FPGA IMPLEMENTATION OF A NEAR-ML SPHERE DETECTOR FOR 802.16E BROADBAND WIRELESS SYSTEMS

Chris Dick (Xilinx, San Jose, CA, USA; chris.dick@xilinx.com); Milos Trajkovic (Signum Concepts; San Diego, CA, USA; milos.trajkovic@signumconcepts.com); Slobodan Denic (Signum Concepts; San Diego, CA, USA; slobodan.denic@signumconcepts.com); Dragan Vuletic (Signum Concepts; San Diego, CA, USA; dragan.vuletic@signumconcepts.com); Raghu Rao, (Xilinx, San Jose, CA, USA; raghu.rao@xilinx.com), fred harris (San Diego State University (SDSU), San Diego, CA, USA; fharris@mail.sdsu.edu); Kiarash Amiri (Rice University, Houston, TX, USA; kiaa@rice.edu).

ABSTRACT

Spatial multiplexing multiple-input-multiple-output (MIMO) communication systems have recently drawn significant attention as a means to achieve tremendous gains in wireless system capacity and link reliability. The optimal hard decision detection for MIMO wireless systems is the maximum likelihood (ML) detector. ML detection is attractive due to its superior performance (in terms of BER). However, direct implementation grows exponentially with the number of antennas and the modulation scheme, making its ASIC or FPGA implementation infeasible for all but low density modulation schemes using a small number of antennas. Sphere decoding (SD) solves the ML detection problem in a computationally efficient manner. However, even with this complexity reduction, real-time implementation on a DSP processor is generally not feasible and high-performance parallel computing platforms such as FPGAs are increasingly being employed for this class of applications. The sphere detection problem affords many opportunities for algorithm and micro-architecture optimizations and tradeoffs. This paper provides an overview of and FPGA implementation of a sphere detector and channel matrix pre-processor applicable to the 802.16e air interface protocol. The architecture of the design is presented along with resource utilization data and BER performance curves.

1. INTRODUCTION

Spatial division multiplexing (SDM) MIMO processing significantly increases the spectral efficiency, and hence capacity, of a wireless communication system: it is a core component of next generation wireless systems, for example, WiMAX and other OFDM-based communication schemes. Sphere detection is a prominent method of simplifying the detection complexity in both SDM and SDMA systems while maintaining BER performance comparable with optimum maximum-likelihood (ML) detection [1], [2]. There are several approaches for realizing sphere detectors, and the algorithmic landscape is rich with methods that enable the designer to make various tradeoffs between performance, e.g. throughput of the wireless channel, BER, and implementation complexity [3]. While the algorithm (e.g. K-best or depth-first-search (DFS)) and hardware architecture obviously have a strong influence on the resulting BER performance of the MIMO detector, the channel matrix pre-processing that is typically conducted prior to sphere detection likewise has a significant influence on the BER performance [4]. The channel matrix pre-processing can range from very simple processing that might, for example, compute a preferred order for processing the spatially multiplexed data streams, based on variance computations applied to the channel matrix, through to much more sophisticated matrix factorization techniques that determine the preferred order for processing the streams in a more optimal (in the BER sense) manner.

This paper describes the field programmable gate array (FPGA) implementation of a detector for spatial multiplexing MIMO in 802.16e broadband wireless systems. By utilizing a channel matrix pre-processor that realizes a type of successive interference cancellation similar in concept to that employed in BLAST (Bell Labs Layered Space Time) processing, the detector achieves close to ML performance.

2. MIMO SYSTEM MODEL

Let us assume wireless MIMO system that has M_T number of transmit antennas and M_R number of receive antennas, where $M_R \geq M_T$. All transmit antennas use the same channel for simultaneous communication with the receive antennas. The complex input-output MIMO model can be written as:

$$\tilde{\mathbf{y}} = \tilde{\mathbf{H}}\tilde{\mathbf{s}} + \tilde{\mathbf{n}} \qquad (1)$$

where $\tilde{\mathbf{H}}$ denotes the $M_R \times M_T$ complex channel matrix, $\tilde{\mathbf{s}} = [\tilde{\mathbf{s}}_1, \tilde{\mathbf{s}}_2, ..., \tilde{\mathbf{s}}_{M_T}]^T$ is the M_T-dimensional transmitted vector from n transmit antennas, $\tilde{\mathbf{y}} = [\tilde{\mathbf{y}}_1, \tilde{\mathbf{y}}_2, ..., \tilde{\mathbf{y}}_{M_R}]^T$ is the complex receive vector of dimension $M_R \times 1$ and $\tilde{\mathbf{n}}$ is a circularly symmetric complex additive white Gaussian noise vector of size M_R. The values for $\tilde{\mathbf{s}}_j, j = 1,...,M_T$ are chosen from the complex set of symbols Ω with p bits per symbol, i.e. $|\Omega| = 2^p$, where the set of all possible transmitted vector symbols is denoted by Ω^p. The M_T parallel streams may use different modulation densities such as 4-, 16- or 64-QAM.

The detection process requires computing the solution to (1), and the goal is to reduce the required compute complexity by using simple arithmetic operations, while simultaneously retaining the numerical integrity of the final result. The matrix elements in (1) are formed from complex valued scalar quantities. However, this complex valued system of equations can be decomposed into a system of equations employing only real-valued numbers as follows

$$\mathbf{y} = \mathbf{H}\mathbf{s} + \mathbf{n} \qquad (2)$$

corresponding to

$$\begin{pmatrix} \Re(\tilde{\mathbf{y}}) \\ \Im(\tilde{\mathbf{y}}) \end{pmatrix} = \begin{pmatrix} \Re(\tilde{\mathbf{H}}) & -\Im(\tilde{\mathbf{H}}) \\ \Im(\tilde{\mathbf{H}}) & \Re(\tilde{\mathbf{H}}) \end{pmatrix} \begin{pmatrix} \Re(\tilde{\mathbf{s}}) \\ \Im(\tilde{\mathbf{s}}) \end{pmatrix} + \begin{pmatrix} \Re(\tilde{\mathbf{n}}) \\ \Im(\tilde{\mathbf{n}}) \end{pmatrix} \qquad (3)$$

with the new matrices having larger dimensions $M=2M_T$ and $N=2M_R$. While the dimensinality of the key matrices has increased, the arithmetic required to manipulate the matrix elements is now simplified due to the real-valued nature of the entries. Each $\mathbf{s}_i, i = 1,...,M$ in $\mathbf{s}$ is chosen from the set of real numbers Ω', which in the case of 64-QAM modulation is $\Omega' = \{\pm 7, \pm 5, \pm 3, \pm 1\}$.

In addition to the real-valued decomposition (RVD) described above, the modified RVD (M-RVD) presented in [2] is employed in our design to improve the BER performance. The new reordering of real and imaginary values of the complex components in (1) is summarized as follows:

$$\hat{\mathbf{y}} = \hat{\mathbf{H}}\hat{\mathbf{s}} + \hat{\mathbf{n}} \qquad (4)$$

or:

$$\begin{pmatrix} \Re(\tilde{\mathbf{y}}_1) \\ \Im(\tilde{\mathbf{y}}_1) \\ . \\ . \\ . \\ \Re(\tilde{\mathbf{y}}_{M_R}) \\ \Im(\tilde{\mathbf{y}}_{M_R}) \end{pmatrix} = \hat{\mathbf{H}} \begin{pmatrix} \Re(\tilde{\mathbf{s}}_1) \\ \Im(\tilde{\mathbf{s}}_1) \\ . \\ . \\ . \\ \Re(\tilde{\mathbf{s}}_{M_T}) \\ \Im(\tilde{\mathbf{s}}_{M_T}) \end{pmatrix} + \begin{pmatrix} \Re(\tilde{\mathbf{n}}_1) \\ \Im(\tilde{\mathbf{n}}_1) \\ . \\ . \\ . \\ \Re(\tilde{\mathbf{n}}_{M_R}) \\ \Im(\tilde{\mathbf{n}}_{M_R}) \end{pmatrix} \qquad (5)$$

The matrix $\hat{\mathbf{H}}$ is the permuted channel matrix of (3) whose columns are reordered to match the other vectors of the new decomposition ordering in (5). There is no extra computational cost associated with this new reordering. The optimum detector for the system described in (4) would be the maximum-likelihood detector which minimizes the value $\|\mathbf{y} - \mathbf{Hs}\|^2$ across all possible combinations of the vector $\mathbf{s}$. For high order modulation and large number of antennas, the number of calculations in the detection scheme grows exponentially, and the corresponding compute requirements render the real-time implementation of ML detection impractical. A reduced complexity quasi-ML algorithm can be formulated starting with the QR decomposition of the channel matrix $\mathbf{H}$ and defining the distance metric as shown in (6)

$$D(s) = \|\mathbf{y} - \mathbf{Hs}\|^2$$
$$= \|\mathbf{Q}^H\mathbf{y} - \mathbf{Rs}\|^2 = \sum_{i=M}^{1} \left| y'_i - \sum_{j=i}^{M} R_{i,j} s_j \right|^2 \qquad (6)$$

where **H=QR**, **QQ$^{\mathbf{H}}$=I** and $\mathbf{y}' = \mathbf{Q}^H\mathbf{y}$. The final term in (6) is a consequence of the upper triangular structure of the matrix **R**. The norm in (6) can be computed in M iterations, starting from the M^{th} ($i=M$) and progressing to the first antenna (i = 1*)*. For the first iteration the initial partial norm is defined as zero $T_{M+1}(\mathbf{s}^{(M+1)}) = 0$. Using the notation from [2], the partial Euclidian distance at each iteration can be calculated as:

$$T_i(\mathbf{s}^{(i)}) = T_{i+1}(\mathbf{s}^{(i+1)}) + |e_i(\mathbf{s}^{(i)})|^2 \qquad (7)$$

with $\mathbf{s}^{(i)} = [\mathbf{s}_i, \mathbf{s}_{i+1}, ..., \mathbf{s}_M]^T$, $i = M, M-1, ..., 1$ and

$$|e_i(\mathbf{s}^{(i)})|^2 = \left| y'_i - R_{i,i}s_i - \sum_{j=i+1}^{M} R_{i,j}s_j \right|^2$$
$$= |b_{i+1}(\mathbf{s}^{(i+1)}) - R_{i,i}s_i|^2 \qquad (8)$$

The iterative algorithm defined in equations (7) and (8) can be viewed as a tree traversal with each level of the tree i corresponding to processing symbols from the i^{th} antenna. The tree traversal can be performed using several different methods [5]. In our implementation we elect to employ a breadth-first search due to the attractive feedforward structure (and hence hardware friendly) nature of the approach. At each level only the K nodes with the smallest T_i are chosen for the expansion. This type of detector is called a K-best detector [2], [5].

The order in which the antennas are processed by the sphere detector has a profound impact on the BER (bit error rate) performance. So prior to sphere detection, channel reordering is applied. The proposed method is a V-BLAST-like channel reordering [6]. The method determines the optimum detection order of columns of the complex channel matrix defined in (1) by calculating the norms of the rows of the pseudo-inverse of the channel matrix over several iterations. Depending on the iteration count, the row with the maximum or minimum norm is selected. The row with the minimum Euclidian norm represents the influence of the strongest antenna while the row with the maximum

Euclidian norm represents the influence of the weakest antenna. The novel approach first processes the weakest stream. All subsequent iterations process the streams from highest to lowest power. The iteration process is illustrated in the following equations:

$$\tilde{\mathbf{G}}_j = \tilde{\mathbf{H}}_j^{\dagger} = \left(\tilde{\mathbf{H}}_j' \times \tilde{\mathbf{H}}_j\right)^{-1} \times \tilde{\mathbf{H}}_j',\ j = 1..M_T - 1$$
$$k_j = \arg\max_k \left\| \left(\tilde{\mathbf{G}}_j\right)_k \right\|^2 \text{ for } j = 1$$
$$k_j = \arg\min_k \left\| \left(\tilde{\mathbf{G}}_j\right)_k \right\|^2 \text{ for } j \neq 1$$
$$\tilde{\mathbf{H}}_1 = \tilde{\mathbf{H}},\ \tilde{\mathbf{H}}_{j+1} = \tilde{\mathbf{H}}_{j,\lfloor k_j \rfloor} \tag{9}$$

where $\tilde{\mathbf{H}}_{j,\lfloor k_j \rfloor}$ represents the deflated channel matrix excluding the detected k_j column which is placed on the $(M_T - j + 1)_{th}$ column space of the new sorted matrix. One can envision that every iteration of the reordering method operates on the smaller matrix and the last step will be calculated on 2×2 matrix. The last remaining column will be placed on the 1st column space of the sorted matrix.

3. FPGA HARDWARE IMPLEMENTATION

In this section the main features of the FPGA implementation are presented. The target technology is Xilinx Virtex®-5 FPGA technology. The design flow employs System Generator [7] for design capture simulation and verification. In order to support the different number of antenna/user and modulation orders, the detector is designed for the most demanding 4×4, 64-QAM case.

The block diagram of the MIMO 802.16e broadband wireless receiver is shown in the Figure 1.

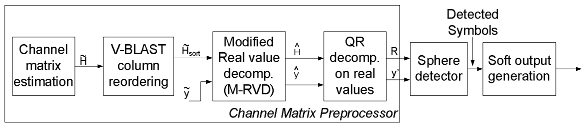
Detected Symbols
Channel matrix estimation
$\tilde{H}$
V-BLAST column reordering
$\tilde{H}_{sort}$
$\tilde{y}$
Modified Real value decomp. (M-RVD)
$\hat{H}$
$\hat{y}$
QR decomp. on real values
R
y'
Sphere detector
Soft output generation
Channel Matrix Preprocessor

Figure 1: Block diagram of the MIMO 802.16e broadband wireless receiver.

It is assumed that the channel matrix is perfectly known to the receiver which can be accomplished by classical means of channel estimation [8]. After channel reordering and QR decomposition, the sphere detector (SD) is applied. In preparation for engaging a soft-input-soft-output channel decoder (e.g. Turbo decoder), soft outputs are produced by computing the log-likelihood ratio (LLR) of the detected bits.

The main architectural elements of the system include the data sub-carrier processing and managing the system sub-modules to process the desired number of sub-carriers in real time while simultaneously minimizing processing latency. The channel matrix is estimated for every data sub-carrier which limits the available processing time for every channel matrix. For the selected FPGA, with a target clock frequency of 225MHz and a communication bandwidth of 5 MHz (corresponding to 360 data sub-carriers in a WiMAX system), the available number of processing clock cycles per channel matrix interval is calculated in (10)

$$num\_cycles = \frac{(102.9us/360)}{(1/225MHz)} \cong 64 \tag{10}$$

The design is optimized to meet the timing specification defined in (10). Hence, the sub-modules are configured in a pipeline fashion to accommodate the high throughput of the channel matrix coefficient stream. Besides the high data rate, managing the latency of the sub-modules was also an important issue guiding the design architecture. The latency issues were solved by introducing Time Division Multiplexing (TDM) of the successive channel matrices. This approach provided more processing time between the matrix elements of the same channel while still sustaining high data throughput. The number of channels comprising a TDM grouping varies as a function of the specific sub-module. The channel matrix inversion process employs 5 channels in the TDM scheme, while 15 channels are time division multiplexed in the real-valued QR decomposition module.

3.1. Channel Matrix Reordering

To meet the high data rate requirements of the system the channel ordering processing is realized using the pipelined architecture as shown in Figure 2. The channel matrix is successively deflated in dimension as it progresses through the processing pipeline. Buffer memories organized in a ping-pong manner are incorporated to store the sorted columns. The first iteration buffer holds the estimated matrix values and its size is determined by the number of data sub-carriers. Buffers for the additional iterations store 5 channels employed in the TDM structure.

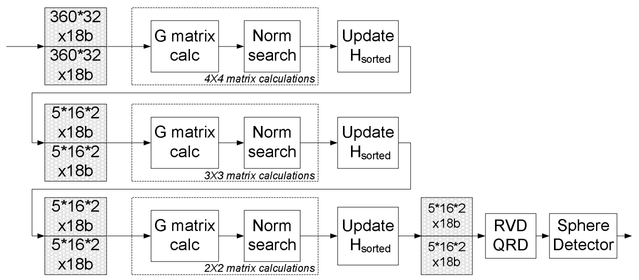

Figure 2: Iterative channel matrix reordering algorithm.

The calculation of the $\tilde{\mathbf{G}}$ matrix is the most demanding component in Figure 2. The heart of the process is matrix inversion which is realized using QR decomposition (QRD). A common method for realizing QRD is based on Givens Rotations [9]. This paper presents a novel approach for performing the complex rotations which are the fundamental computations in the systolic array we are using. Some well known algorithms for angle estimation and planar rotations, such as CORDIC, introduce very high system latency for the numerical accuracy required in this application, and this is unacceptable for our system. The goal was to find an alternative solution for vector rotation and phase estimation using the FPGA embedded DSP resources (DSP48 [10]). The algorithm described in [9] shows that the diagonal cell of the systolic structure rotates the input vector to the real axis and delivers the angle measured in this process to the off-diagonal cells where an additional set of rotations are applied. Denoting the complex value in a diagonal cell as *Z,* we observe that multiplication of the value contained in an off-diagonal cell with the complex conjugate Z^* and scaling by $1/sqrt\left(|Z|^2\right)$ produces the desired rotation. The division is done as a multiplication with a reciprocal value calculated using a polynomial approximation on the defined interval. Analysis of the function $f(x)$, defined in (11), showed the range where the function is close to linear. A first order approximation can be applied to data in the interval $[c,+\infty)$, where the constant c is chosen to be as large possible, while producing an acceptably small error for our application. The input data range is determined in the following manner $0<|Z|^2<2; \Re(Z), \Im(Z) \in [-1,1]$; hence the values lower than c have to be scaled in order to be in range $[c,2]$. The input scaling is implemented by shifting the data by the number of sign bits carried in the value, and the shift factor is determined using a binary search of the leading bits. Since the $sqrt$ function is approximated, the shift factor is chosen to be an even number of bits so the output compensation scaling of the result will just be simple shifts. Taking all these facts into account, the constant was determined to be $c = 0.25$. The first order approximation is illustrated in (11):

$$f(x+\Delta x) = f(x) + \Delta x \cdot f'(x); \quad f(x) = \frac{1}{\sqrt{x}} \tag{11}$$

The values $f(x)$ and $f'(x)$ are mapped to FPGA memories [10] while the multiplication is done using the DSP48 slice. Using the approximation, the final signal flowgraph of the complex rotator in the diagonal systolic cell is shown in the Figure 3.

The data sent to the off-diagonal cells are actually in-phase and quadrature components of the rotated vector scaled by the corresponding approximated value. The multiplication process in the off-diagonal cell is defined in (12):

$$\left[(I+jQ)*(I_{rot}-jQ_{rot})\right]\frac{1}{\sqrt{I_{rot}^2+Q_{rot}^2}} = \\ =(I+jQ)\left(\frac{I_{rot}}{\sqrt{I_{rot}^2+Q_{rot}^2}} - j\frac{Q_{rot}}{\sqrt{I_{rot}^2+Q_{rot}^2}}\right) \tag{12}$$

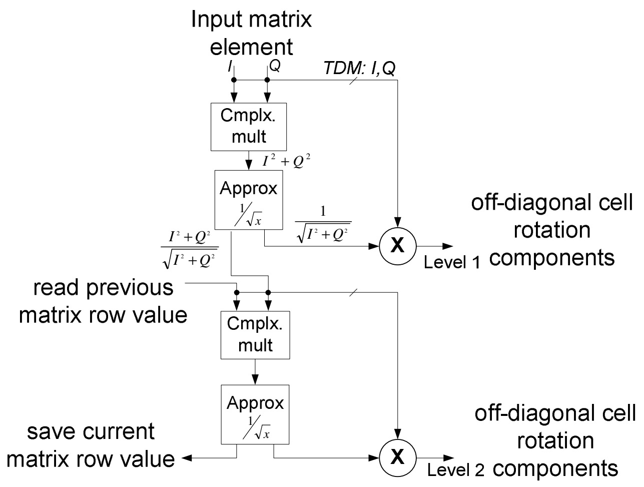
Input matrix element
I
Q
TDM: I,Q
Cmplx. mult
I^2+Q^2
Approx $1/\sqrt{x}$
$\frac{1}{\sqrt{I^2+Q^2}}$
$\frac{I^2+Q^2}{\sqrt{I^2+Q^2}}$
X
Level 1
off-diagonal cell rotation components
read previous matrix row value
Cmplx. mult
Approx $1/\sqrt{x}$
save current matrix row value
X
Level 2
off-diagonal cell rotation components

Figure 3: Block diagram of the diagonal systolic array cell.

High data throughput is obtained using a pipelined architecture for the diagonal and off-diagonal cells while the latency introduced by the approximation module and complex multiplier was managed by time division multiplexing (TDM) of the hardware across 5 channels. The number of diagonal and off-diagonal cells implemented for the 4×4 matrix is 1 and 7, respectively while the processing time to decompose a single matrix is $4\times4=16$ data cycles. The data is delivered at the rate of one sample every 3 clock cycles, so that total time to decompose a single matrix is $3\times4\times4=48$ clock cycles (out of the available 64). The rest of the available cycles are used for IO and initializing the memories for the next 5-channel TDM sub-frame. Back substitution of the decomposed matrix [9] and further reordering operations in (9) are implemented in the same TDM manner using established and published algorithms.

3.2. Modified Real-Valued QR decomposition

After obtaining the optimal ordering of the channel matrix columns, the QR decomposition (QRD) on the real-valued matrix coefficients is applied. The functional unit used for this QRD processing is similar to the QRD engine designed to compute the inverse matrix, but with some modifications. The input data in this case are real valued and the systolic array structure has a correspondingly higher degree. In essence, the higher order matrix is processed, as explained in (5), and in order to meet the desired timing constraints the input data consumption rate had to be 1 input sample per clock cycle. This introduced challenges around processing latency problems which couldn't be addressed with a 5-channel TDM structure. The number of channels in a TDM group was increased to 15 to provide more time between the successive elements of the same channel matrix.

3.3. Sphere Detector (SD)

The norm computation defined in (8) is done in the PED blocks of the SD. Depending on the level of the tree, three different PED blocks are used: the root node PED block calculates all possible PEDs (tree level index is $i = M = 8$). The second level PED block computes 8 possible PEDs for each of the 8 survivor paths generated in the previous level. This will give us 64 generated PEDs for the tree level index $i = 7$. The third type of PED block is used for all other tree levels which compute the closest-node PED for all PEDs computed on the previous level. This will fix the number of branches on each level to $K = 64$, thus propagating to the last level $i = 1$ and producing 64 final PEDs along with their detected symbol sequences. The closest-node search is presented in the sort-free SD [2].

The pipeline architecture of the SD allows data processing on every clock cycle, thus the number of PED blocks necessary at every tree level is only one. The total number of PED units is equal to the number of tree levels, which for 4×4 64-QAM, is 8. The block diagram of the SD is illustrated in the Figure 4. The input data are provided from the real-valued QRD and the outputs are saved and analyzed further in the design, depending on the type of decoding process.

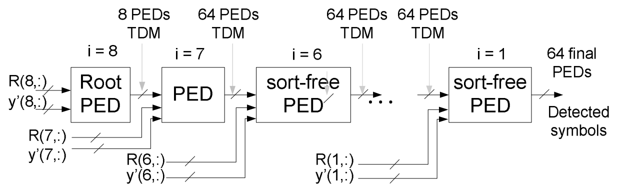

Figure 4: Block diagram of the sphere detector

3.4. Soft output decoding

Two types of decoding process can be employed in the SD, hard decoding, that determines the sequence having minimum distance metric through all levels in the tree, and soft decoding which represents each output bit as a Log-Likelihood Ratio (LLR) value, these values typically being supplied as an a priori input values to the channel decoder. Although soft decoding isn't part of the material reported in this paper, the sphere detector implementation provides support for the generation of soft outputs for use in the iterative detector/decoder shown in Figure 5. Floating point simulations have shown a significant improvement in BER performance when iterative soft detection is used [5]. The iterative structure presents challenges for the memory architecture of the system, and additional memory buffers must be used to store block of bits along with their PEDs, so increasing the memory footprint of the FPGA design.

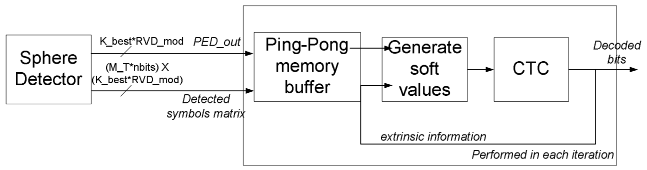

Figure 5: Block diagram of the iterative soft decoding

Generating Soft Values block in the Figure 5 is designed based on the algorithm described in [12]. LLR values are computed, saturated to 3 bits and decoded further using a CTC. The algorithm uses extrinsic information provided by the CTC as a priori LLR, L_A and based on the stored PED value calculates the output LLR, L_E. High latency and delay introduced by the soft generating block is minimized by using a parallel/pipelined architecture.

4. FPGA RESOURCE UTILIZATION

The architecture described in the previous sections was realized using the Xilinx System Generator for DSP [7] design flow and implemented on a Virtex®-5 XC5VFX130T-2FF1738 FPGA [10]. The target clock frequency was 225MHz. As mentioned earlier, the most computationally demanding 4×4 64-QAM configuration has been designed and tested. The achievable raw data rate in this case is calculated as follows in (13):

$$D = \frac{1}{T_{symb}} * num\_subcarriers * M_T * nbits$$

$$D = \frac{1}{102.9 \mu s} * 360 * 4 * 6 = 83.965 [Mbps] \qquad (13)$$

The implementation and simulation included the detection process illustrated in the Figure 1 with the exclusion of the soft output generation module. Table I shows the resource consumption for each of the key functional units in the design. The percentage utilization values in the table indicate FPGA area expressed relative to a XC5VFX130T device.

TABLE I
RESOURCE FOOTPRINT SUMMARY BY SUB-SYSTEM

Function	Slices	LUTs/FFs	DSP48	Block RAM
Channel preproc.	9,999 (48%)	20,339/29,954 (24%)	159 (49%)	105 (17%)
RVD QRD	1,715 (8%)	4,418/5,556 (5%)	30 (9%)	27 (4%)
Sphere Detector	2,445 (11%)	3,113/6,525 (3%)	48 (15%)	12 (2%)

5. SIMULATION RESULTS

The entire detection chain was realized using System Generator for DSP. Design validation employed not only the simulation semantics of the MATLAB/Simulink® [13]

environment but also the co-simulation capabilities of System Generator [7]. In our simulation we assume that the channel matrix is known to the receiver. In-phase and quadrature components of the channel matrix coefficients are drawn from a normal distribution and delivered from MATLAB to the System Generator modeling environment. The bit error rate (BER) is computed using this simulation framework. Figure 6 contrasts the BER plots for our fixed-point hard decision implementation, the floating-point version of the model and the ML curve. We note that there is virtually no difference between the floating-point software model and the hardware implementation. For a BER of 10^{-5} the difference is only 0.006dB.

6. CONCLUSION

This paper has described the hardware implementation of a sphere detector for 802.16e systems. Unlike many papers on this topic we have described the algorithmic and architecture details of the channel matrix pre-processor preceding the sphere detector. There are many ways to realize the preprocessing, and while our method is computationally complex, the resulting BER performance is close to ML. We note that the pre-processor is the largest functional unit in the design requiring 5.3x more multipliers than the QRD block and 3.3x more multipliers than the sphere detector itself. However, considering the system benefits delivered by the design, and noting that new generation FPGAs provide in excess of 2000 multipliers, the cost of the circuit is warranted.

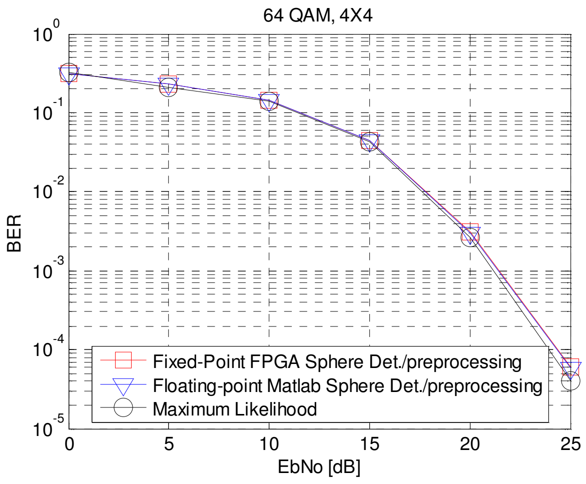

Figure 6: BER curves comparing the 4x4 64-QAM system for the floating point MATLAB simulation (hard decision), System Generator design (hard decision) and ML curve.

7. REFERENCES

[1] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner and H. Bolcskei, "*VLSI implementation of MIMO detection using the sphere decoding algorithm*," IEEE JSSC, vol. 40, no. 7, pp. 1566–1577, Jul 2005.

[2] K. Amiri, C. Dick, R. Rao, J. R. Cavallaro, "*Flex-Sphere: An FPGA configurable sort-free sphere detector for multi-user MIMO wireless system*", Proceedings of the 2008 Software Defined Radio Technical Conference, Oct. 26-30, 2008, Washington D.C.

[3] L. G. Barbero, J. S. Thompson, "*Rapid Prototyping of a Fixed-Throughput Sphere Decoder for MIMO Systems*", IEEE International Conference on Communications, Volume 7, Page(s):3082 – 3087, June 2006.

[4] L. G. Barbero, J. S. Thompson, "*A fixed-complexity MIMO detector based on the complex sphere decoder*" Signal Processing Advances in Wireless Communications, 2006. SPAWC '06. IEEE 7th Workshop on, Jul. 2006.

[5] Z. Guo, P. Nilsson, "*Algorithm and implementation of the K-best sphere decoding for MIMO detection*", IEEE Journal Selected Areas in Communications, Volume 24, Issue 3, Page(s): 491 – 503, March 2006.

[6] P.W. Wolniansky, G. J. Foschini, G.D. Golden, "*V-BLAST: an architecture for realizing very high data rates over the rich-scattering wireless channel*", Proc. URSI International Symposium on Signals, Systems and Electronics (ISSSE '98), Atlanta, GA, pp. 295–300, Sept. 1998

[7] Xilinx, "*System Generator for DSP – User Guide*", Release 10.1, March 2008.

[8] A. R. S. Bahai, B. R. Saltzberg and M. Ergen, Multi-carrier digital communications theory and applications of OFDM, Springer, 2004.

[9] M. Karkooti, J.R. Cavallaro, C. Dick, "*FPGA Implementation of Matrix Inversion Using QRD-RLS Algorithm*", Conference Record of the Thirty-Ninth Asilomar Conference on Signals, Systems and Computers, page(s): 1625-1629, 2005.

[10] Xilinx, "*Virtex-5 FPGA – User Guide*", UG190 (v4.5), January 2009.

[11] Xilinx, "*IEEE 802.16e CTC decoder v3.0*", Product specification DS634, October 2007.

[12] B. M. Hochwald, S. ten Brink, "*Achieving near-capacity on a multiple-antenna channel*" IEEE Trans. Commun., vol. 51, no. 3, pp. 389–399, Mar. 2003.

[13] The Mathworks, Simulink – Simulation and Model Based Design, http://www.mathworks.com/products/simulink/