FPGA implementation of a near-ML sphere detector for 802.16e broadband wireless systems

chris dick: Xilinx
Milos Trajkovic, Slobodan Denic, Dragan Vuletic: Signum Concepts
Raghu Rao: Xilinx
fred harris: Signum Concepts & San Diego State University
Kiarash Amiri: Rice University
Agenda

- Spatial multiplexing MIMO
- Quasi-ML SM decoders and sphere detection
- Useful approximations for efficient hardware implementation
- Guided tour of sphere detector implementation for 802.16e
- Implementation results
### Specifications
- 5MHz WiMAX
- 360 data sub-carriers
- antenna ordering to be updated for every sub-carrier for every OFDM symbol
  - extremely rapid update; could probably be relaxed
- 2, 3 or four antennas programmable dynamically at run-time
- QPSK, 16-QAM and 64-QAM configurable on a per user basis
FPGA Sphere Detector
Effective FPGA implementation is intersection of:

- **Algorithm optimizations**
  - K-best, depth-first search, sort-free,…

- **Architecture**
  - clock rates
  - hardware folding

- **Micro-architecture optimizations that are possible with the FPGA**
  - bit-width optimizations
  - math optimizations: $\ell^1$ vs $\ell^2$ vs $\ell^\infty$ norms to simplify hardware footprint for sphere detector for example
Sphere Detection – QRD Pre-processing

- **Maximum Likelihood detection**

\[ \hat{s} = \arg \min_{s \in \Lambda^{MT}} \|y - Hs\|^2 \]

- **Key element of complexity reduction is suitable factoring of channel matrix** \(H\)

\[ H = QR \]

\(M_R \times M_T\) matrix \(Q\) has orthonormal columns and \(R\) is upper triangular

- **Distance metric can be written as**

\[ d(s) = c + \|\hat{y} - Rs\|^2 \quad \text{with} \quad \hat{y} = Q^H y = Rs^{ZF} \]

\[ \|\hat{y} - R s\|^2 \]
Example: Solving for Antenna 2, Real Component of Symbol

- Solve for symbol transmitted on antenna 1 by performing hypothesis test
  - recall that for QPSK the coordinates for the constellation points are \( \Omega = \{ \pm 3, \pm 1 \} \)
- Form all of the possible required products \( R_s \) and differences as shown

\[
\begin{align*}
\mathbf{d}(s_{-3}) &= \left[ \begin{array}{c} \tilde{y}_0 \\ \tilde{y}_1 \\ \tilde{y}_2 \\ \tilde{y}_3 \\ \end{array} \right] - \left[ \begin{array}{cccc} R_{0,0} & R_{0,1} & R_{0,2} & R_{0,3} \\ 0 & R_{1,1} & R_{1,2} & R_{1,3} \\ 0 & 0 & R_{2,2} & R_{2,3} \\ 0 & 0 & 0 & R_{3,3} \\ \end{array} \right] \left[ \begin{array}{c} \mathfrak{R}(\hat{s}_{M_T=1}) \\ \mathfrak{R}(\hat{s}_{M_T=2}) \\ \mathfrak{R}(\hat{s}_{M_T=2}) \\ \mathfrak{R}(\hat{s}_{M_T=2}) \\ \end{array} \right] \end{align*}
\]

\[
\begin{align*}
\mathbf{d}(s_{-1}) &= \left[ \begin{array}{c} \tilde{y}_0 \\ \tilde{y}_1 \\ \tilde{y}_2 \\ \tilde{y}_3 \\ \end{array} \right] - \left[ \begin{array}{cccc} R_{0,0} & R_{0,1} & R_{0,2} & R_{0,3} \\ 0 & R_{1,1} & R_{1,2} & R_{1,3} \\ 0 & 0 & R_{2,2} & R_{2,3} \\ 0 & 0 & 0 & R_{3,3} \\ \end{array} \right] \left[ \begin{array}{c} \mathfrak{R}(\hat{s}_{M_T=1}) \\ \mathfrak{R}(\hat{s}_{M_T=2}) \\ \mathfrak{R}(\hat{s}_{M_T=2}) \\ \mathfrak{R}(\hat{s}_{M_T=2}) \\ \end{array} \right] \end{align*}
\]

\[
\begin{align*}
\mathbf{d}(s_{+1}) &= \left[ \begin{array}{c} \tilde{y}_0 \\ \tilde{y}_1 \\ \tilde{y}_2 \\ \tilde{y}_3 \\ \end{array} \right] - \left[ \begin{array}{cccc} R_{0,0} & R_{0,1} & R_{0,2} & R_{0,3} \\ 0 & R_{1,1} & R_{1,2} & R_{1,3} \\ 0 & 0 & R_{2,2} & R_{2,3} \\ 0 & 0 & 0 & R_{3,3} \\ \end{array} \right] \left[ \begin{array}{c} \mathfrak{R}(\hat{s}_{M_T=1}) \\ \mathfrak{R}(\hat{s}_{M_T=2}) \\ \mathfrak{R}(\hat{s}_{M_T=2}) \\ \mathfrak{R}(\hat{s}_{M_T=2}) \\ \end{array} \right] \end{align*}
\]

\[
\begin{align*}
\mathbf{d}(s_{+3}) &= \left[ \begin{array}{c} \tilde{y}_0 \\ \tilde{y}_1 \\ \tilde{y}_2 \\ \tilde{y}_3 \\ \end{array} \right] - \left[ \begin{array}{cccc} R_{0,0} & R_{0,1} & R_{0,2} & R_{0,3} \\ 0 & R_{1,1} & R_{1,2} & R_{1,3} \\ 0 & 0 & R_{2,2} & R_{2,3} \\ 0 & 0 & 0 & R_{3,3} \\ \end{array} \right] \left[ \begin{array}{c} \mathfrak{R}(\hat{s}_{M_T=1}) \\ \mathfrak{R}(\hat{s}_{M_T=2}) \\ \mathfrak{R}(\hat{s}_{M_T=2}) \\ \mathfrak{R}(\hat{s}_{M_T=2}) \\ \end{array} \right] \end{align*}
\]

- Now make a decision for the real component of the symbol sent on antenna 1

\[
\mathfrak{R}(\hat{s}_{M_T=1}) = \min \{ \mathbf{d}(s_{-3}), \mathbf{d}(s_{-1}), \mathbf{d}(s_{+1}), \mathbf{d}(s_{+3}) \}
\]
### Process of Sphere Decoding

- The decoding process can be mapped to finding a shortest path (with minimum Euclidean distance) in a tree topology.

\[
\|\hat{y} - R \mathbf{s}\|^2 = \\
\begin{vmatrix}
\hat{y}_1 - (R_{1,1} s_1 + R_{1,2} s_2 + \ldots + R_{1,M_T} s_M) \\
\hat{y}_2 - (R_{2,2} s_2 + \ldots + R_{2,M_T} s_M) \\
\vdots \\
\hat{y}_{M_T} - (R_{M_T,M_T} s_M)
\end{vmatrix}^2
\]

\[
= \begin{vmatrix}
-b_1 \\
b_2 \\
\vdots \\
b_{M_T}
\end{vmatrix}^2
\]

- **Computational complexity for one solution**
  - **Adds**: $M(M+1)/2 + 2M - 1$
  - **Mults**: $M(M+1)/2$ complex mults
  - 2M square operators
  - **Mem** for $H$: $M^2$ complex values

- **E.g. 16x16 systems**
  - $N_{\text{add}} = 167$
  - $N_{\text{mult}} = 136$ complex mults
  - 32 square operators
  - **Mem** for $H$: 256 complex values
Mapping Distance Computation to FPGA

- The actual system we are considering is 4x4 and the matrix equation looks like

\[
\begin{bmatrix}
\tilde{y}_0 \\
\tilde{y}_1 \\
\tilde{y}_2 \\
\tilde{y}_3 \\
\tilde{y}_4 \\
\tilde{y}_5 \\
\tilde{y}_6 \\
\tilde{y}_7
\end{bmatrix} - 
\begin{bmatrix}
R_{0,0} & R_{0,1} & R_{0,2} & R_{0,3} & R_{0,4} & R_{0,5} & R_{0,6} & R_{0,7} \\
R_{1,1} & R_{1,2} & R_{1,3} & R_{1,4} & R_{1,5} & R_{1,6} & R_{1,7} \\
R_{2,2} & R_{2,3} & R_{2,4} & R_{2,5} & R_{2,6} & R_{2,7} \\
R_{3,3} & R_{3,4} & R_{3,5} & R_{3,6} & R_{3,7} \\
R_{4,4} & R_{4,5} & R_{4,6} & R_{4,7} \\
R_{5,5} & R_{5,6} & R_{5,7} \\
R_{6,6} & R_{6,7} \\
R_{7,7}
\end{bmatrix}
\]

- The element wise product and subtraction maps well to the DSP48 element in the FPGA
- Leverages the dedicated routing between cascaded DSP48’s to minimize the critical path and hence maximize the FPGA clock frequency
- DSP48 tile contains a multiplier for computing the products and add/sub for accumulating the partial-products

- Sphere detector essentially a streaming dataflow computation
- Parallel processing pipeline for computing the incremental distance metric
Sphere Detection Tree

Example: $M_T = 4$ antennas, $K=3$, 16-QAM

\[
\begin{align*}
|\tilde{y}_7 - R_{7,7}s_7|^2 \\
+ |\tilde{y}_6 - R_{6,7}s_7 - R_{6,6}s_6|^2 \\
+ |\tilde{y}_5 - R_{5,7}s_7 - R_{5,6}s_6 - R_{5,5}s_5|^2
\end{align*}
\]

- $K$ survivors at each level
- requires sort
  \[\rightarrow\text{ high-latency process}\]

MIN value: Detected Vector

\[\{\hat{s}_i\} \]

\[\{\hat{s}_1\} \]

\[\{\hat{s}_2\} \]

\[\{\hat{s}_3\} \]

\[\{\hat{s}_4\} \]
- Two levels in tree correspond to one antenna
- First 2 levels fully expanded
- Expansion of the minimum node rather sorting whole level
  - approximate sort avoids requirement for high-latency min-search at each level
Micro-Architecture Optimizations to Reduce Cost: Simplified Norms

Let $T_i = X_i^2$

partial sphere constraint is

$$X_i = \sqrt{X_{i+1}^2 + |e_i|^2} < r$$

approximate the $\ell^2$ norm with a different norm

$$X_i \approx f\left(|X_{i+1}| + |e_i|\right)$$

$$X_i \approx |X_{i+1}| + |e_i| \text{ or}$$

$$X_i \approx \max\left(|X_{i+1}|, |e_i|\right)$$

In the complex-valued case, corresponding approximations for $|e_i|$ are

$\ell^1$ norm: $|e_i| = |\Re\{e_i\}| + |\Im\{e_i\}| \text{ or}$

$\ell^\infty$ norm: $|e_i| = \max\left(|\Re\{e_i\}|, |\Im\{e_i\}|\right)$

- Motivation for using simplified norm is to reduce complexity of implementation
  - Minimize FPGA resource footprint
  - Mechanism that permits tradeoff between FPGA architectural elements, e.g. DSP48’s for LUTs
$L-1, L-2$ Norm Comparison

- Negligible impact using simpler $L-1$ norm versus $L-2$ norm
- $K$-Best sphere detector is quasi MLD solution
  - Already some BER degradation introduced with $K$-Best
- Eliminates MPYs in implementation which may be beneficial

Floating-point simulation
802.16e Sphere Detector
Antenna Ordering

- Figure below shows the antenna ordering pipeline

\[
G_j = H_j^\dagger = \left( H_j^* \cdot H_j \right)^{-1} \cdot H_j^*, \quad j = 1, \ldots, M_T
\]

\[
k_j = \arg \max_k \left\| \left( G_j \right)_k \right\|^2 \quad \text{for } j = 1
\]

\[
k_j = \arg \min_k \left\| \left( G_j \right)_k \right\|^2 \quad \text{for } j \neq 1
\]

\[
H_1 = H, \quad H_{j+1} = H_{j[k_j]}
\]
Key to achieving good BER performance is around how the antenna ordering is performed
- the ‘V-BLAST channel reordering’ component above

Assume we have the channel estimate
802.16e Sphere Detector
Antenna Ordering Processing Pipeline

- Linear processing pipeline of $M_T-1$ stages employed
- Each stage: QRD matrix inversion, norm, updated $H$
- Buffer memories store intermediate results and support the TDM pipeline
- Antenna ordering: Top

Each stage of the pipeline realized using a QRD-style systolic array

4X4 matrix calculations

3X3 matrix calculations

2X2 matrix calculations
Matrix Inversion using complex QRD Pipeline Processing

- QRD using Givens rotations
- Angle estimation and vector rotation performed using complex multiplication and approximation
- High-speed pipelined architecture using Xilinx DSP tiles (DSP48)
- Heavy pipelining and time division multiplexing of hardware employed to meet latency/throughput requirements of 802.16e

\[ f(x + \Delta x) = f(x) + \Delta x \cdot f'(x); \quad f'(x) = \frac{1}{\sqrt{x}} \]

\( f(x) \) and \( f'(x) \) stored in FPGA BRAM
addition and \( \Delta x \cdot f'(x) \) realized using DSP48

\[ k'_{1,1} = \frac{k'_{0,0}}{\sqrt{|h_{1,0}|}} \]

\( k_{1,1} \) multiplication performed in inner cell

Perform rotations to successively null matrix entries
Approximating $1/\sqrt{x}$

$f(x + \Delta x) = f(x) + \Delta x f'(x)$; $f(x) = 1/\sqrt{x}$
Key to performance is on determining suitable order in which to process the antennas.

Innovative element of this design is on the use of interference cancellation in channel matrix pre-processing and novel real-valued ordering of the sphere detector tree.

BER plot for fixed-point design generated from simulation of System Generator model.

Floating-point matlab model overlay also shown.
802.16e Sphere Detector
Tree Search

- Once the tree ordering is determined a detection tree is assembled using real-valued decomposition and approximate sorting strategy to minimize detection latency.

\[
T_i(s^{(i)}) = T_{i+1}(s^{(i+1)}) + |e_i(s^{(i)})|^2
\]

\[
b_{i+1}(s^{(i+1)}) = \hat{\gamma}_i - \sum_{j=i+1}^{M} R_{ij} s_j
\]
802.16e Sphere Detector Development/Deployment

Deployment Environment
- Microprocessor
- DSP Processor

Development / Debugging Environment (Host PC)
- Standalone C/C++ Program
- MATLAB/Simulink
- SysGen HW Co-sim API

Gigabit Ethernet Switch

LocalLink Interface

Ethernet Co-simulation Protocol

- Virtex-6 LX240T: 241,152 LCs, 768 MPYs
- USB, ethernet
- Hardware cosim support using Xilinx DSP tools

Virtex-5 FPGA
802.16e Sphere Detector
BERT using HW Cosim

- System Generator HW Cosim module from previous slide
802.16e Sphere Detector
BERT using HW Cosim: Demo UI

MIMO 802.16e Sphere Detector Demo

![Graph showing BER vs Eb/No (dB)]

- **Settings**:
  - Eb/No Range
    - Lowest Value: 0 dB
    - Highest Value: 25 dB
  - Step Size: 5 dB
  - Modulation Type:
    - QPSK
    - QAM-16
    - QAM-64
  - Number of Antennas:
    - 2x2
    - 3x3
    - 4x4

- **Buttons**:
  - Initialize
  - Configure
  - Start

- **Status**:
  - Ready!
V-BLAST channel ordering is key

Each sub-carrier for each OFDM symbol
- could probably be relaxed to reduce FPGA area

Main compute complexity is in the antenna ordering

802.16e 5 MHz bandwidth, 360 data sub-carriers, 4x4 antenna configuration, 64-QAM

Real-time operation at 102.9 microsecond frame rate

Compute performance
- ~53E9 MPYs/second, ~21E6 matrix inversions/second, 225E6 path metric computations/second

<table>
<thead>
<tr>
<th>Function</th>
<th>Slices</th>
<th>LUT/FFT Pairs</th>
<th>Block Memory (18k)</th>
<th>DSP Slices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel Pre-processor</td>
<td>9,999</td>
<td>20,339/29,954</td>
<td>105</td>
<td>159</td>
</tr>
<tr>
<td>RVD QRD</td>
<td>1,715</td>
<td>4,418/5,556</td>
<td>27</td>
<td>30</td>
</tr>
<tr>
<td>Sphere Detector</td>
<td>2,445</td>
<td>3,113/6,525</td>
<td>12</td>
<td>48</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>14,159</td>
<td>27,870/42,035</td>
<td>144</td>
<td>237</td>
</tr>
</tbody>
</table>
Conclusion

- Key to performance is on determining suitable order in which to process the antennas

- Innovative element of this design is on the use of interference cancellation in channel matrix pre-processing and novel real-valued ordering of the sphere detector tree

- **Might not want to lock this function down in ASIC/ASSP**
  - desire to have flexibility to tailor ML-decoder to air-interface requirements
  - upgrade with new algorithms

- **Algorithmic, architecture and micro-architecture optimizations required for good implementation**

- **Architectural transparency provided by System Generator design flow allows for production of FPGA optimized implementation**