



Institut für Mikroelektronische Systeme



Leibniz  
Universität  
Hannover

# Hardwarebeschleuniger für Interference Alignment in In-House Mehrbenutzer-Kommunikationssystemen

Markus Kock, Holger Blume



ITG Workshop "Sound, Vision & Games", 22. September 2015, Hannover

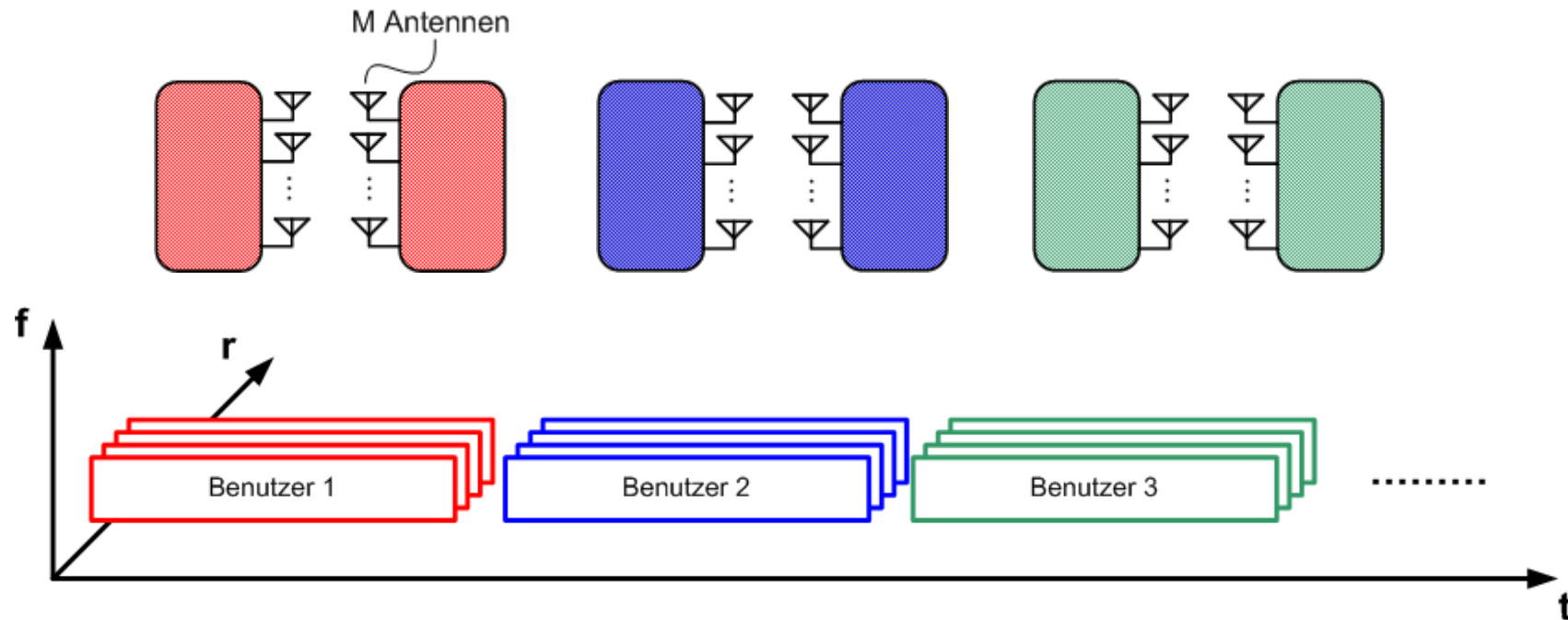
## Outline

- 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

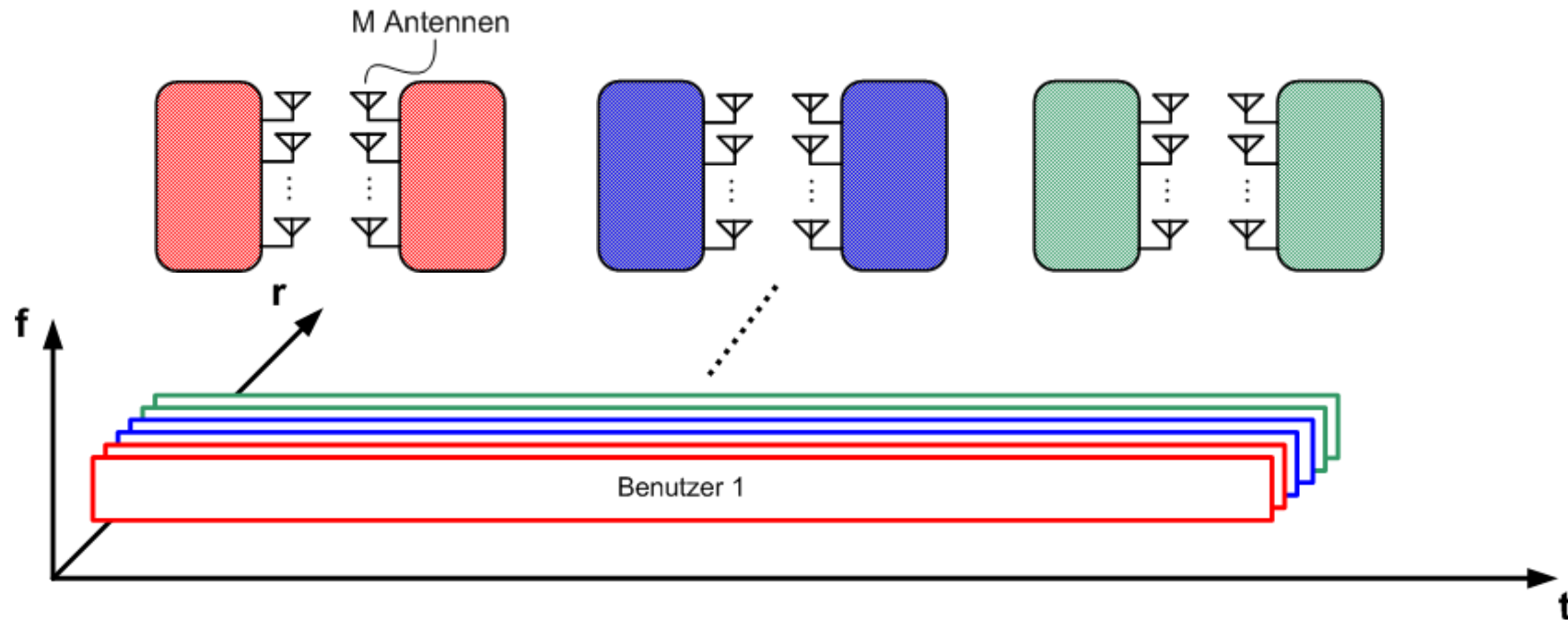
- Dedicated hardware accelerator for Minimum Mean Square Error (MMSE) Interference Alignment (IA)
- Digital baseband processing
- Low-latency real-time operation (latency  $< 1$  ms)

# Multi-User MIMO Communication System, TDMA



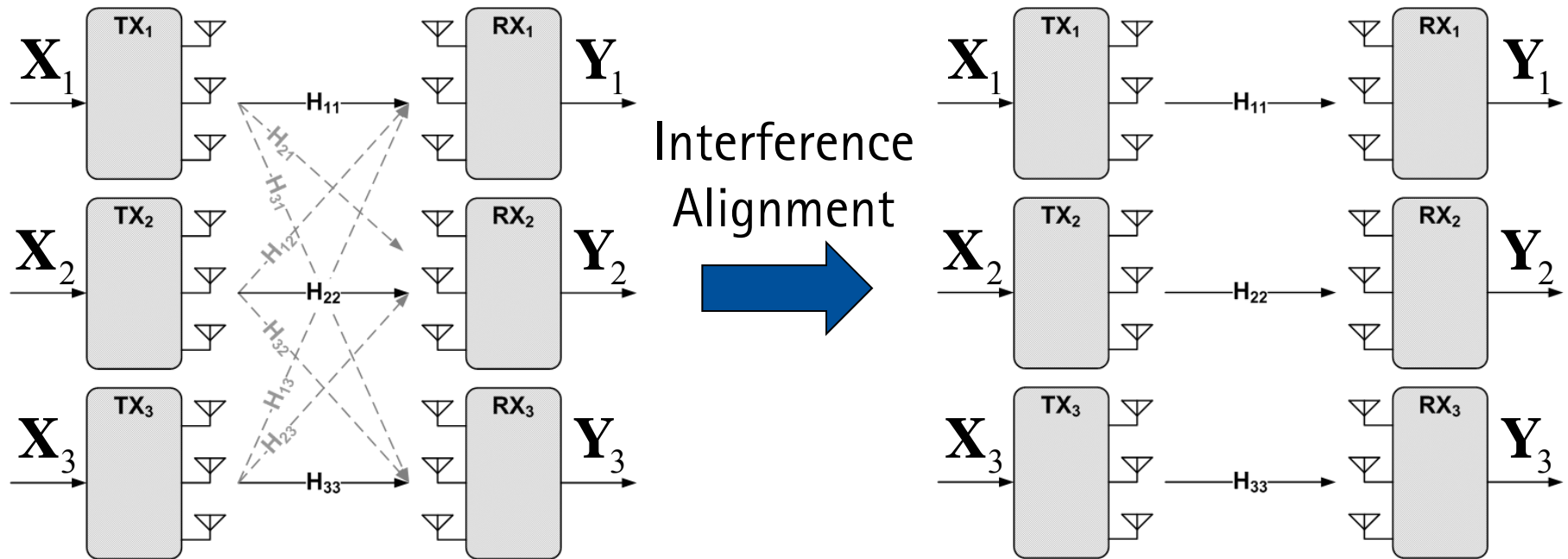
- MIMO spatial multiplexing: multiple antennas per user, one data stream per antenna
- Channel capacity shared by all users

# 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

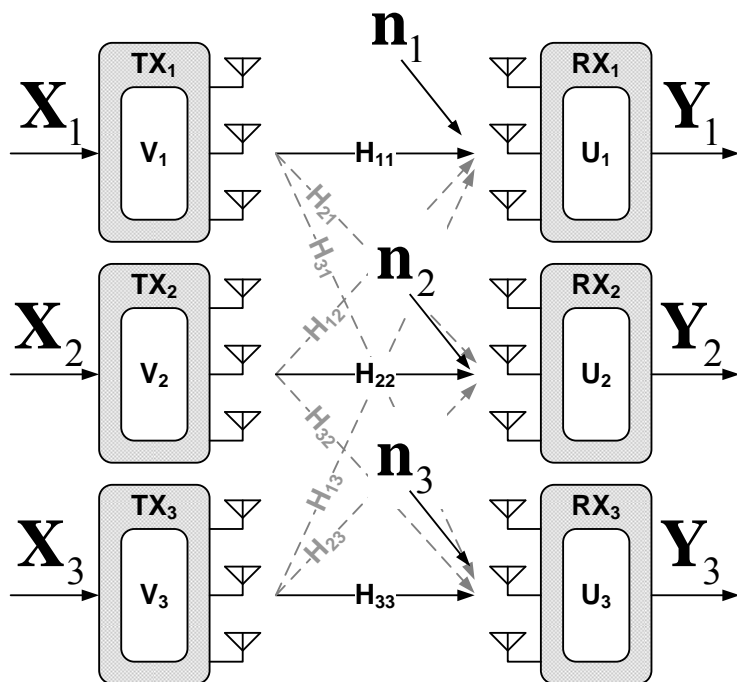


➔ Channel capacity scales with number of users  $K$

# Interference Alignment System Model

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

$$\text{Received signal: } \mathbf{Y}_k = \mathbf{U}_k^T \left( \sum_{j=1}^K \mathbf{H}_{kj} \mathbf{V}_j \mathbf{X}_j + \mathbf{n}_k \right)$$



- $K$ : #users,  $d$ : datastreams / user  
 $N_t$ : antennas / TX,  $N_r$ : antennas / RX
- Problem formulation:  
Determine  $\mathbf{V}_k$  and  $\mathbf{U}_k$  for given  $\mathbf{H}_{kj}$
- 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  $\mathbf{V}_k$  and  $\mathbf{U}_k$  ( $< 1$  ms)
- Additional system latencies: channel estimation, transmit CSI, distribute  $\mathbf{V}_k$  and  $\mathbf{U}_k$



## MMSE-IA Algorithm

- MMSE criterion: minimize overall interference + noise  
Algorithm<sup>[1]</sup>:

1. Start with arbitrary  $\mathbf{V}_k$

2. Update

$$\mathbf{U}_k = \left( \sum_{j=1}^K \mathbf{H}_{kj} \mathbf{V}_j \mathbf{V}_j^H \mathbf{H}_{kj}^H + \sigma^2 \mathbf{I} \right)^{-1} \mathbf{H}_{kk} \mathbf{V}_k$$

$$\mathbf{V}_k = \left( \sum_{j=1}^K \mathbf{H}_{jk}^H \mathbf{U}_j \mathbf{U}_j^H \mathbf{H}_{jk} + \lambda_k \mathbf{I} \right)^{-1} \mathbf{H}_{kk}^H \mathbf{U}_k$$

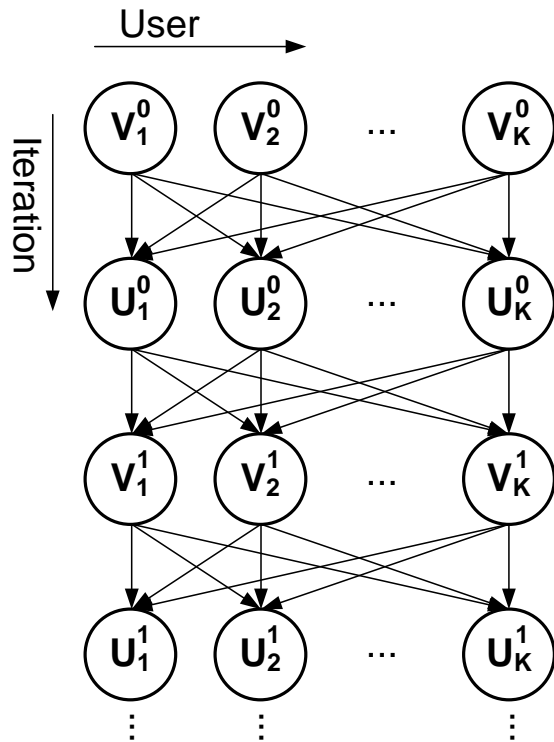
Lagrange multiplier  $\lambda_k$  iteratively determined  
to satisfy TX power constraint  $\|\mathbf{V}_k\|_F^2 \leq 1$

3. Compute system MSE
4. Repeat steps 2 and 3 until convergence

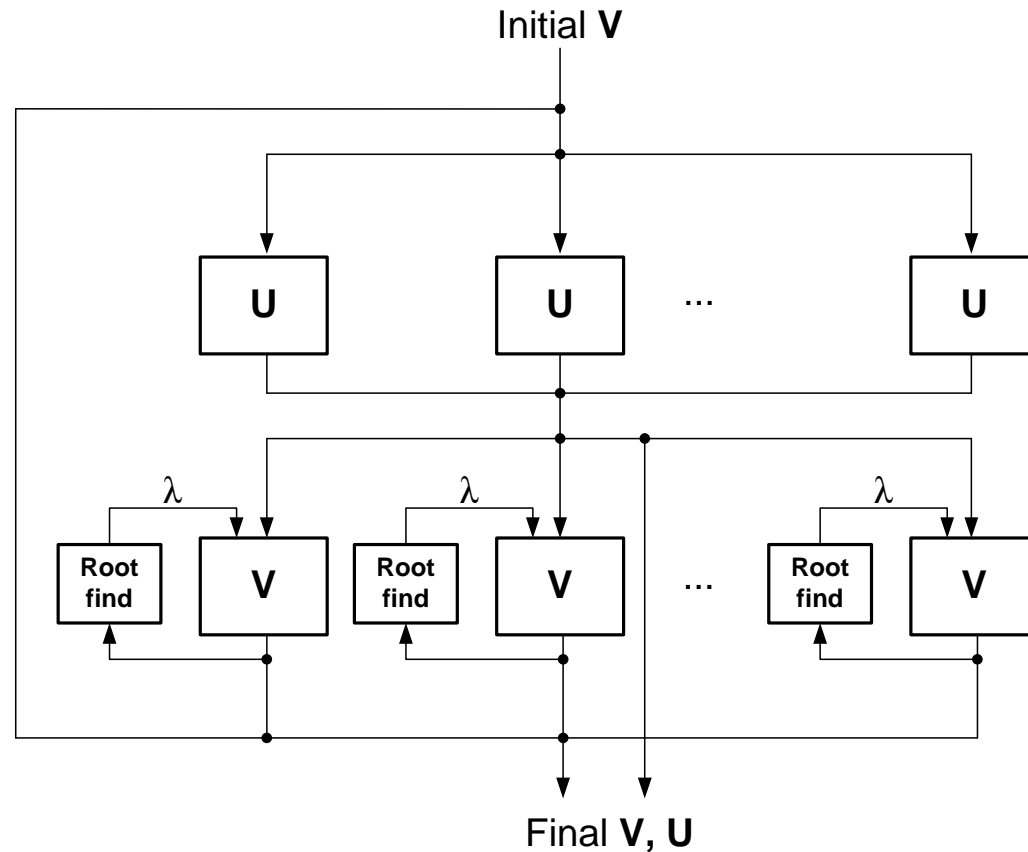
[1]: D. Schmidt, C. Shi, R. Berry, M. Honig, and W. Utschick, "Minimum Mean Squared Error Interference Alignment", 2009

# Parallelization

Inherent data dependencies limit parallelization



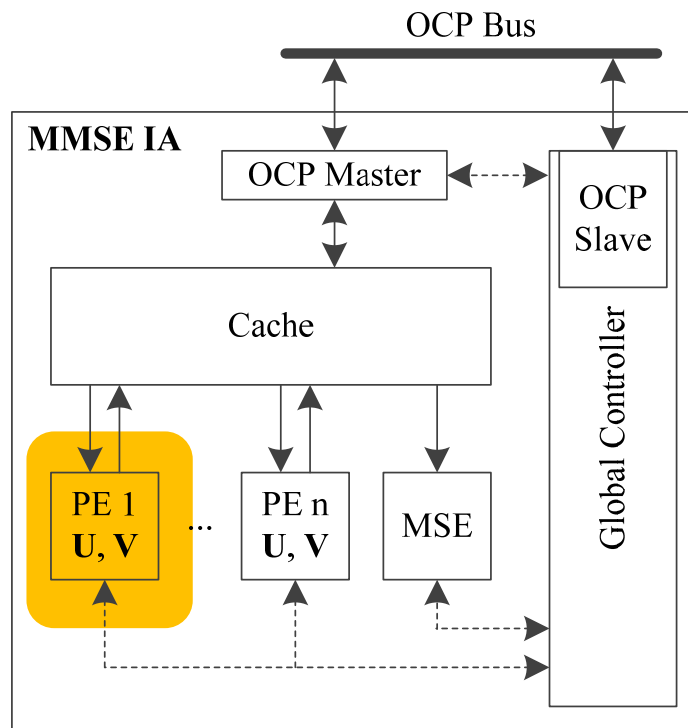
Data dependencies



Maximum parallel HW data flow

# Hardware System Architecture

Dedicated accelerator for integration in SDR SoCs

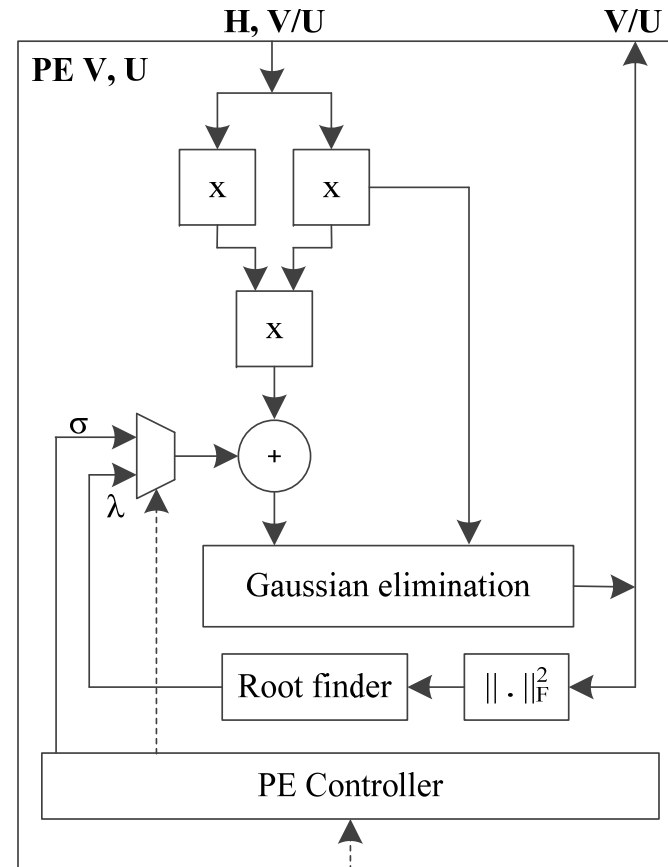


Top-level

- All communication via OCP or AXI on-chip busses
- Local matrix cache (BRAM)
  - $\mathbf{H}_{jk}$  channels
  - $\mathbf{V}_k$  precoders
  - $\mathbf{U}_k$  decoders
- Variable number of processing elements (PE) for computing  $\mathbf{V}_k$  and  $\mathbf{U}_k$
- 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



PE detail

## Low-Latency Equation System Solver

- Inner loop contains matrix inversion
  - Solve equation system instead

$$\mathbf{U}_k = \left( \sum_{j=1}^K \mathbf{H}_{kj} \mathbf{V}_j \mathbf{V}_j^H \mathbf{H}_{kj}^H + \sigma^2 \mathbf{I} \right)^{-1} \mathbf{H}_{kk} \mathbf{V}_k = \mathbf{Q}_k^{-1} \mathbf{B}_k$$

$$\mathbf{Q}_k \mathbf{U}_k = \mathbf{B}_k$$

- Candidates: SVD, LU, QR, ...
- Criterion: low-latency
  - Gaussian elimination
    - Small matrix sizes → 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

$$\mathbf{Q}_k \mathbf{U}_k = \mathbf{B}_k$$

Augmented System:

$$[\mathbf{Q}_k \mid \mathbf{B}_k] = \left[ \begin{array}{cccc|ccc} q_{11} & q_{12} & \cdots & q_{1N} & b_{11} & \cdots & b_{1d} \\ q_{21} & q_{22} & \cdots & q_{2N} & b_{21} & \cdots & b_{2d} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \cdots & \vdots \\ q_{N1} & q_{N2} & \cdots & q_{NN} & b_{N1} & \cdots & b_{Nd} \end{array} \right] \xrightarrow{\text{Bareiss}} [\mathbf{I} \mid \mathbf{U}_k]$$

## Two-Step Bareiss Algorithm Result

- Two unknowns eliminated after one step

$$\left[ \begin{array}{cccc|ccc} \tilde{q}_{11} & 0 & \cdots & \tilde{q}_{1N} & \tilde{b}_{11} & \cdots & \tilde{b}_{1d} \\ 0 & \tilde{q}_{22} & \cdots & \tilde{q}_{2N} & \tilde{b}_{21} & \cdots & \tilde{b}_{2d} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & \tilde{q}_{NN} & \tilde{b}_{N1} & \cdots & \tilde{b}_{Nd} \end{array} \right]$$

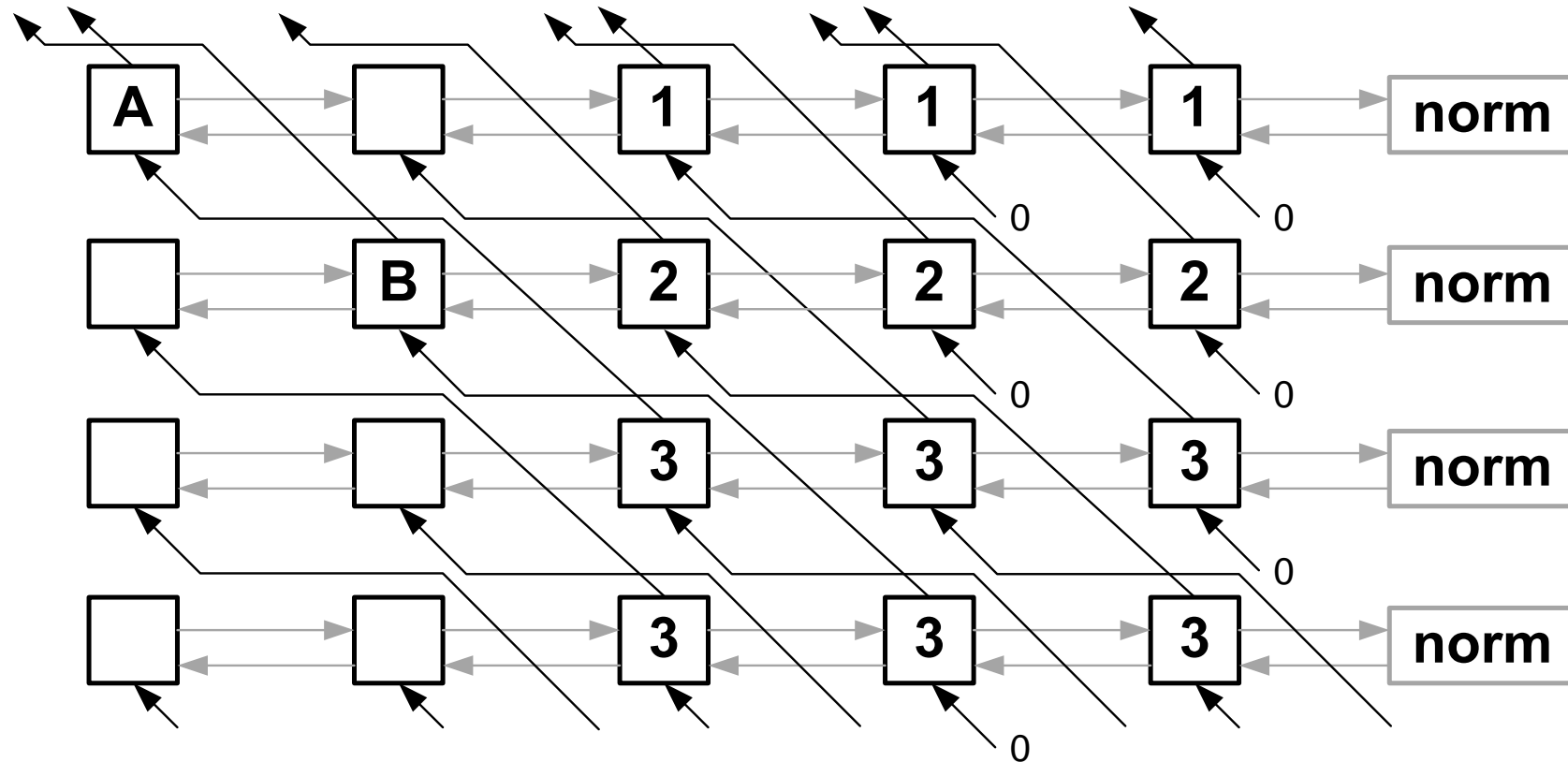
One common factor per row  
 → Skip multiplication  
 Fewer operations compared  
 to single step elimination

- Repeat to obtain diagonal form

$$\left[ \begin{array}{ccc|ccc} d_{11} & & 0 & \tilde{u}_{11} & \cdots & \tilde{u}_{1d} \\ & d_{22} & & \tilde{u}_{21} & \cdots & \tilde{u}_{2d} \\ & & \ddots & \vdots & \cdots & \vdots \\ & 0 & & \tilde{u}_{N1} & \cdots & \tilde{u}_{Nd} \end{array} \right]$$

Final division required for  
 each result coefficient  $u$

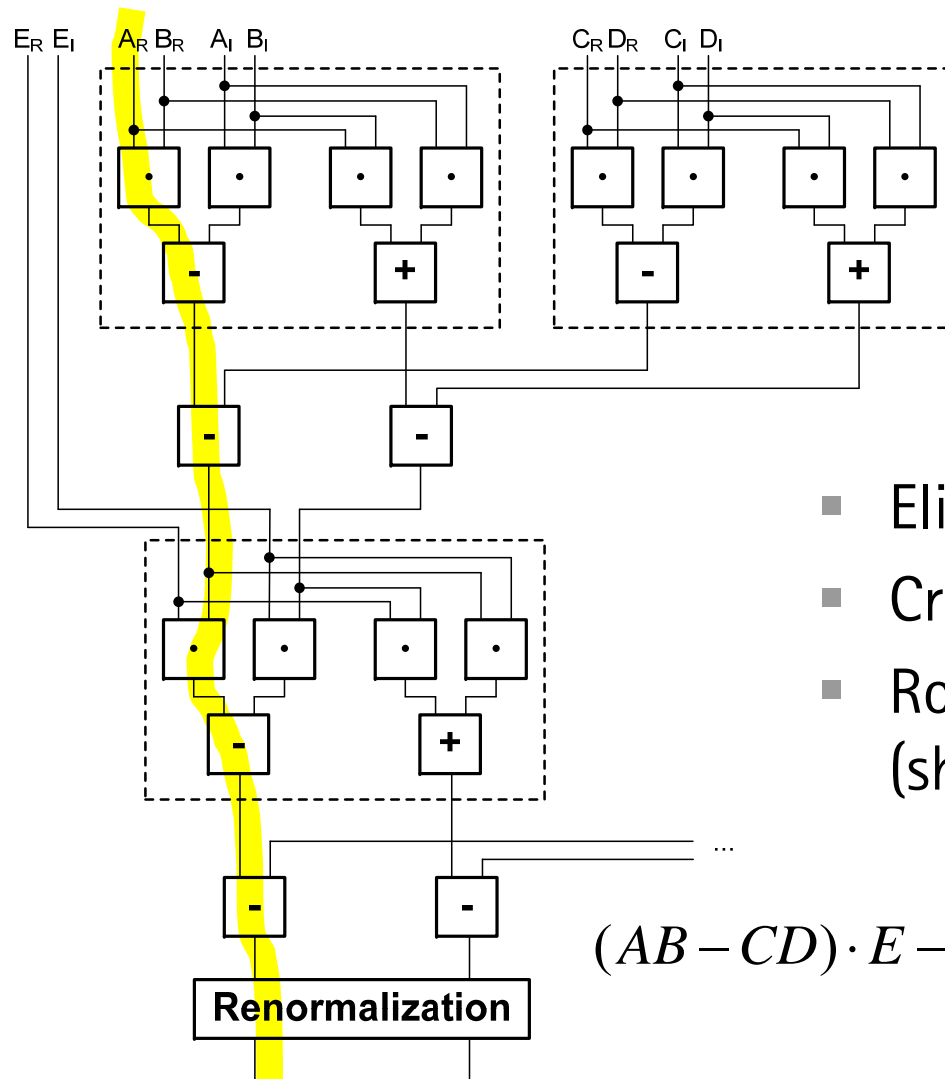
## Two-Step Bareiss Systolic Array Processor



- 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

$$(AB - CD) \cdot E - \dots$$

**Renormalization**

## 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  $\mu$ s for worst-case system ( $N_t = N_r = 11, K = 19$ )

K	$N_{t,r}$	d	$n_{PE}$	$n_{MM}$	FF		LUT		DSP48E1	
3	2	1	3	3	51531	7.50%	81560	23.73%	364	42.13%
			1		20942	3.05%	32702	9.52%	148	17.13%
			1	16585	2.41%	25682	7.47%	80	9.26%	
5	3	1	1	3	32980	4.80%	56974	16,58%	330	38,19%
			1	1	24916	3.62%	43514	12.66%	232	26,85%

## Conclusion

- Hardware acceleration required for very low-latency MMSE-IA
- Resource requirements prohibitive for large system configurations
- Worst-case processing latency  $< 520 \mu\text{s}$  is achievable

Thank you for your attention!

Questions?