ISBN 978-981-11-0008-6
Proceedings of 2016 6th International Workshop on Computer Science and Engineering (WCSE 2016)
Tokyo, Japan, 17-19 June, 2016, pp. 103-109
doi: 10.18178/wcse.2016.06.018

The Design of an Emulation Framework for Variable Run-Time MIMO Detection Systems Based on FPGA Platform

Tzu-Ting Tseng, Wei-En Liang, and Chung-An Shen[+]

Department of Electronic and Computer Engineering, National Taiwan University of Science and Technology, Taiwan

Abstract. This paper presents an emulation framework, including software simulation model and FPGA-based hardware platform, to investigate the size of buffering memories with the complexity and signal optimality of variable run-time MIMO detection system. This framework can evaluate the practical BER performance and system complexity with different sizes of buffering memories. Furthermore, the emulation platform based on the Xilinx FPGA is designed and implemented to provide more close-to-reality and insightful information. Experiments results show that this proposed framework can be used as the tool to facilitate the design decision about the buffering memory and to elucidate the trade-off between complexity and signal optimality.

Keywords: MIMO detection, variable, run-time, FPGA.

1. Introduction

The Multiple-Input Multiple-Output (MIMO) technology, where multiple antennas are employed at both sides of a wireless link, has been recognized as a promising scheme for modern wireless systems [1]-[2]. In the MIMO system, the MIMO detector is deployed at the receiver to recover transmitted symbols from the received signal and results in high system complexity, mainly due to its involvement with numerous matrices and vectors operations. Furthermore, the Sphere Detection Algorithm (SDA) and its variants [3]-[8] have been adopted as the detector of choice, because of its ability to realize (close-to) optimal Maximum-Likelihood (ML) Bit Error Rate (BER) performance with reduced complexity [4]. The SDA scheme can be effectively expressed as a class of tree-searching algorithms finding the shortest path in a tree structure. In general, tree-searching approaches can be classified as variable run-time [3]-[6] and fixed run-time [7]-[8], where the former method leads to a run-time, i.e., processing throughput, that is dependent on the channel condition, while the later one result in a desirable constant processing throughput no matter of the channel conditions. However, in order to maintain a fixed run-time, system-level efficiencies have to be sacrificed. For example, the fixed run-time K-Best detectors [7]-[8] result in higher complexity and lower throughput compared with the variable run-time schemes [3]-[6].

From the system design perspective, the variable run-time schemes are problematic mainly because it arises difficulties to architect a robust and efficient buffering memory scheme. To be specific, a buffering memory is typically used at the receiver of the MIMO system to store received symbols that are waiting to be processed by the detector and the size of this memory is determined based on the symbol arrival rate as well as on the throughput of the MIMO detector. This memory needs to be sufficiently large so that all (or most) arriving symbols can be accommodated, as the symbol drop degrades the signal quality. However, on the other hand, the memory sub-system could be over designed with excessively large size and causes a waste of resource and results in an increment of system complexity. As a result, if the processing engine contains an un-deterministic throughput, it would be extremely difficult to make the design decision about the size of the

[+] Corresponding author. Tel.: +886-2-27303275.
E-mail address: cashen@mail.ntust.edu.tw.

buffering memory so that a balance between the desired signal quality and system complexity can be achieved.

This paper presents an emulation framework, including software simulation and FPGA-based hardware platform, to investigate the size of buffering memories with the complexity and BER performance for variable run-time MIMO detection systems. A simulation model is implemented to study the BER performance and system complexity with different sizes of buffering memories. More importantly, in order to provide more close-to-reality and insightful evaluations of system performances, a practical emulation platform based on the Xilinx FPGA has been designed and implemented. This platform can provide information about the system complexity in terms of practical hardware utilization and can measure realistic BER index. This proposed framework can be served as the tool to facilitate the design decision about the size of the buffering memory and to elucidate the trade-off between system complexity and signal optimality.

2. Background

Figure 1 presents a simplified MIMO system model with a variable run-time tree-searching MIMO detection scheme and a front-end buffering memory. The run-time variability is resulted from the characteristic of the algorithm where the number of searched nodes is affected by the condition of the wireless link. When the channel condition is good, the algorithm can reach the (close-to) optimal result by only searching a small amount of tree nodes, while, on the other hand, a large part of the tree nodes are visited in an ill-conditioned environment. This run-time variability results in a highly undesired un-deterministic processing throughput due to the difficulties of designing memory management schemes. For example, at the scenario where the processing throughput is heavily degraded due to the adverse channel, the incoming symbols will be accumulated in the buffering memory waiting to be processed. In this case, if the size of the memory is not large enough, stored symbols might need to be dropped from the memory (memory overflow) and the signal quality would be severely degraded. On the other hand, in order to avoid excessive symbol drops, the size of the memory needs to be sufficiently large. However, an over-design memory could significantly increase complexity of the system. Nowadays, embedded memories are predicted to consume approximately 50% of power consumption and area for modern System on Chip (SOC) [9]. An overwhelmingly large buffering memory by all means worsen the situation.

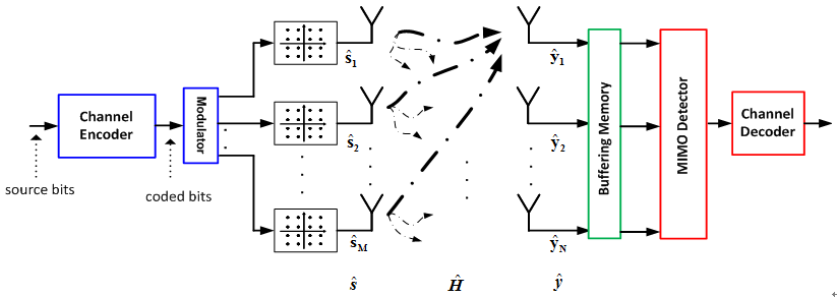


Fig. 1: Simplified system model of MIMO systems.

To address the aforementioned dilemma of complexity versus optimality, a practical and realistic emulation framework is inevitable. This framework needs to emulate a variable run-time MIMO detection system in order to investigate the impact of buffering memory to the system performance. Specifically, this platform needs to measure the BER performance jointly considering MIMO detection as well as symbol drop due to memory overflow. More importantly, this platform needs to include hardware system so that a practical hardware complexity can be elucidated. In short, this emulation framework needs to provide insightful and realistic evaluation results for the system designer to jointly design the MIMO detector and buffering memory scheme.

3. The Proposed Emulation Framework

3.1. The Simulation Model

The initial stage of the proposed framework is to implement a simulation model that can evaluate, as close to the reality as possible, the behavior of a practical variable run-time MIMO detection system and can be served as a prototype to later design the practical emulation platform. Several parameters are defined to clearly reflect the operation of the system. Firstly, the number of searched nodes of the tree-searching MIMO detector is used to characterize the run-time of the MIMO detector and is a variable dependent on the channel conditions. For example, assuming a 4×4 MIMO system with 64-QAM modulation, Figure 2 shows the variability of the algorithm [3] in terms of Cumulative Distribution Function (CDF) of number of searched nodes. It can be seen from Fig. 2 that the number of searched node illustrates a stochastic manner with the average signal-to-noise ratio, Eb/No. Furthermore, the number of clock cycles for visiting one node is architecture-dependent and could be different for various design choices and structures. For example, the architecture presented in [3] achieves a one-node-per-cycle scheme. The combination of these two parameters indicates the number of consumed cycles to search the tree and represents the processing throughput of the MIMO detector. Moreover, the size of the buffering memory denotes the number of received symbols that can be stored and the symbol arrival rate depicts the frequency that the symbols arrive at the buffering memory.

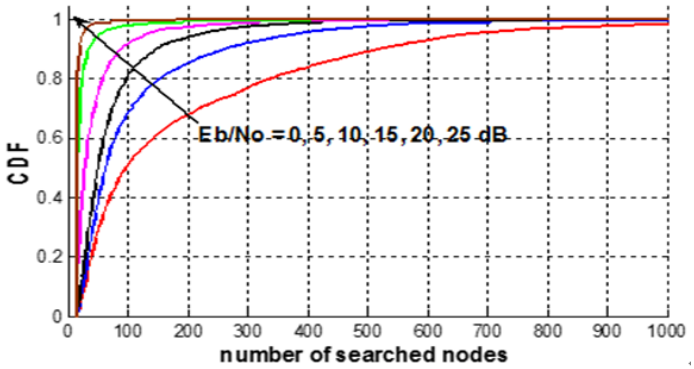


Fig. 2: CDF (Cumulative Distribution Function) of number of searched nodes of the variable run-time detector presented in [3].

In addition, we clearly define the Figure of Merit (FOM) to benchmark the performance of the system. In particular, the optimality of the signal is measured using the BER. Traditionally, when studying the BER of MIMO detectors, it is usually assumed that the detector can execute with infinite time and no memory overflow is taken into consideration. In other words, the BER comes solely from detection errors. However, in the practical system, the run-time constraint and memory overflow might occur and result in the drop of symbol and degradation of signal. Therefore, unlike the conventional definition of BER, in this work the BER performance is comprised of the conventional detection error as well as the errors due to memory overflow and is expressed as:

$$BER = BER_{det\_error} + BER_{drop} \quad (1)$$

For example, the BER performances with different sizes of buffering memories, Qmax, are simulated and summarized in Fig. 3. The simulation setup is a 4×4 MIMO system with 64-QAM modulation and the detector is implemented using the algorithm shown in [3]. It is shown in Fig. 3 that the BER performance drops when the size of the buffering memory is reduced. This is because that the BERdrop term of Eq. (1) increases when the buffering memory is not large enough and the symbol drop occurs frequently. It is also shown in Fig. 3 that when the memory size is sufficiently large, the BER performance is almost on top of the optimal curve, where only the BERdet_error term attributes to the BER performance.

Similarly, the evaluation of complexity also includes the complexity of buffering memory the complexity of the MIMO detector. As the complexity of the MIMO detector is dependent on its architecture and can be referred to the literature [3]-[8], in this work we mainly focus on the complexity of the buffering memory. In the following we will evaluate the complexity of the buffering memory based upon the FPGA platform.

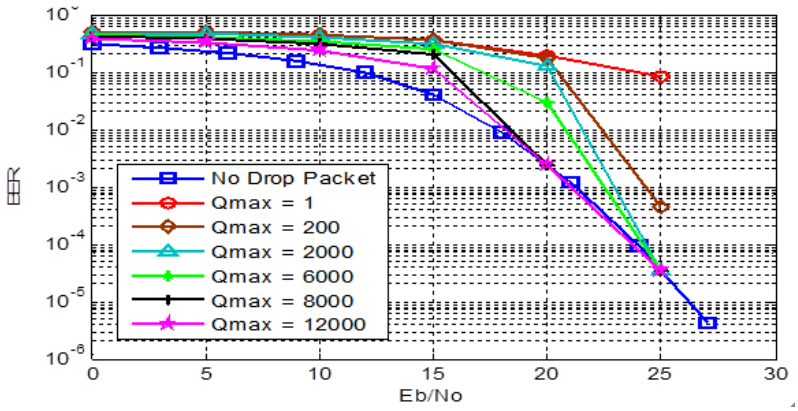


Fig. 3: BER performances with various buffering sizes assuming a 4×4 MIMO system with the 64-QAM modulation scheme and detector of [3].

3.2. Overview of the FPGA-Based Emulation Platform

The aim of the emulation platform is to practically evaluate the performance and measure the complexity of the run-time variable MIMO detection scheme with buffering memories. The architectural overview of the proposed platform is presented in Fig. 4, which includes software components as well as hardware sub-systems based on the Spartan6 XC6SLX150T FPGA. For the software part, the PC running Matlab code plays the transmitter role sending symbols in a fixed data rate to the detector which is implemented based on the FPGA. Furthermore, the PC is connected to the FPGA board through USB interfaces and two FIFOs of 4096×16 bits that are implemented on the FPGA and served as interfaces between the software and hardware. Specifically, the In_FIFO is used for writing data from PC to the detector and the Out_FIFO is for reading data from the detector to PC. In addition, the transmission rate between the transmitter and receiver of Fig. 4 is determined as follows. We first define the targeted processing throughput Thput, in terms of bits-per-second (bps), of the MIMO detector as follows

$$Thput = f \times S_b \times {}^{1}/_{AI} \tag{2}$$

where S_b denotes bits per symbol, f is the operating frequency of the detector system, and AI represents the data arrival interval between the PC the and detector, in terms of cycles per symbol. For example, assuming the detector system is operating at the frequency of 100MHz (f = 100MHz) and the detector can visit one tree node at every clock cycle [3], in order to support a processing throughput of 100Mbps (*Thput* = 100Mbps) for the MIMO system with 4×4, 64-QAM configuration, the data arrival interval is set to be 24 (*AI* = 24). In other words, the PC is statically send a symbol to the detector every 24 cycles.

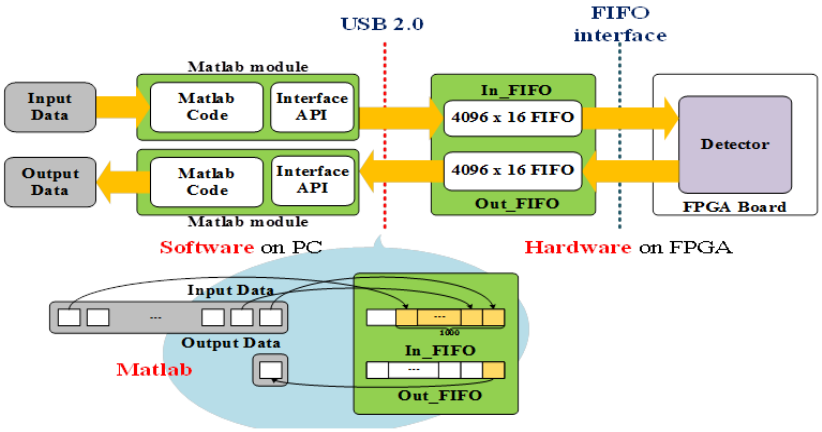
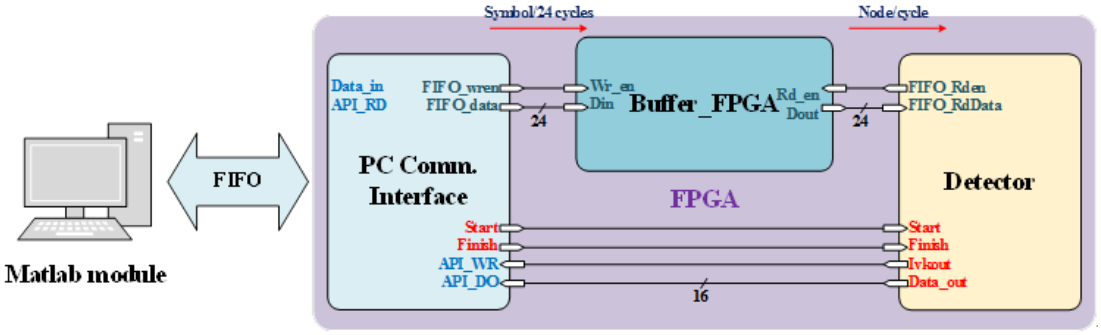


Fig. 4: The overview of the proposed emulation platform.

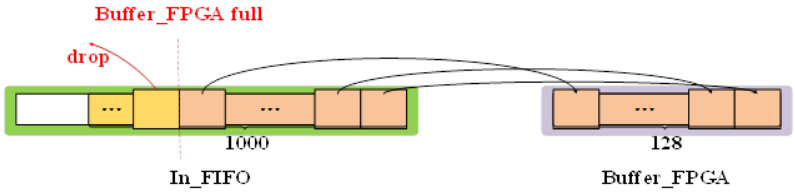
3.3. Hardware Sub-system on the FPGA

The architecture of the hardware sub-system of the proposed emulation platform is further presented in Fig. 5(a). This sub-system is implemented based on the FPGA board and contains three modules, PC

Communication Interface, Buffer, and the MIMO detector. The PC Communication Interface is designed as the communication interface between the PC and the FPGA. It receives input data from the PC to Data_in port on the FPGA through the In_FIFO channel. When the first data is received, the Start flag in the PC Communication Interface is invoked to notice the detector ready for computing. The Buffer_FPGA emulates the front-end buffering memory of the detector and receives the input data with a period of 24 cycles. The detector will request data for processing from this Buffer_FPGA block and execute a depth-first tree searching operation [3] to visit the tree node at every clock cycle. After all the data is transmitted from PC, the Finish flag will be invoked to notify the detector to be ready for finishing the computation of the rest data stored in Buffer_FPGA. In addition, as mentioned previously, the detector might be occupied for a long time by a symbol during an ill channel condition. In this situation, the buffering memory could experience a memory overflow. The idea of overflow for the buffering memory, i.e., Buffer_FPGA in this framework, is shown in Fig. 5(b), where the incoming symbols that are waiting in the In_FIFO will be abandoned.



(a)



(b)

Fig. 5: (a) The hardware architecture of the platform on the FPGA and (b) the illustration of symbol drop due to the overflow of Buffer_FPGA.

3.4. Operation Flow of the Emulation Platform

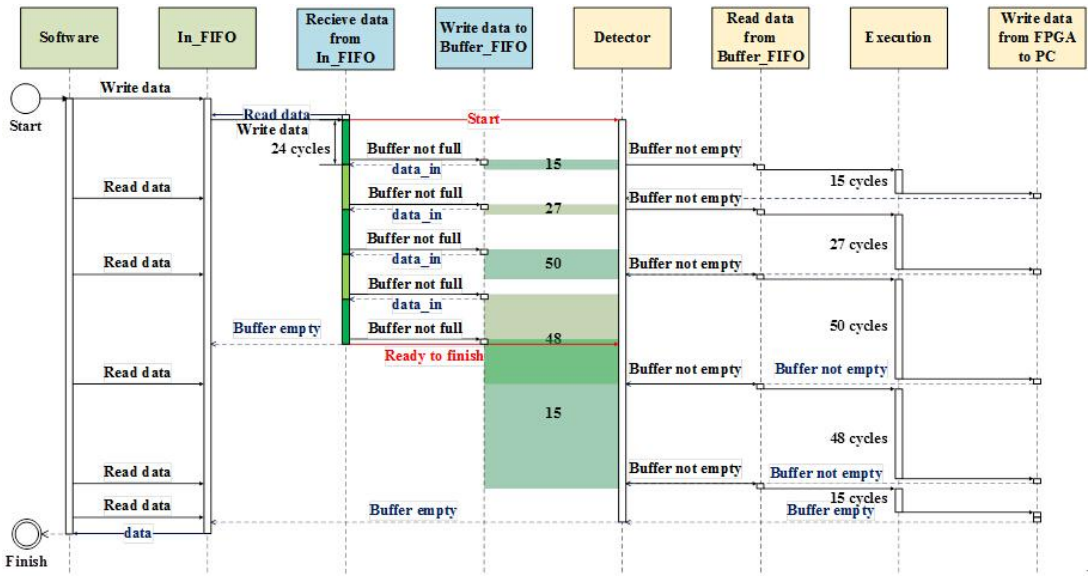


Fig. 6: The operation flow of the proposed emulation framework.

Figure 6 further presents the operation flow of the proposed emulation framework, whose architecture has been illustrated in Fig. 4 and Fig. 5. At the beginning, the PC will send symbols to the In_FIFO continuously until the In_FIFO is full. Each symbol contains important information that denotes the required visited nodes, i.e., cycles, to process that symbol. The operation flow shown in Fig. 6 represents an example of five symbols, where the required cycles are 15, 27, 50, 48 and 15. Once the In_FIFO is filled, the PC Communication Interface module starts to receive input symbols from In_FIFO with a period of 24 cycles and saves them into the Buffer_FPGA block. As a hand-shaking protocol, the Start flag is invoked when the first data begins to receive from FPGA and the Finish flag is invoked when there's no data exists in the In_FIFO. In the following, the detector starts to enumerate the tree in a one-node-per-cycle manner and returns the detected results once a symbol is finished. The detector stop reading data from FIFO_FPGA when the Finish flag is invoke. It is noted that sending symbols from the PC to the In_FIFO and receiving detected results from the out_FIFO are executed in a parallel fashion.

4. Experimental Results and Complexity Analysis

In this section will present the complexity of the hardware sub-system of the framework and illustrate the trade-offs between system complexity and drop rate of the symbol with different Buffer_FPGA sizes. The hardware sub-system is implemented using the Xilinx Spartan6 XC6SLX150T FPGA and the system throughput is set to be 100Mbps. Furthermore, the drop rate is calculated as the portion of symbol drop during the process of symbol transmission, i.e., symbol drop/total symbol. Table I presents the experimental results for symbol drop rates with different sizes of Buffer_FPGA. It can be shown in Table I that the drop rate decreases with the size of the buffering memory, due to the lower possibility of memory overflow, while the system complexity increases significantly. This clearly illustrates the design trade-offs between signal quality versus system complexity. Moreover, the drop rate versus hardware utilization in terms of Slice registers and LUTs documented in Table I and in Fig. 7. In addition, the size of FIFO is annotated in Fig. 7. It is shown in Table I and Fig. 7 that the employment of the buffer with size 24×512 bits can achieve 0 drop rate at the cost of the highest complexity. In addition, it can be observed from Fig. 7 that the number of used slice registers is almost decreases linearly with the drop rate. However, as shown in Fig. 7, the slopes between the number of used slice LUTs and the drop rate is dramatically increased when the buffer size is extreme large or extreme small. Therefore, the experimental results provide insightful information for the system designer to architect a MIMO detector system with the optimal balance between the signal quality and the system complexity. This platform can be scaled to evaluate different configurations of MIMO systems with variable run-time detectors and buffering memories.

Table I. Symbol Drop Rate with Different Sizes of Buffering Memories

FIFO(bits) / Utilization	1.536k (24×64)	3.072k (24×128)	6.144k (24×256)	12.288k (24×512)
Drop rate	0.243	0.184	0.061	0
No. of Slice Registers	79	85	97	103
No. of Slice LUTs	67	69	78	102
No. of BRAM	1 (9k)	1 (9k)	1 (9k)	1 (18k)

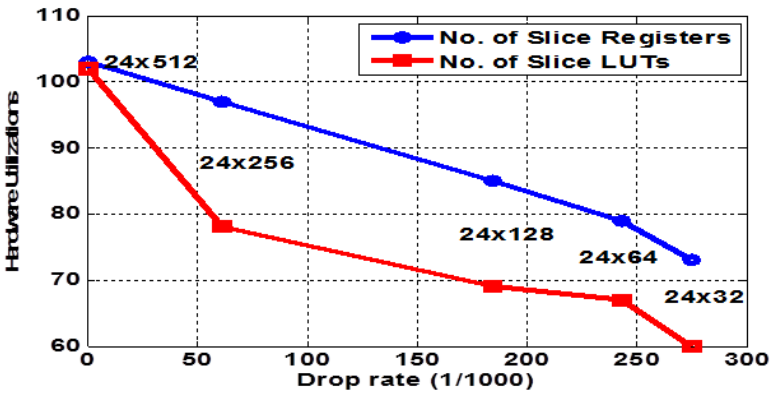


Fig. 7: Drop rates versus hardware utilizations.

5. Conclusion

An emulation framework has been designed and implemented for investigating the size of buffering memories on the complexity and BER performance for variable run-time MIMO detection schemes. The revised BER jointly considering symbol drop and detection error has been evaluated using this framework. Furthermore, the design trade-off between the complexities, illustrated using hardware on the FPGA, and signal optimality has been presented in this paper. Experiments results have shown that this proposed framework can be used as the tool to facilitate the design decision about the buffering memory for the variable run-time MIMO detector systems.

6. References

[1] D. Gesbert, M. Shafi, Da-Shan Shiu, P. J. Smith, and A. Naguib, "From theory to practice: An overview of MIMO space-time coded wireless systems," *IEEE Jour. Selected Areas in Comm.*, vol. 21, no. 3, pp. 281-302, Apr. 2003.

[2] E. G. Larsson, F. Tufvesson, O. Edfors, and T. L. Marzetta, "Massive MIMO for next generation wireless systems," *IEEE Comm. Magazine*, vol. 52, no. 2, pp. 186-195, Feb. 2014.

[3] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bolcskei, "VLSI implementation of MIMO detection using the sphere decoding," *IEEE Jour. Solid-State Circuits*, vol. 40, no. 7, pp. 1566-1577, Jul 2005.

[4] C.-A. Shen and A. M. Eltawil, "A Radius Adaptive K-Best Decoder With Early Termination: Algorithm and VLSI Architecture," *IEEE Trans. Circuits Syst. I*, vol. 57, no. 9, pp. 2476-2486, Sep 2010.

[5] C. H. Yang and D. Markovic, "A flexible DSP architecture for MIMO sphere decoding," *IEEE Trans. Circuits Syst. - I*, vol. 56, no. 10, pp. 2301-2314, Oct. 2009.

[6] C.-A. Shen, A. M. Eltawil, S. Mondal, and K. N. Salama, " A Best-First Soft/Hard Decision Tree Searching MIMO Decoder for a 4 x 4 64-QAM System," *IEEE Trans. VLSI Systems*, vol. 20, no. 8, pp. 1537-1541, Aug. 2012.

[7] L. G. Barbero and J. S. Thompson, "Fixing the Complexity of the Sphere Decoder for MIMO Detection", *IEEE Trans. on Wireless Comm.*, Vol 7, No 6, pp 2131 – 2142, June 2008.

[8] M. Shabany and P. G. Gulak, "A 675 Mbps, 4 x 4 64-QAM K-Best MIMO Detector in 0.13μm CMOS," *IEEE Trans. VLSI Systems*, vol. 20, no. 1, pp. 135-147, Jan 2012.

[9] F. Clermidy, C. Bernard, R. Lemaire, J. Martin, I. Miro-Panades, Y. Thonnart, P. Vivet, and N. When, "A 477 mW NOC-based digital baseband for MIMO 4G SDR," *IEEE Intl. Solid-State Circuits Conference*, pp. 279-279, Oct. 2010.