operations. The slicer is not used in the case of LMMSE.

V. Results and Discussion

The designed processor takes 97 clock cycles to process one symbol vector for 4×4 antenna configuration and 64 QAM for SSFE with level update vector [11111111]. The throughput for the SSFE with a clock frequency of 200 MHz can be calculated as,

$$\text{Throughput} = \frac{4 \times 6 \times 200 \text{ MHz}}{97 \text{ clock cycles}} = 49.48 \text{ Mbps}.$$

TABLE II
Number of clock cycles for a single iteration with three simulation blocks

Modes	Algorithm	Clock Cycle	Throughput
1	SSFE [11111111]	97	49.48 Mbps
2	LMMSE	203	23.64 Mbps
3	SSFE [11111222]	408	11.78 Mbps
4	SSFE [11111223]	982	4.88 Mbps

It can be seen from Table II that the processor takes more clock cycles for SSFE [11111222] and [11112223]. Theoretically, these two modes need a lot more calculation than LMMSE or SSFE [1111111]. For the SSFE methods, the data dependency in each level increases the number of clock cycles. Some of the operations during the algorithm execution are summarized in Table III.

TABLE III
Number of operations

Operation	[11111111]	LMMSE	[11111222]	[11112223]
ADD	28	173	318	544
SLICER	8	0	47	69
MUL	52	140	363	612
STREAM	91	91	91	91
LDW	7	84	255	138
STW	8	65	72	184

The number of addition operations does not only include the additions for the algorithm, but for several other purposes like loop indexing for the code. The number of multiplications is high for SSFE [11111222] and SSFE [11112223] as the number of symbol candidates are also higher in these modes. The higher number of memory accesses cannot be avoided with a processor that uses simple function units because the LSU has to continuously access the $\mathbf{R}$ matrix or $\mathbf{y}$ vector.

The euclidian distance calculation is also a recursive process. Hence, the earlier euclidian distance needs to be stored in the memory also.

A comparison with different other implementations is presented in Table IV. The results are normalized for the clock frequency of 200 MHz. In [3] and [13] the results are presented for a 2×2 antenna configuration based on a software programmable solutions. As such, their throughput is lower than the implementations presented in this work. The implementation presented in [14] provides a higher throughput than this work. However, the architecture is fully optimized for that particular level update vector only. Furthermore, the processor is programmed with assembly language. The work presented in this paper provides comprehensive results with different level update vectors for 4×4 with high throughput. In addition, the architecture can support LMMSE to meet the requirements of an adaptive detector.

The processor is designed with simple function units like adders and multipliers which can be used for any algorithm. The slicer unit is used for SSFE only and cannot be used by other algorithms. A more general SFU for LMMSE and SSFU can be implemented to accelerate both algorithms.

TABLE IV
IMPLEMENTATION COMPARISON

Reference	Architecture	m vector	Throughput (norm.)
[3]	TMS320C6416	[1,1,1,1]	25.05 Mbps
[13]	GPU	[1,1,4,4]	11.09 Mbps
[14]	TTA	[1,1,1,1,1,2,2,2]	48.5 Mbps
Proposed	TTA	[1,1,1,1,1,1,1,1]	49.48 Mbps
Proposed	TTA	[1,1,1,1,1,2,2,2]	11.78 Mbps

The detector selection can be done following the simulation results of Figs. 2 - 4 and Table II. The detector selection unit of the adaptive detector can choose LMMSE over SSFE [11111222] for an uncorrelated channel. In case of a moderately correlated channel and 16 QAM configuration, SSFE [11111222] can be chosen over SSFE [11112223] as they provide nearly the same BER performance. The same can be said about LMMSE and SSFE [11111111]. For 64 QAM, the difference between SSFE [11111222] and SSFE [11112223] is nearly 2 dB. Therefore, depending on the BER requirement either of the modes can be chosen.

VI. CONCLUSION

This paper describes a novel design of an adaptive detector on a TTA processor using high level language. The adaptive detector implementation is necessary for the physical layer of the cognitive radio systems. Different detectors are simulated on a Matlab link level simulator. The design is then converted in C language and mapped on the TTA processor using the TCE tool. The processor provides more flexibility than most of the other detector implementations available. It is possible to reach the LTE target throughput with deep deep submicron technologies (DDSM), i.e. 65 nm or below. The target throughput could also be reached by multi-core TTA processor. The energy efficiency of the processor would also provide interesting result.

VII. ACKNOWLEDGEMENT

This research was supported by the Finnish Funding Agency for Technology and Innovation (Tekes), Broadcom Communications Finland, Nokia Solutions and Networks (NSN) and Xilinx.

REFERENCES

[1] D. Cabric, S. M. Mishra, and R. W. Brodersen, "Implementation issues in spectrum sensing for cognitive radios," in *38th Anual Asilomar Conference on Signals, Systems and computers*, November 2004.

[2] E. N. Onggosanusi, A. G. Dabak, and S. Hosur, "Multimode Detection," U.S. 2006/0018410 A1 Patent, January 2006.

[3] M. Li *et al.*, "Selective spanning with fast enumeration: A near maximum-likelihood MIMO detetor designed for parallel programmable baseband architectures," in *International Conference of Communications*, 2008.

[4] A. Burg, S. Haene, D. Perels, P. Luethi, N. Felber, and W. Fichtner, "Algorithm and VLSI architecture for linear MMSE detection in MIMO-OFDM systems," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2006.

[5] J. Niskanen, J. Janhunen, and M. Juntti, "Selective spanning with fast enumeration detector implementation reaching LTE requirements," in *18th European Signal Processing Conference (EUSIPCO)*, Aalborg, Denmark, August 2010.

[6] H. Corporaal, "Microprocessor architectures: from VLIW to TTA", J. Wiley, 1998.

[7] P. Jääskeläinen, V. Guzma, A. Cilio, T. Pitkänen, and J. Takala, "Codesign toolset for application-specific instruction-set processors," in *Multimedia on Mobile Devices 2007*, vol. 6507 of *Proceedings of SPIE* pp. 1-11, San Jose, Calif, USA, January 2007.

[8] O. Esko, P. Jääskelääinen, P. Huerta, C. S. L. Lama, J. Takala, and J. I. Martinez, "Customized Exposed Datapath Soft-Core Design Flow with Compiler Support", in *20th International Conference on Field Programmable Logic and Applications*, Milano, Italy, August-September, 2010.

[9] E. Suikkanen, J. Janhunen, S. Shahabuddin, and M. Juntti, "Study of Adaptive Detection for MIMO-OFDM systems", in *2013 International Symposium on System on Chip (SoC)*, Tampere, Finland, 23-24 Oct. 2013.

[10] Z. Guo and P. Nilsson, "Algorithm and implementation of the k-best sphere decoding for MIMO detection", in *IEEE Journal of Selected Areas of Communications*, vol. 24, no. 3, pp. 491 - 503, March 2006.

[11] D. Wubben, R. Bohnke, V. Kühn, and K. D. Kammeyer, "MMSE extension of V-BLAST based on sorted QR decomposition", in *IEEE 58th Vehicular Technology Conference*, Orlando, Florida, USA, Oct. 2003.

[12] A. El-Amawy and K. R. Dharmarajan, "Parallel VLSI algorithm for stable inversion of dense matrices", in *Computers and Digital Techniques, IEE Proceedings E*, vol. 136, no. 6, pp. 575 - 580, Nov. 1989.

[13] T. Nyländen, J. Janhunen, O. Silvén, and M. Juntti, "A GPU implementation for two MIMO-OFDM detectors", in *2010 International Conference on Embedded Computer Systems (SAMOS)*, Samos, Greece, Jul. 2010.

[14] J. Janhunen, T. Pitkanen, O. Silvén, and M. Juntti, "Fixed- and floating-point processor comparison for MIMO-OFDM detector", in *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 8, pp. 1588 - 1598, Dec. 2011.