MIMO Detector Implementations Using High-level Synthesis Tools from Different Generations

Tuomo Hänninen, Muhammad Saad Saud, Hamid Yadegar Amin and Markku Juntti
Centre for Wireless Communications
P.O. Box 4500, FI-90014 University of Oulu, Finland
Email: tuomo.hanninen@oulu.fi, hamid.yadegaramin@oulu.fi, markku.juntti@oulu.fi

***Abstract*—In 2011, we proposed a new receiver structure called *Frequency domain linear MMSE filter with sphere detection* for single-carrier frequency-division multiple access (SC-FDMA) uplink transmission. Frame error rate simulations and complexity estimations were used to define the most practical sphere detector algorithm for this receiver. High-level Synthesis (HLS) tools were used for comparing different architectures for the sphere detector. After 2011, the HLS design approach has gained more popularity and the tools have evolved. In this paper, HLS tools from different generations were used for implementing the same K-best list sphere detector. The results were compared and the overall experience of the optimization process and the evolution of the HLS tools was discussed. Additionally, the evolution of FPGA technology was discussed. In conclusion, the HLS tools have evolved into practical implementation tools even for high complex wireless communication algorithms. Moreover, 25% increase in throughput and 58% lower resource usage was achieved with the latest generation FPGA.**

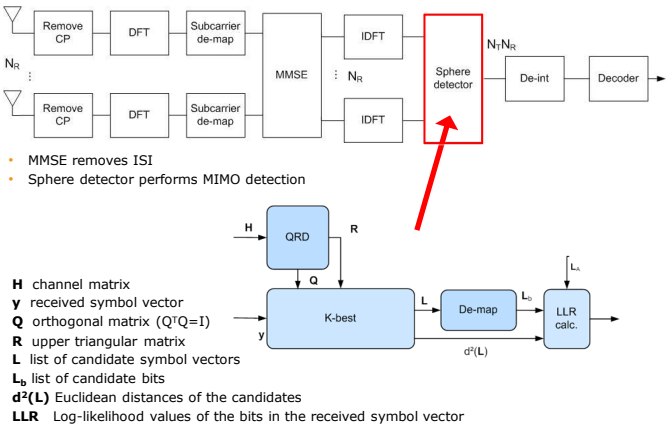

Fig. 1. Receiver with sphere detection.

I. Introduction

Fourth generation (4G) wireless cellular systems [1] use single-carrier frequency-division multiple access (SC-FDMA) as the uplink transmission scheme instead of orthogonal frequency division multiplexing (OFDM) [2]. The reduced peak-to-average power ratio of single-carrier transmission reduces the mobile transmitter cost by allowing cheaper power amplifiers [3]. In single-carrier transmission an equalizer is needed in the receiver to cancel the inter-symbol interference (ISI). Multiple-input multiple-output (MIMO) antenna configuration [4], [5] is used in 4G systems to increase the peak data rates. Similar to any spatial multiplexing based MIMO transmission, a spatial equalizer is required in the receiver to cancel the inter-antenna interference (IAI). Both linear and nonlinear receiver structures have been considered for MIMO receivers with an emphasis on ones operating in OFDM systems, wherein ISI is not a problem. Different variants of *sphere detector* (SD) [6], which calculate the maximum likelihood (ML) solution with reduced complexity, have received a lot of attention in the literature. For systems employing forward error control (FEC) coding, the soft output *list sphere detector* (LSD) [7] is needed.

In 2011, we proposed a new receiver structure *Frequency domain linear MMSE filter with sphere detection* for SC-FDMA uplink transmission and compared this to conventional *frequency domain linear MMSE equalization with soft demodulation* receiver structure [8]. Two different tree search algorithms were considered for our receiver, namely the K-best list LSD algorithm [9] and the selective spanning with fast enumeration (SSFE) algorithm [10]. The receiver structure and sphere detector are illustrated in Figure 1. In [11], we implemented the K-best LSD algorithm on a field-programmable gate array (FPGA) using an early version of Xilinx Vivado High-level Synthesis (HLS) tool to get a good understanding of its complexity. The HLS tool enabled the implementation and comparison of different macroarchitectures in a relatively short time. The architecture optimization was done in the C language level, which gave a clear benefit in terms of design time and effort. The area efficiency could have been further optimized with the conventional design approach, but HLS was well suited for this type of architecture evaluation. There are studies showing that the HLS tools can produce competitive quality of results compared to hand written register-transfer level (RTL) abstraction [12]. The HLS tools have been earlier used for MIMO detector FPGA implementations e.g. in [13] and for application-specific integrated circuit (ASIC) implementation e.g. in [14].

In addition to results reported in [11], we have implemented the same LSD algorithm blocks using different HLS tools from a year 2010 until 2017. This paper compares these implementation results and discusses the evolution of HLS tools and FPGA technology. Our first implementations were carried out using the Mentor Graphics Catapult C 2010 HLS

tool. Later we adopted the AutoESL AutoPilot 2011 and currently we are using the latest version of the Xilinx Vivado HLS 2017. In this paper, the same C language input has been used with different tools. The original 40 nm Virtex 6 FPGA is not supported by the latest Xilinx tool anymore. Therefore, the latest implementations are targeted for the 28 nm Virtex 7 FPGA. Initially, the same macro-architecture, parametrization and FPGA optimizations (e.g. bit-width optimization and efficient use of embedded DSP blocks) were used for the implementations. The complexity and throughput results are compared. Additionally, the overall experience of the optimization process and the evolution of the HLS tools are discussed. The portfolio of currently available FPGAs is relatively large. Therefore, in addition to 28 nm FPGA implementation, the Vivado HLS 2017 tool was also used to implement the K-best LSD on the latest 20 nm and 16 nm FPGAs. The benefit of smaller silicon technology is discussed.

The rest of this paper is organized as follows. In Section II, we introduce the short history of the HLS tools used in this paper. Additionally, we compare the synthesis time and overall user experience of the tools. The evaluation of the HLS tools is divided into two sections. Section III discusses the evolution from Catapult C 2010 to AutoPilot 2011. Similarly, the Section IV discusses the evolution from AutoPilot 2011 to Vivado HLS 2017. The effect of FPGA technology as well as the effect of the built-in Vivado HLS 2017 implementation strategies are studied in Section V. The final conclusions are given in Section VI.

II. Evolution of HLS Tools and Synthesis Time

The Behavioral Compiler introduced in 1994 by Synopsys can be considered as the first HLS product. Although, at that time, behavioral synthesis or algorithmic synthesis were more common terms. The real emerging of the HLS tools started in 2004 with the next generation HLS products. These tools accepted the popular C language as an input and enabled the required flexibility and ease of use. Mentor Graphics Catapult C and AutoESL AutoPilot are few examples of the HLS tools of this generation. In late 2010, the AutoESL Design Technologies Inc. was still a small independent HLS vendor. In 2011, Xilinx acquired AutoESL Design Technologies Inc. and the AutoPilot tool name was later changed into Vivado HLS.

The initial tool for our MIMO detector architecture comparison was the 2010 version of the Mentor Graphics Catapult C. However, due to very long synthesis time, the iterative design process was not efficient. The second evaluated tool was the AutoESL AutoPilot. It immediately impressed with remarkably fast synthesis. The disadvantage was the high number of bugs. After Xilinx acquiring the AutoESL, the AutoPilot soon became the Vivado HLS. The first Xilinx release of the tool was improved in the terms of usability. The disadvantage was the fact that other vendors FPGAs were not supported anymore. The latest version of Xilinx Vivado HLS tool is the Vivado HLS 2017. The average synthesis times of 8-best LSD MIMO detector implementation for these three tools are described in Table I. It should be noted that the desktop PC performance used to run the tools was not fully equal. Yet, the results give an overview of the user experience of different HLS tools used in this paper.

TABLE I
8-best LSD synthesis time

	Min	Max
Mentor Graphics Catapult C 2010	4 hours	48 hours
AutoESL AutoPilot 2011	20 min	2h 15 min
Xilinx Vivado HLx 2017	20 min	40 min

III. Catapult C 2010 vs AutoPilot 2011

The Catapult C and AutoPilot synthesis performance in terms of scheduling and resource usage were compared with a smaller design than the K-best LSD due to very long synthesis time of the Catapult C with the 8-best LSD design. Here, the small design was the non-iterative version of the log-likelihood ratio (LLR) calculation block of the 8-best LSD. The development time in the number of iterations vs. resource usage and processing time for 1200 subcarriers using Catapult C and AutoPilot are shown in Figures 2 and 3, respectively.

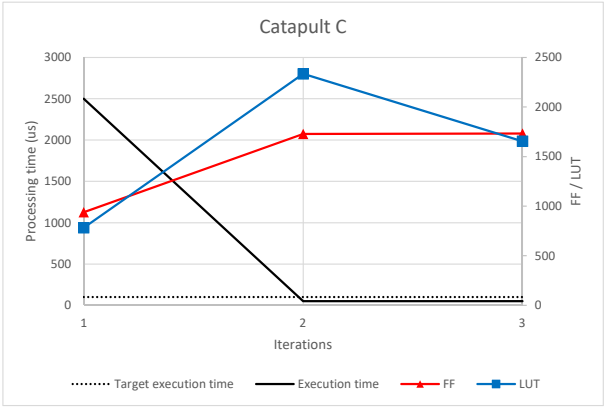

Fig. 2. Processing time and resource usage for LLR – Catapult C 2010

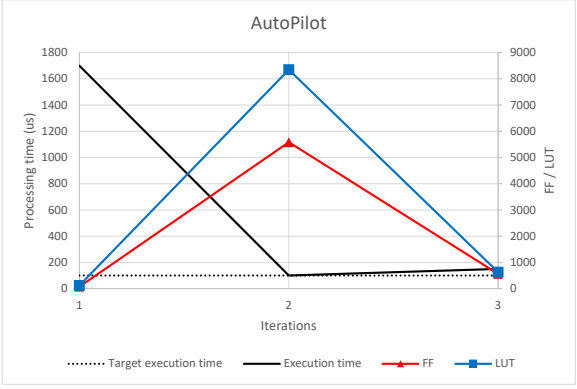

Fig. 3. Processing time and resource usage for LLR – AutoPilot 2011

Tables II and III show the target latencies, achieved latencies and achieved resource usages for the Catapult C and AutoPilot, respectively. No target latency was set for the first iteration. It can be seen that the scheduling is very different with these tools. It is easier to achieve the target latency with the Catapult C. However, the resource usage is higher than that with the AutoPilot. From the user experience point of view, the Catapult C had a better schedule viewer and it showed the architecture view of all the loops and directives. Hence, tracing back to C code was easier.

TABLE II
TARGET LATENCY VS ACHIEVED LATENCY - CATAPULT C 2010

Target latency	Achieved latency	FF	LUT
-	625	939	783
16	16	1728	2335
12	1	1733	1655

TABLE III
TARGET LATENCY VS ACHIEVED LATENCY - AUTOPILOT 2011

Target latency	Achieved latency	FF	LUT
-	625	72	120
16	8	5584	8348
24	34	562	630

IV. AUTOPILOT 2011 VS VIVADO HLS 2017

The Catapult C 2010 achieved the target latency accurately if the design was relatively small. However, due to very long synthesis time with the complex 8-best tree search algorithm, the design process was not efficient. Therefore, the implementation of the K-best tree search algorithm was completed using the AutoPilot as described in Section III and reported in [11].

In this paper, Vivado HLS 2017 tool was used to create reference implementation for the AutoPilot implementation. The complexity and throughput results maximizing the throughput are compared in Table IV. With AutoPilot, the 8-best LSD MIMO detector design including the sorting operation did not achieve the target throughput for the 4×4 64-QAM system. Sort-free architecture was exploited to achieve the target throughput of 347 Mbps. The so called sort-free architecture can be exploited to increase the performance of the SD tree search algorithms. However, it often increases the resource usage dramatically especially if the K is not smaller than the number of constellation points. With 2017 version of the Vivado HLS tool the target throughput was exceeded by 44% without the need for using the sort-free architecture.

V. FPGA TECHNOLOGY COMPARISON

Due to latest FPGA technology updates, the portfolio of the currently supported FPGAs by HLS tools is relatively large. In addition to 40 nm and 28 nm FPGA implementations, the K-best LSD MIMO detector was also implemented on the 20 nm and 16 nm FPGAs to get good understanding of the technology influence on the results. Table V describes the implementation

TABLE IV
AUTOPILOT 2011 VS VIVADO HLS 2017

HLS tool	AutoPilot	AutoPilot	Vivado HLS
FPGA	Virtex 6	Virtex 6	Virtex 7
Technology	40nm	40nm	28nm
Architecture	Original	Sort-free	Original
LUT	8619	69383	81445
FF	12511	97676	80742
DSP	54	228	565
BRAM	7	287	0
Frequency [MHz]	247	231	167
Throughput [Mbps]	93	347	501

resuls for the Xilinx high-end Virtex FPGA family. Figure 4 highlights the throughput gains. Xilinx mid-range Kintex FPGA family implementations were also carried out. However, those are not reported herein due to high correlation to Virtex results.

TABLE V
2017 FPGA TECHNOLOGY COMPARISON - VIRTEX FAMILY

	Virtex 7	Virtex UltraSCALE	Virtex Ultra-SCALE+
Technology	28nm	20nm	16nm
LUT	81445	71960	65583
FF	80742	52930	34138
DSP	565	565	565
Freq [MHz]	167	172	208
Throughput [Mbps]	501	516	624
Dynamic power [W]	1.76	0.84	0.9
Energy [nJ/bit]	3.5	1.62	1.45

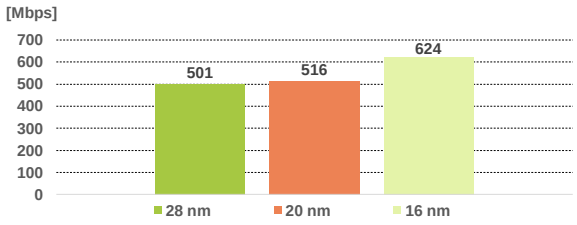

Fig. 4. Throughput [Mbps]

Each of the implementations achieve the target throughput for LTE SC-FDMA 4×4 64-QAM uplink with a conventional K-best architecture. As it is seen in Table. V and Figure 4, the latest FPGAs schedule with higher throughput and less resources. As an instance, the latest 16 nm implementation, has shown to achieve 25% increase in throughput in addition to approximately, 19% saving in LUT usage and 58% saving in FF usage in comparison to one with 28 nm process. This fact suggests that more complex algorithms with higher MIMO configuration orders or higher modulation rates could potentially be implemented with the latest FPGAs without massive parallelism or implementation complexity. In addition to capability of achieving a higher throughput, the more advanced FPGAs have shown a better power performance as well. Table VI demonstrate the power consumption of Virtex Family FPGAs with the throughputs normalized to 501 Mbps.

Figure 5 highlights the significantly higher improvement in dynamic power consumption while moving from 28 nm to 20 nm FPGA compared to step from 20 nm to 16 nm FPGA.

TABLE VI
2017 FPGA TECHNOLOGY COMPARISON - VIRTEX FAMILY (NORMALIZED THROUGHPUT)

	Virtex 7	Virtex UltraSCALE	Virtex UltraSCALE+
Technology	28nm	20nm	16nm
LUT	81445	71922	64369
FF	80742	52930	34138
DSP	565	565	565
Freq [MHz]	167	167	167
Throughput [Mbps]	501	501	501
Power [W]	1.76	0.85	0.75
Energy [nJ/bit]	3.5	1.70	1.48

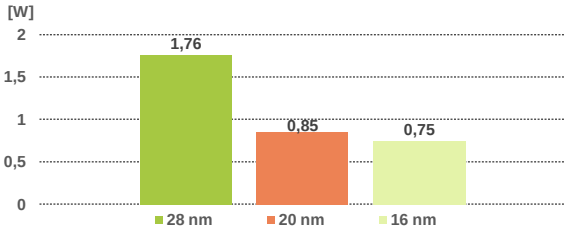

Fig. 5. Dynamic power [W]

In order to improve the throughput of the implementation, Xilinx Vivado provides implementation strategies [15]. Considering the requirement, software tries different optimizations in placing and routing. However, each of them gives a different performance in terms of power. Table VII, presents performance of four different implementation strategies for Virtex 7 to achieve the maximum throughput. The first strategy (A: PerformanceRefinePlacement), increases the placer effort in the post-placement optimization phase and disables timing relaxation in the router. The second strategy (B: PerformanceNetDelaylow), compensates for the optimistic delay estimation and adds extra delay cost to distance and high fanout connections. Third strategy (C: AreaExplore), uses multiple optimization algorithms to get potentially fewer LUTs. The fourth strategy (D: Flow run post rout opt), enables physical optimizations in post implementation phase including routing. The result have shown that in addition to achieve the highest throughput, strategy D manages to perform a better performance in terms of power. Optimization strategy D results are reported in Table VI for Virtex 7. Similar iterative approach was exploited for all the other results.

VI. CONCLUSION

In this paper, the evolution of HLS tools was studied in two steps. The evolution from the early version of Catapult C to AutoPilot enabled significantly shorter synthesis time which is crucial for the iterative HLS design approach. Also lower resource usage was achieved with AutoPilot. The evolution from AutoPilot to the latest version of Vivado HLS has enabled even shorter maximum synthesis time. Moreover, more advanced built-in optimization strategies enabled higher maximum throughput achieving the required 64-QAM 4×4 MIMO LTE/LTE-A detection requirement easily.

TABLE VII
VIVADO HLS 2017 IMPLEMENTATION STRATEGIES

Strategy	A	B	C	D
Technology	28nm	28nm	28nm	28nm
LUT	81450	81446	83105	81445
FF	80742	80742	80742	80742
DSP	565	565	565	565
Freq (MHz)	167	167	167	167
Throughput(Mbps)	501	501	501	501
Power(W)	1.77	1.80	2.27	1.76
Energy(nJ/bit)	3.53	3.60	4.50	3.5

The effect of FPGA technology was also studied. 25% higher throughput and 58% lower FF usage was archived by using the latest 16 nm FPGA over the 28 nm FPGA. Moreover, 57% lower dynamic power consumption was achieved with the 16 nm FPGA. Finally, the built-in Vivado HLS implementation strategies were compared. As a result, 22% lower power consumption was achieved with carefully selected strategy.

In conclusion, the HLS tools have evolved into practical implementation tools even for high complex wireless communication algorithms. While choosing between algorithms with different performance-complexity ratio, it should be noted that few tens of percentage difference in performance or complexity falls within the variation of FPGA technology selection.

REFERENCES

[1] 3rd Generation Partnership Project (3GPP), "Physical layer procedures," 3GPP TS 36.213 V11.0.0, Tech. Rep., 2012.

[2] Z. Zvonar, "Multiuser detection in asynchronous CDMA frequency-selective fading channels," *Wireless Personal Communications*, vol. 2, no. 4, pp. 373–392, 1995.

[3] D. Falconer, S. Ariyavisitakul, A. Benyamin-Seeyar, and B. Eidson, "Frequency domain equalization for single-carrier broadband wireless systems," *IEEE Communications Magazine*, vol. 40, no. 4, pp. 58–66, Apr. 2002.

[4] E. Telatar, "Capacity of multi-antenna Gaussian channels," *European Transactions on Telecommunications*, vol. 10, no. 6, pp. 585–595, Nov. 1999.

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

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

[7] B. Hochwald and S. ten Brink, "Achieving near-capacity on a multiple-antenna channel," *IEEE Communications Letters*, vol. 51, no. 3, pp. 389–399, Mar. 2003.

[8] J. Ketonen, J. Karjalainen, M. Juntti, and T. Hänninen, "MIMO detection in single carrier systems," in *Proceedings of the 19th European Signal Processing Conference*, Aug. 2011.

[9] K. Wong, C. Tsui, R. K. Cheng, and W. Mow, "A VLSI architecture of a K-best lattice decoding algorithm for MIMO channels," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, May 2002.

[10] M. Li, B. Bougart, E. Lopez, and A. Bourdoux, "Selective spanning with fast enumeration: A near maximum-likelihood MIMO detector designed for parallel programmable baseband architectures," in *Proceedings of the IEEE International Conference on Communications*, May 2008.

[11] T. Hänninen, J. Janhunen, and M. Juntti, "Novel detector implementations achieving 3G LTE downlink and uplink requirements," in *SDR-WInnComm*, Jan. 2013.

[12] Berkeley design technology Inc, "An independent evaluation of: High-level synthesis tools for Xilinx FPGAs," Tech. Rep., 2010.

[13] M. Myllyl, M. Juntti, and J. Cavallaro, "Architecture design and implementation of the increasing radius - list sphere detector algorithm." in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing. Taipei, Taiwan, 553556.*, 2009.

[14] M. Myllylä, "Detection algorithms and architectures for wireless spatial multiplexing in mimo-ofdm systems," *vol. C380 of Acta Universitatis Ouluensis, Doctoral thesis*, 2011.

[15] Xilinx, "Vivado Design Suite User Guide Implementation," UG904 (v2013.4), Tech. Rep., 2013.