Hardwarebeschleuniger für Interference Alignment in In-House Mehrbenutzer-Kommunikationssystemen

Markus Kock, Holger Blume
Outline

- Interference Alignment
  - increased channel capacity in multi-user scenarios
  - Physical layer technique
- MMSE Interference Alignment Algorithm
- Hardware architecture
  - Parallelization
  - Low-latency operations
- Implementation results
- Conclusion
Objectives

- Dedicated hardware accelerator for Minimum Mean Square Error (MMSE) Interference Alignment (IA)
- Digital baseband processing
- Low-latency real-time operation (latency < 1 ms)
Multi-User MIMO Communication System, TDMA

- MIMO spatial multiplexing: multiple antennas per user, one data stream per antenna
- Channel capacity shared by all users
Multi-User MIMO Communication System, IA

- MIMO spatial multiplexing: multiple antennas per user, one data stream per antenna
- Channel capacity shared by all users
- Interference Alignment: simultaneously transmitting users
Multi-User MIMO System: Goal

Channel capacity scales with number of users $K$
Interference Alignment System Model

- Scenario: multi-user point-to-point communication system
- Linear precoding and decoding at TX and RX, respectively

Received signal: \( Y_k = U_k^T \left( \sum_{j=1}^{K} H_{kj} V_j X_j + n_k \right) \)

- \( K \): #users, \( d \): datastreams / user
- \( N_t \): antennas / TX, \( N_r \): antennas / RX
- Problem formulation:
  Determine \( V_k \) and \( U_k \) for given \( H_{kj} \)
- Several approaches feasible:
  Max SINR, Max Sum-Rate, MMSE, ...
Fast-changing channels

- Channel coherence time depends on scenario
- Precoding matrices need to be adapted to the channel within channel coherence time
- High data throughput AND low-latency realtime computation required

⇒ Low-latency computation of $V_k$ and $U_k$ ($< 1$ ms)
- Additional system latencies: channel estimation, transmit CSI, distribute $V_k$ and $U_k$
MMSE–IA Algorithm

- MMSE criterion: minimize overall interference + noise

Algorithm\(^{[1]}\):

1. Start with arbitrary \( \mathbf{V}_k \)
2. Update
   \[
   \mathbf{U}_k = \left( \sum_{j=1}^{K} \mathbf{H}_{kj} \mathbf{V}_j \mathbf{V}_j^H \mathbf{H}_{kj}^H + \sigma^2 \mathbf{I} \right)^{-1} \mathbf{H}_{kk} \mathbf{V}_k
   \]
   \[
   \mathbf{V}_k = \left( \sum_{j=1}^{K} \mathbf{H}_{kj}^H \mathbf{U}_j \mathbf{U}_j^H \mathbf{H}_{jk} + \lambda_k \mathbf{I} \right)^{-1} \mathbf{H}_{kk}^H \mathbf{U}_k
   \]
   Lagrange multiplier \( \lambda_k \) iteratively determined to satisfy TX power constraint \( \| \mathbf{V}_k \|_F^2 \leq 1 \)
3. Compute system MSE
4. Repeat steps 2 and 3 until convergence

\(^{[1]}\) D. Schmidt, C. Shi, R. Berry, M. Honig, and W. Utschick, "Minimum Mean Squared Error Interference Alignment", 2009
Parallelization

Inherent data dependencies limit parallelization

Data dependencies  Maximum parallel HW data flow

Institute of Microelectronic Systems
Hardware System Architecture

Dedicated accelerator for integration in SDR SoCs

- All communication via OCP or AXI on-chip busses
- Local matrix cache (BRAM)
  - $H_{jk}$ channels
  - $V_k$ precoders
  - $U_k$ decoders
- Variable number of processing elements (PE) for computing $V_k$ and $U_k$
- Controller
Processing Element

- Compute V or U for one user at a time (mode select)
- Main complexity: Gaussian elimination, shared by V and U modes
- Mode V
  - Iterative root-finding for $\lambda_k$
- Mode U
  - No iterations required
Low-Latency Equation System Solver

- Inner loop contains matrix inversion
  - Solve equation system instead
    \[
    U_k = \left( \sum_{j=1}^{K} H_{kj} V_j V_j^H H_{kj}^H + \sigma^2 I \right)^{-1} \quad H_{kk} V_k = Q_k^{-1} B_k
    \]
    \[
    Q_k U_k = B_k
    \]
- Candidates: SVD, LU, QR, ...
- Criterion: low-latency
  - Gaussian elimination
    - Small matrix sizes \(\Rightarrow\) sufficient precision
    - Latency: one multiplication per eliminated unknown
Two-Step Bareiss Algorithm

- Variation of Gaussian elimination
- Integer-preserving, division-free (elimination loop)
- Eliminate two unknowns per step
- Row-wise normalization after each elimination step
- One final division required per result coefficient

$$Q_k U_k = B_k$$

Augmented System:

$$\begin{bmatrix} q_{11} & q_{12} & \cdots & q_{1N} & b_{11} & \cdots & b_{1d} \\ q_{21} & q_{22} & \cdots & q_{2N} & b_{21} & \cdots & b_{2d} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \cdots & \vdots \\ q_{N1} & q_{N2} & \cdots & q_{NN} & b_{N1} & \cdots & b_{Nd} \end{bmatrix} \xrightarrow{\text{Bareiss}} \begin{bmatrix} I \mid U_k \end{bmatrix}$$
Two-Step Bareiss Algorithm Result

- Two unknowns eliminated after one step

\[
\begin{bmatrix}
\tilde{q}_{11} & 0 & \ldots & \tilde{q}_{1N} \\
0 & \tilde{q}_{22} & \ldots & \tilde{q}_{2N} \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \ldots & \tilde{q}_{NN}
\end{bmatrix}
\begin{bmatrix}
\tilde{b}_{01} \\
\tilde{b}_{21} \\
\vdots \\
\tilde{b}_{N1}
\end{bmatrix}
\]

One common factor per row
- Skip multiplication
- Fewer operations compared to single step elimination

- Repeat to obtain diagonal form

\[
\begin{bmatrix}
d_{11} & 0 \\
d_{22} & \vdots \\
0 & d_{NN}
\end{bmatrix}
\begin{bmatrix}
\tilde{u}_{11} & \ldots & \tilde{u}_{1d} \\
\tilde{u}_{21} & \ldots & \tilde{u}_{2d} \\
\vdots & \ddots & \vdots \\
\tilde{u}_{N1} & \ldots & \tilde{u}_{Nd}
\end{bmatrix}
\]

Final division required for each result coefficient \(u\)
Two-Step Bareiss Systolic Array Processor

- Fixed-point representation
- Only multiplications and additions/subtractions required
- Data shifted diagonally by two elements per elimination step
Systolic Array Processor Critical Path

- Eliminate two unknowns
- Critical path: 2 MUL + 4 ADD
- Row-wise block renormalization (shift) after each elimination step

\[(AB - CD) \cdot E - \ldots\]
### Implementation Results

- Channel capacity within 0.1% vs. floating-point MATLAB reference
- FPGA synthesis
  - Target: Xilinx Virtex-6 XC6VLX550T-2
  - Software: ISE 14.7
  - Clock constraint: 50 MHz
  - Latency 520 μs for worst-case system (N_t = N_r = 11, K = 19)

<table>
<thead>
<tr>
<th>K</th>
<th>N_{tr}</th>
<th>d</th>
<th>n_{PE}</th>
<th>n_{MM}</th>
<th>FF</th>
<th>LUT</th>
<th>DSP48E1</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>51531</td>
<td>7.50%</td>
<td>81560</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>20942</td>
<td>3.05%</td>
<td>32702</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>16585</td>
<td>2.41%</td>
<td>25682</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>32980</td>
<td>4.80%</td>
<td>56974</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>24916</td>
<td>3.62%</td>
<td>43514</td>
</tr>
</tbody>
</table>
Conclusion

- Hardware acceleration required for very low-latency MMSE-IA
- Resource requirements prohibitive for large system configurations
- Worst-case processing latency < 520 μs is achievable
Thank you for your attention!

Questions?