Rapid Prototyping of a Fixed-Throughput Sphere Decoder for MIMO Systems

Luis G. Barbero, John S. Thompson

Outline

- Introduction
- MIMO System
- Rapid Prototyping
- Sphere Decoder (SD)
- Fixed-Sphere Decoder (FSD)
- Conclusion and Present Work

Introduction

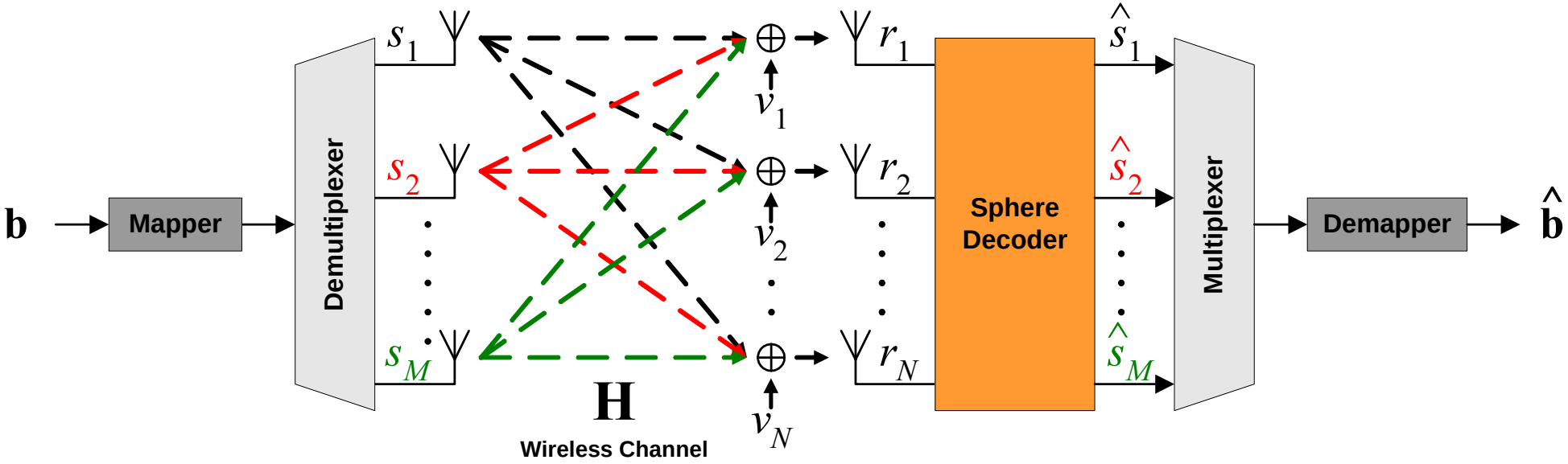
- The use of **multiple antennas** at both transmitter and receiver (MIMO) significantly increases the capacity and spectral efficiency of wireless communication systems.
- The **sphere decoder** provides optimal maximum-likelihood performance in MIMO detection with reduced complexity compared to the maximum-likelihood detector.
- However, it has a **variable complexity**, depending on the noise level and the channel conditions, that hinders its integration into a complete communication system.
- A **fixed-sphere decoder** is proposed to achieve quasi-ML performance with fixed-complexity resulting in a fully-pipelined hardware implementation.
- **Rapid prototyping** is a design methodology where a system-level design, specified in a high-level description language, is quickly translated to a hardware implementation.

MIMO System

- Wireless communication system equipped with M transmit and N receive antennas, denoted as $M \times N$.
- Capacity increase compared to single-antenna systems if the different paths between transmitter and receiver are independent .
  - Improve link quality (BER) $\rightarrow$ Space-Time Coding
  - Increase data rate (bps) $\rightarrow$ Spatial Multiplexing

MIMO System

- Wireless communication system equipped with M transmit and N receive antennas, denoted as $M \times N$.
- Capacity increase compared to single-antenna systems if the different paths between transmitter and receiver are independent .
  - Improve link quality (BER) $\rightarrow$ Space-Time Coding
  - Increase data rate (bps) $\rightarrow$ Spatial Multiplexing



MIMO System Notation

- Assuming symbol-synchronous sampling and ideal timing at the receiver, the received N-vector, using matrix notation, is given by

$$\mathbf{r} = \mathbf{H}\mathbf{s} + \mathbf{v}$$

  - $\mathbf{s} = (s_1, s_2, ..., s_M)^T$: vector P-QAM transmitted symbols with $\mathrm{E}[|s_i|^2] = 1/M$
  - $\mathbf{v} = (v_1, v_2, ..., v_N)^T$: vector i.i.d. complex Gaussian noise samples with $\sigma^2 = N_0$
  - $\mathbf{r} = (r_1, r_2, ..., r_N)^T$: vector received symbols
  - $\mathbf{H}$: $N \times M$ i.i.d. Rayleigh fading channel matrix

    h_{ij} is the complex transfer function from transmitter j to receiver i with $\mathrm{E}[|h_{ij}|^2] = 1$
  - $N \geq M$
  - Number of possible transmitted M-vectors $= P^M$ (i.e. complex constellation $\mathcal{C}$)

Rapid Prototyping System

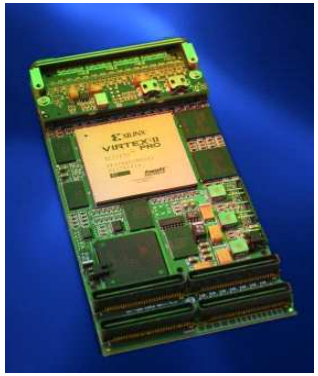
- Prototyping system that concentrates on the analysis of the MIMO algorithm.
- Analysis and hardware implementation of novel algorithms.
- Requirements:
  - Reconfigurable hardware platform
  - Methodology that does not require detailed knowledge of the underlying hardware.
  - Uniform testing environment.
  - Real-time testing $\rightarrow$ Hardware in the loop.
  - Simple and flexible interface between the hardware platform and the host.

Rapid Prototyping System

- Prototyping system that concentrates on the analysis of the MIMO algorithm.
- Analysis and hardware implementation of novel algorithms.
- Requirements:
  - Reconfigurable hardware platform
  - Methodology that does not require detailed knowledge of the underlying hardware.
  - Uniform testing environment.
  - Real-time testing → Hardware in the loop.
  - Simple and flexible interface between the hardware platform and the host.

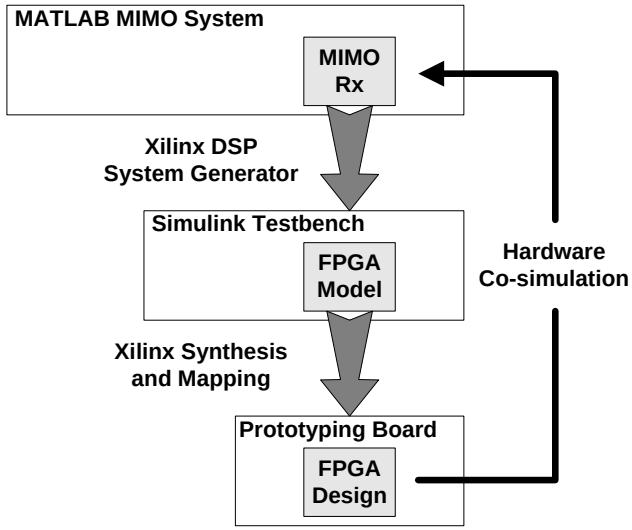
Hardware Platform

- Provided by Alpha Data Ltd. It consists of:
  - PCI Adapter
  - 2 FPGA boards: Xilinx Virtex II (XC2V4000) and Xilinx Virtex II Pro (XC2VP70)



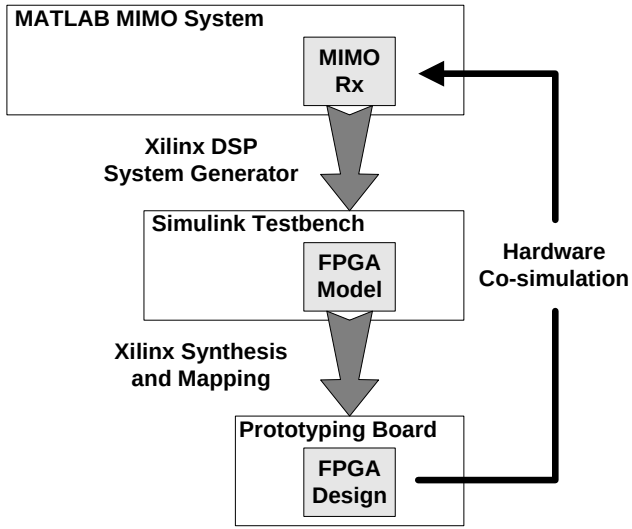
Rapid Prototyping Methodology

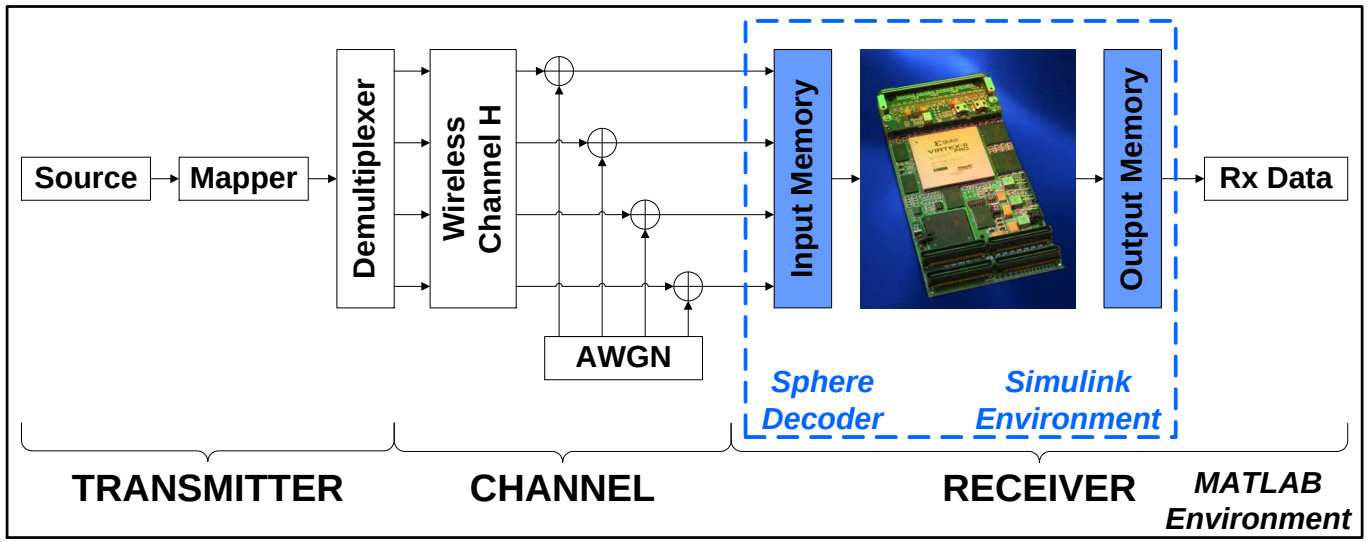
- MATLAB MIMO system for algorithm evaluation.
- Simulink testbench for implementation and debugging using Xilinx's DSP System Generator.
- Xilinx's Integrated Software Environment to generate FPGA bitstream.
- Integration of the FPGA implementation into the MATLAB MIMO system for real-time hardware co-simulation (4×4).



Rapid Prototyping Methodology

- MATLAB MIMO system for algorithm evaluation.
- Simulink testbench for implementation and debugging using Xilinx's DSP System Generator.
- Xilinx's Integrated Software Environment to generate FPGA bitstream.
- Integration of the FPGA implementation into the MATLAB MIMO system for real-time hardware co-simulation (4×4).





Sphere Decoder

- Sphere Decoder Algorithm
- Simulation Results
- FPGA Implementation
- FPGA Results

1. L. G. Barbero and J. S. Thompson, “Rapid prototyping of the sphere decoder for MIMO systems,” in *Proc. IEE/EURASIP Conference on DSP Enabled Radio (DSPeR ’05)*, vol. 1, Southampton, UK, Sept. 2005, pp. 41–47.
2. L. G. Barbero and J. S. Thompson, “Rapid prototyping system for the evaluation of MIMO receive algorithms,” in *Proc. IEEE International Conference on Computer as a Tool (EUROCON ’05)*, vol. 2, Belgrade, Serbia and Montenegro, Nov. 2005, pp. 1779–1782.

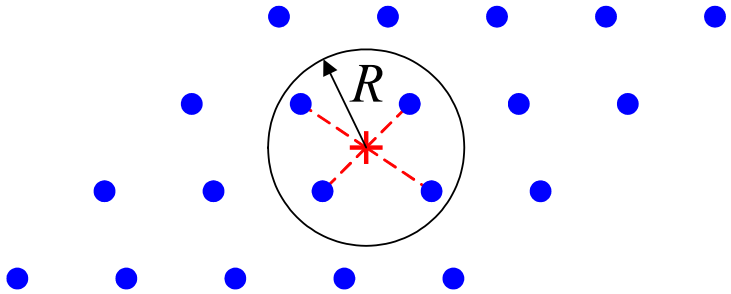
Sphere Decoder (SD)

- Maximum-likelihood performance with reduced complexity.
- Search over only the noiseless received points (defined as the lattice $\mathbf{Hs}$) that lie within a hypersphere of radius R around the received signal $\mathbf{r}$.

$$\hat{\mathbf{s}}_{\mathbf{ml}} = \arg\{\min_{\mathbf{s}} \|\mathbf{r} - \mathbf{Hs}\|^2 \leq R^2\}$$

  - Real SD: widely used, only for QAM constellations. Equivalent real decomposition of the system.

  $\Rightarrow$ Complex SD: more recent version, can be applied to any constellation. More optimized hardware implementation.



Sphere Decoder (SD)

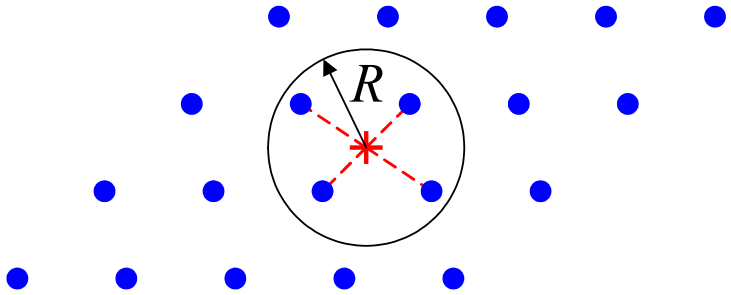
- Maximum-likelihood performance with reduced complexity.
- Search over only the noiseless received points (defined as the lattice $\mathbf{Hs}$) that lie within a hypersphere of radius R around the received signal $\mathbf{r}$.

$$\hat{\mathbf{s}}_{\mathbf{ml}} = \arg\{\min_{\mathbf{s}} \|\mathbf{r} - \mathbf{Hs}\|^2 \leq R^2\}$$

  - Real SD: widely used, only for QAM constellations. Equivalent real decomposition of the system.

  $\Rightarrow$ Complex SD: more recent version, can be applied to any constellation. More optimized hardware implementation.

- Initial radius R is selected according to the noise variance per antenna σ^2.
- The complexity of the algorithm depends on the noise level and the channel conditions.



Sphere Decoder Algorithm (1)

- The sphere constraint (SC) can be written as

$$\hat{\mathbf{s}}_{\mathbf{ml}} = \arg\{\min_{\mathbf{s}} \|\mathbf{U}(\mathbf{s} - \hat{\mathbf{s}})\|^2 \leq R^2\}$$

  - $\mathbf{U}$: $M \times M$ upper triangular matrix, Cholesky decomposition of Gram matrix $\mathbf{G} = \mathbf{H}^H\mathbf{H}$
  - $\hat{\mathbf{s}} = \mathbf{H}^\dagger\mathbf{r}$: least squares estimate of $\mathbf{s}$
  - $\mathbf{H}^\dagger = (\mathbf{H}^H\mathbf{H})^{-1}\mathbf{H}^H$: pseudoinverse of $\mathbf{H}$

Sphere Decoder Algorithm (1)

- The sphere constraint (SC) can be written as

$$\hat{\mathbf{s}}_{\mathbf{ml}} = \arg\{\min_{\mathbf{s}} \|\mathbf{U}(\mathbf{s} - \hat{\mathbf{s}})\|^2 \leq R^2\}$$

  - $\mathbf{U}$: $M \times M$ upper triangular matrix, Cholesky decomposition of Gram matrix $\mathbf{G} = \mathbf{H}^H\mathbf{H}$
  - $\hat{\mathbf{s}} = \mathbf{H}^\dagger\mathbf{r}$: least squares estimate of $\mathbf{s}$
  - $\mathbf{H}^\dagger = (\mathbf{H}^H\mathbf{H})^{-1}\mathbf{H}^H$: pseudoinverse of $\mathbf{H}$

- Solution is obtained recursively for $i = M, ..., 1$. For each level, the points s_i selected as candidates satisfy

$$|s_i - z_i|^2 \leq \frac{T_i}{u_{ii}^2}$$

  where

$$z_i = \hat{s}_i - \sum_{j=i+1}^{M} \frac{u_{ij}}{u_{ii}}(s_j - \hat{s}_j)$$

$$T_i = R^2 - \sum_{j=i+1}^{M} u_{jj}^2 |s_j - z_j|^2$$

Sphere Decoder Algorithm (2)

- The algorithm can be seen as a tree search through M levels with a metric constraint R^2. Each node on the tree has P branches.

$$D_i = \underbrace{u_{ii}^2 |s_i - z_i|^2}_{d_i} + \underbrace{\sum_{j=i+1}^{M} u_{jj}^2 |s_j - z_j|^2}_{D_{i+1}} \leq R^2$$

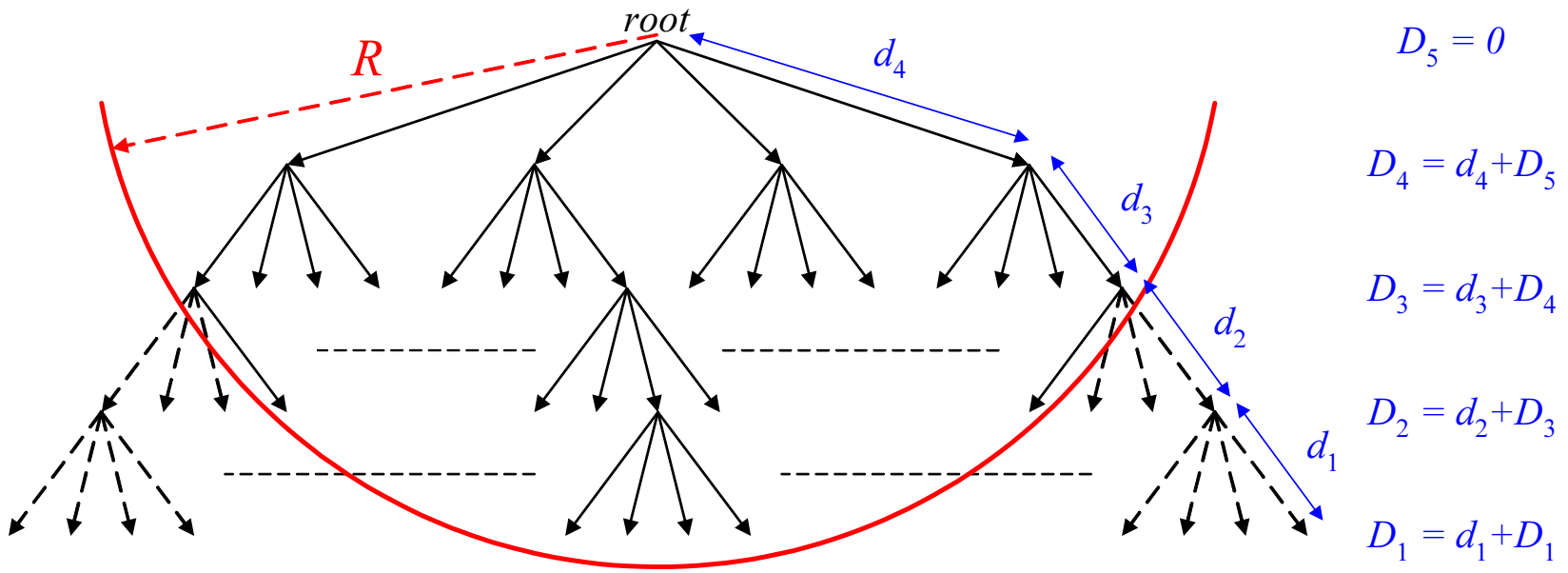
  - D_{i+1}: accumulated (squared) Euclidean distance from the root down to level $i+1$
  - d_i: partial (squared) Euclidean distance from level i

Sphere Decoder Algorithm (2)

- The algorithm can be seen as a tree search through M levels with a metric constraint R^2. Each node on the tree has P branches.

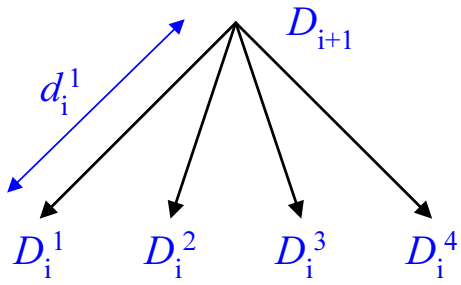
$$D_i = \underbrace{u_{ii}^2 |s_i - z_i|^2}_{d_i} + \underbrace{\sum_{j=i+1}^{M} u_{jj}^2 |s_j - z_j|^2}_{D_{i+1}} \leq R^2$$

  - D_{i+1}: accumulated (squared) Euclidean distance from the root down to level $i+1$
  - d_i: partial (squared) Euclidean distance from level i



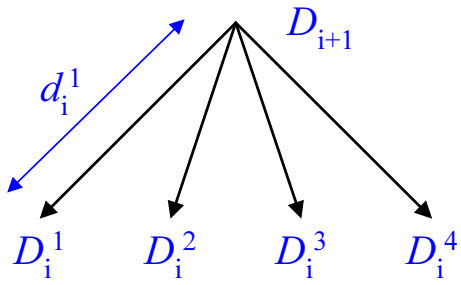
Branch Enumeration

- The order in which we follow the branches from one particular node on any level affects the final complexity.
  - Accumulated distance D_{i+1} on level $i+1$
  - P values $D_i^p = d_i^p + D_{i+1}$ $(p = 1, ..., P)$



Branch Enumeration

- The order in which we follow the branches from one particular node on any level affects the final complexity.
  - Accumulated distance D_{i+1} on level $i+1$
  - P values $D_i^p = d_i^p + D_{i+1}$ $(p = 1, ..., P)$



- **Pohst** enumeration: the different branches are visited in an arbitrary constellation order.
- **Schnorr-Euchner** enumeration: the different branches are visited in increasing order of D_i^p (i.e. increasing distance between s_i^p and z_i).
- Advantages of the Schnorr-Euchner enumeration:
  - It reduces the complexity.
  - The complexity is independent of the initial radius.

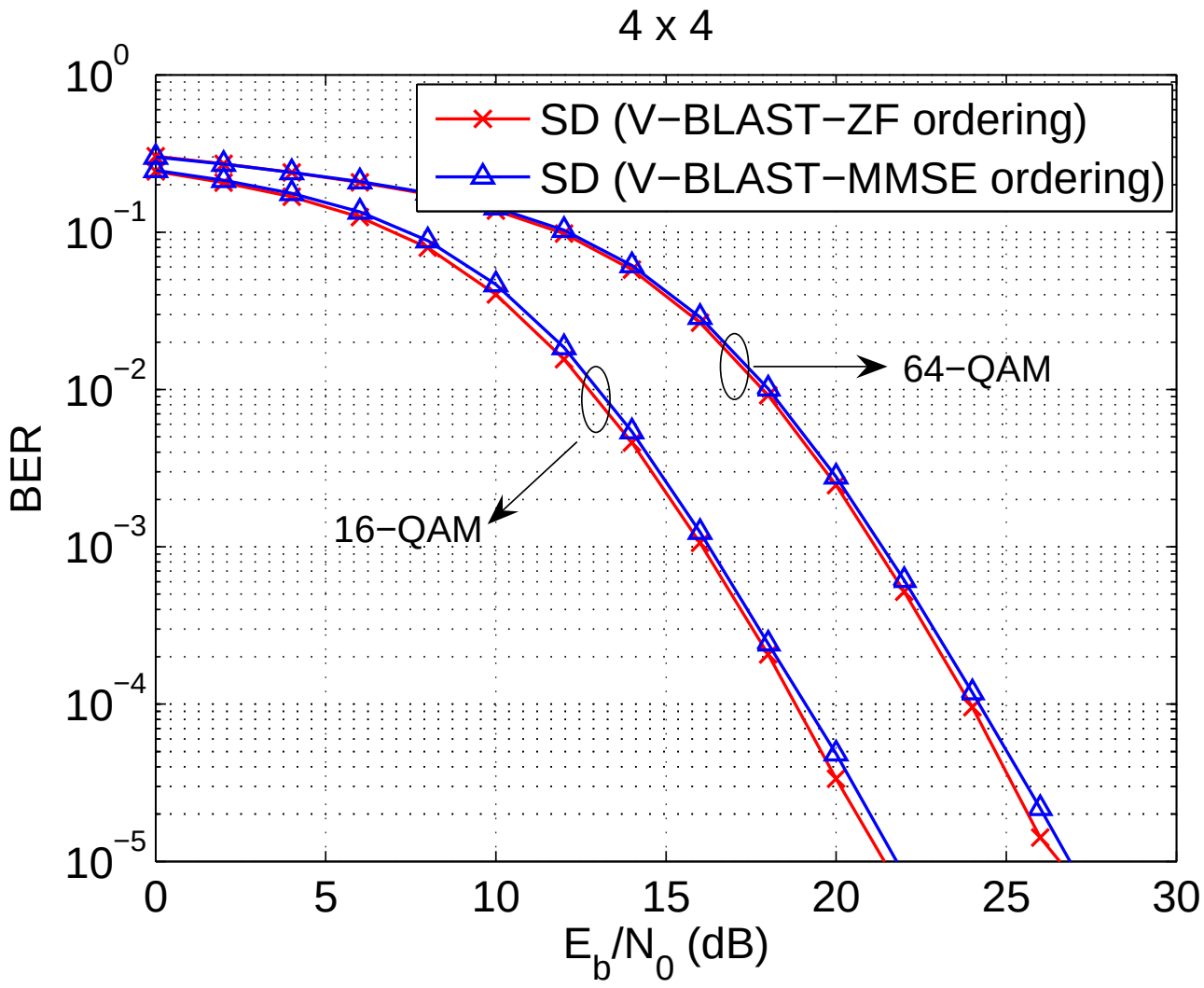
Channel Matrix Ordering

- Ordering of the columns of the channel matrix $\mathbf{H}$, resulting in a complexity reduction of the search stage of the sphere decoder.
- Increase in complexity due to the preprocessing could be considered negligible for packet-based communications.
- Two iterative methods suitable for hardware implementation:
  1. **V-BLAST-ZF**: the signals (i.e. levels i) are detected according to increasing order of the noise amplification (i.e. Euclidean norm of the rows of $\mathbf{H}^{\dagger}$).
  2. **V-BLAST-MMSE**: the signals (i.e. levels i) are detected according to decreasing order of the SINR.

$$\text{SINR}_i = \frac{|\tilde{\mathbf{h}}_i^{\dagger}\mathbf{h}_i|^2}{|\tilde{\mathbf{h}}_i^{\dagger}|^2\sigma^2 M + \sum_{\forall j \neq i, \mathbf{i}_{pre}} |\tilde{\mathbf{h}}_i^{\dagger}\mathbf{h}_j|^2}$$

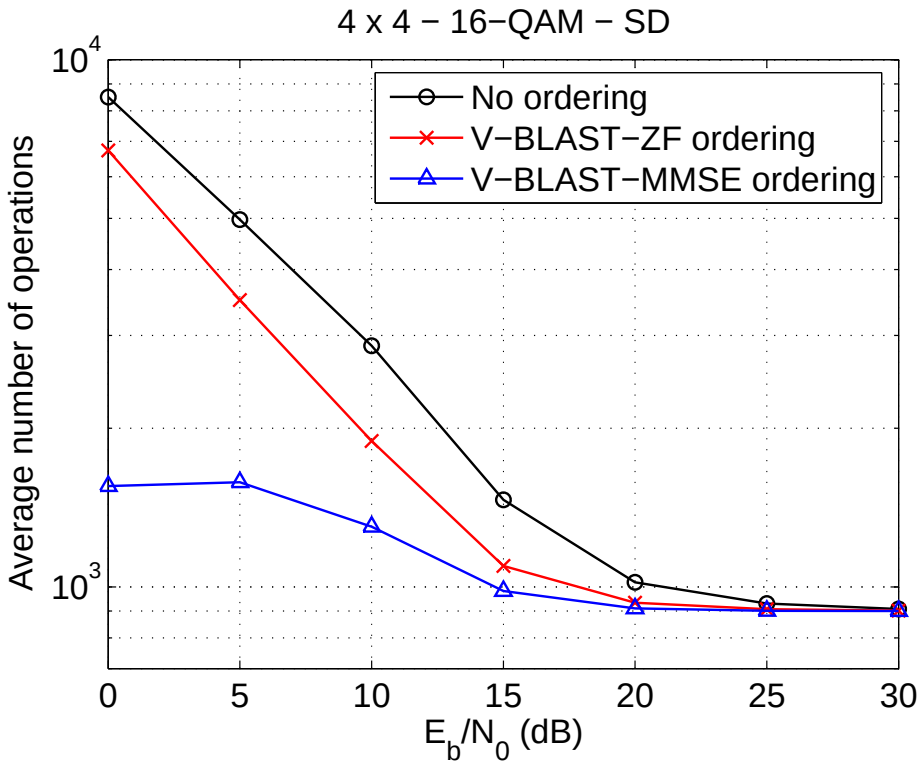
V-BLAST-MMSE largest complexity reduction but performance degradation.

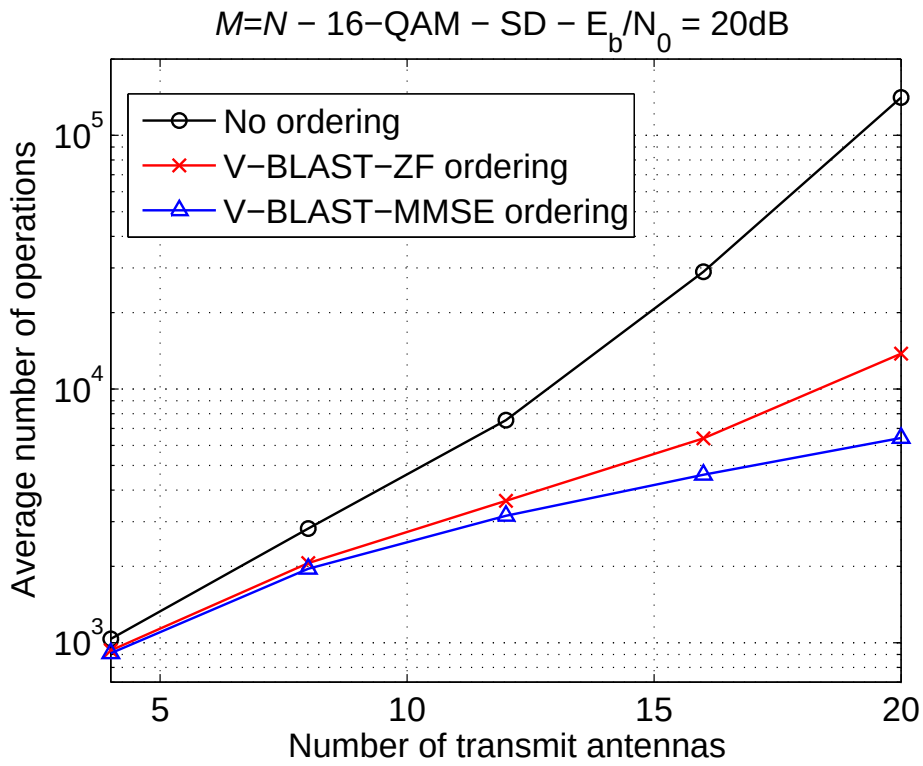
Sphere Decoder Performance



Sphere Decoder Complexity

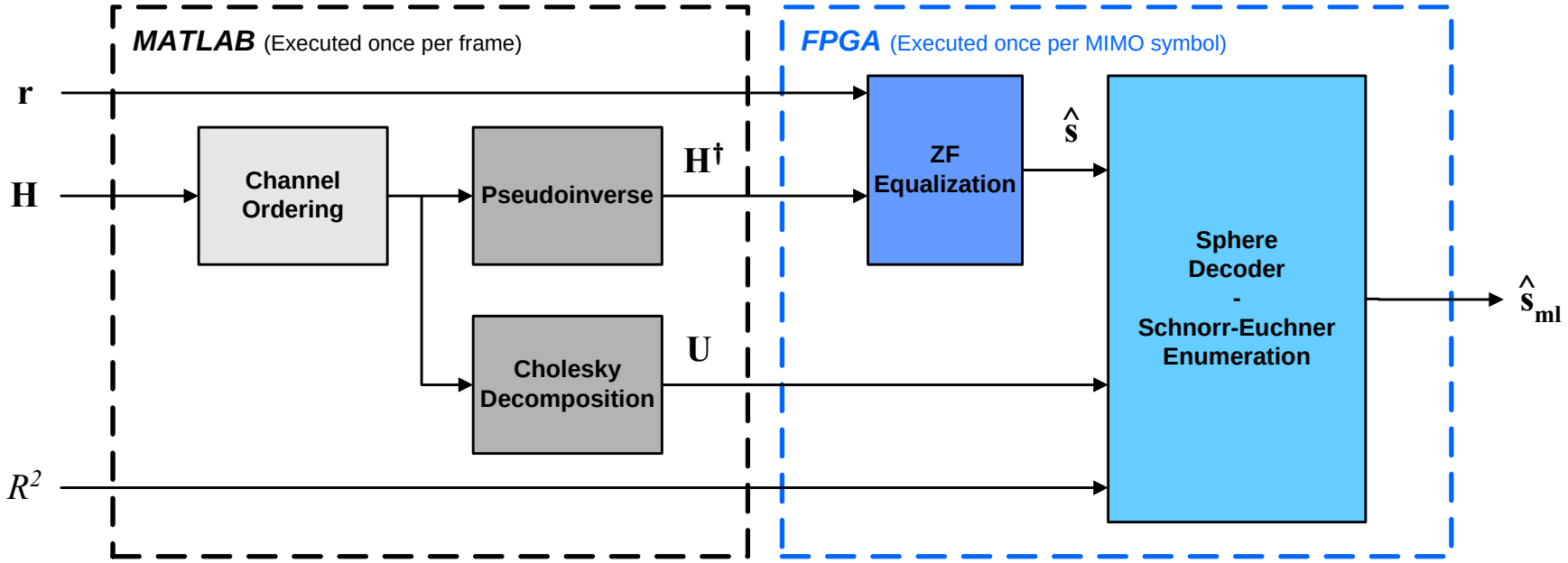
- Schnorr-Euchner enumeration



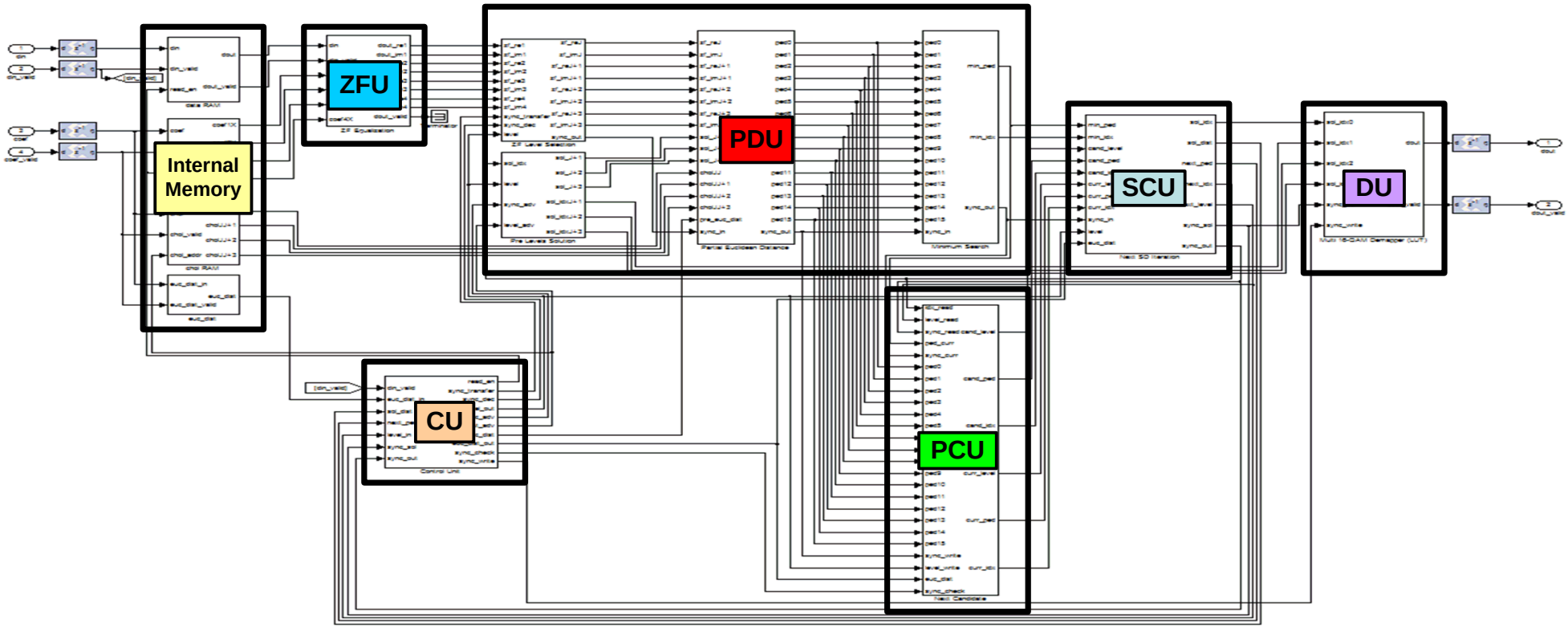


FPGA Implementation (SD)

- FPGA performs the tree search of the sphere decoder.
- MATLAB performs parts of the algorithm that are required once per frame:
  - Ordering of the channel matrix
  - Pseudoinverse
  - Cholesky decomposition
- MATLAB and FPGA are synchronized through a Simulink memory interface.

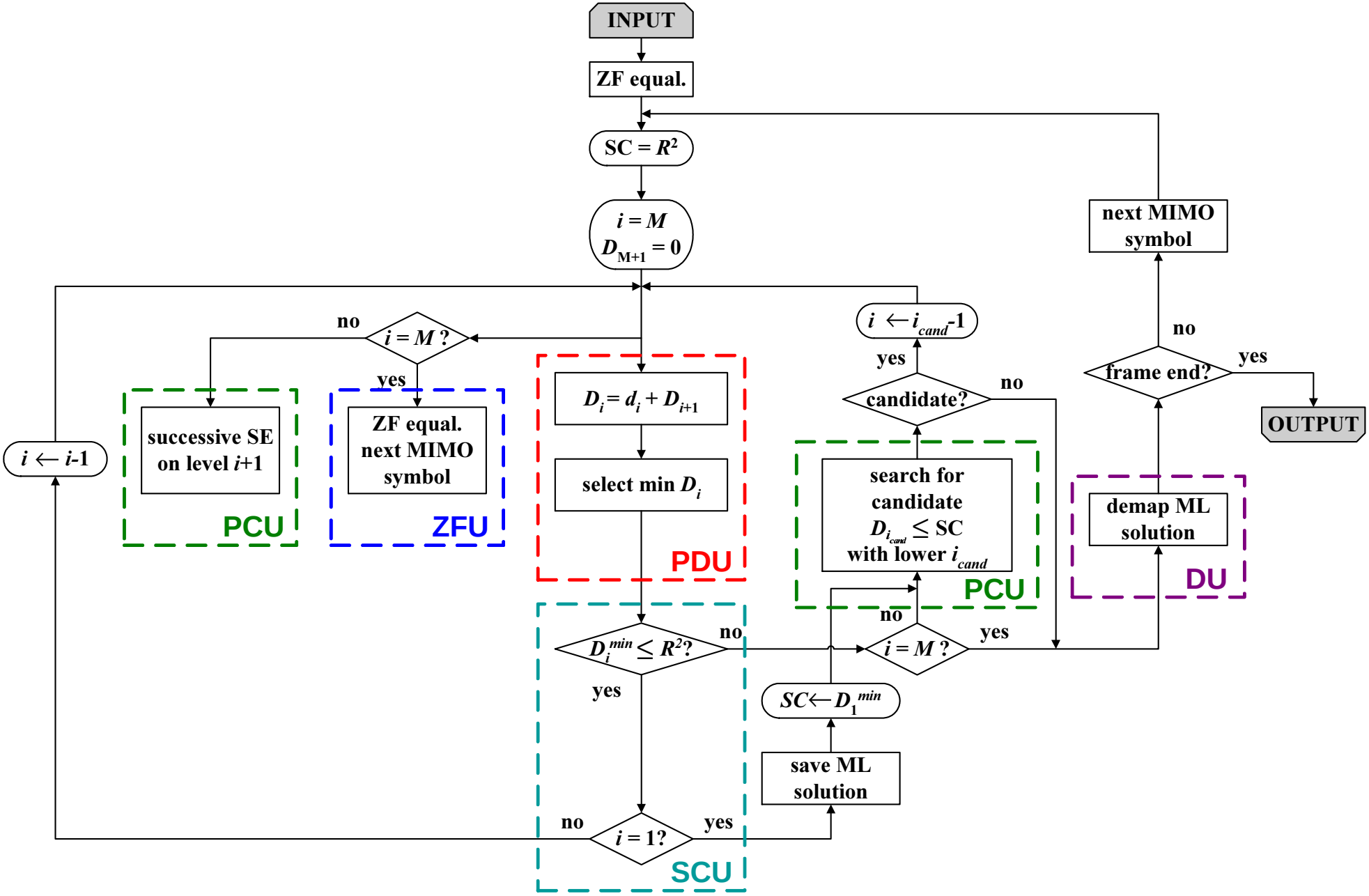


FPGA Architecture (SD)

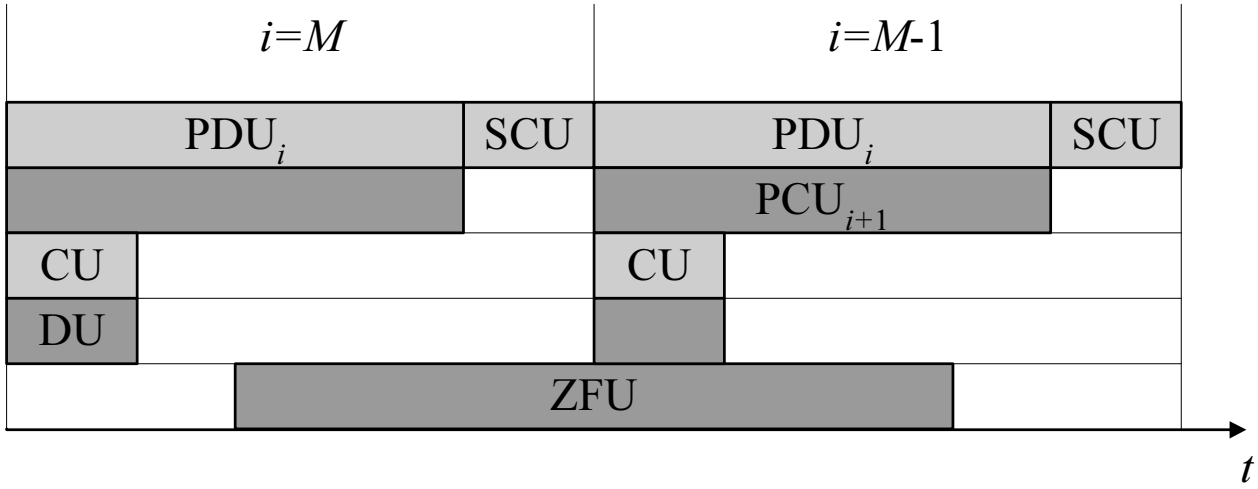


- ZFU: Zero Forcing Unit
- PDU: Partial Distance Unit
- PCU: Partial Candidates Unit
- SCU: Sphere Constraint Unit
- CU: Control Unit
- DU: Demapper Unit

Sphere Decoder Flowchart



FPGA Scheduling (SD)

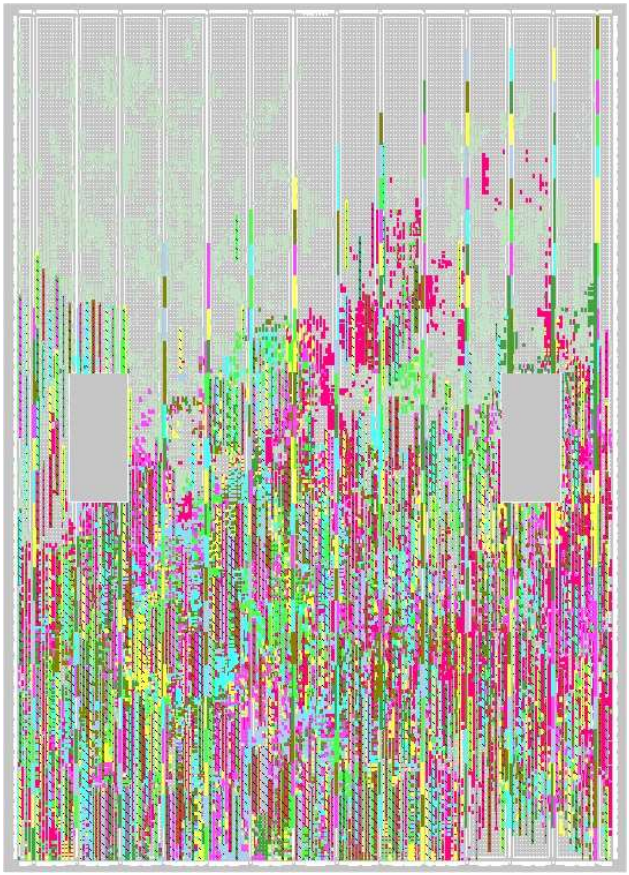


- Light grey: blocks executed in every iteration.
- Dark grey: blocks not executed in every iteration.
- White spaces: unused hardware resources of the design.
- Critical path formed by PDU + SCU → determines the throughput.
- Sphere decoder can not be fully pipelined → suboptimum use of the resources.

FPGA Results (SD)

- 4 sphere decoders in parallel.

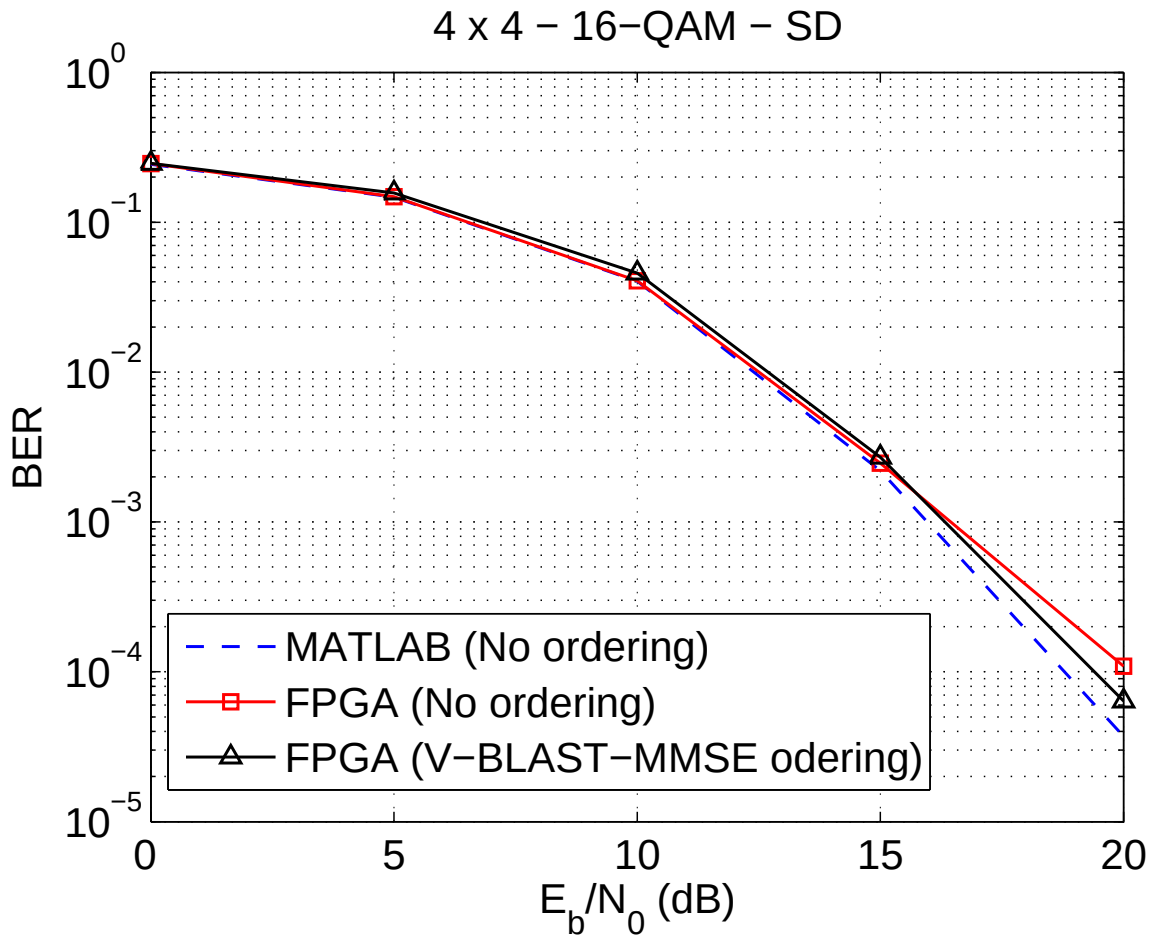
XC2VP70	Use	Percentage
Number of slices	21,467 / 33,088	64%
Number of flip-flops	17,691 / 66,176	26%
Number of 4-input LUTs	36,249 / 66,176	54%
Number of 18x18 multipliers	156 / 328	47%
Number of 18Kb block RAM	183 / 328	55%



- Block RAM: synchronization buffers and internal buffers of the sphere decoder.
- Embedded multipliers: computational complexity.
  - 39 per sphere decoder: 16 (ZFU) + 23 (PDU)
- LUTs: adders and control logic of the sphere decoder.
- Flip-flops: delay nets in the sphere decoder.

FPGA BER Performance (SD)

- Monte Carlo simulations with real-time hardware co-simulation.
- 16 bits used for real and imaginary components quantization.
- R^2 set to end of scale.

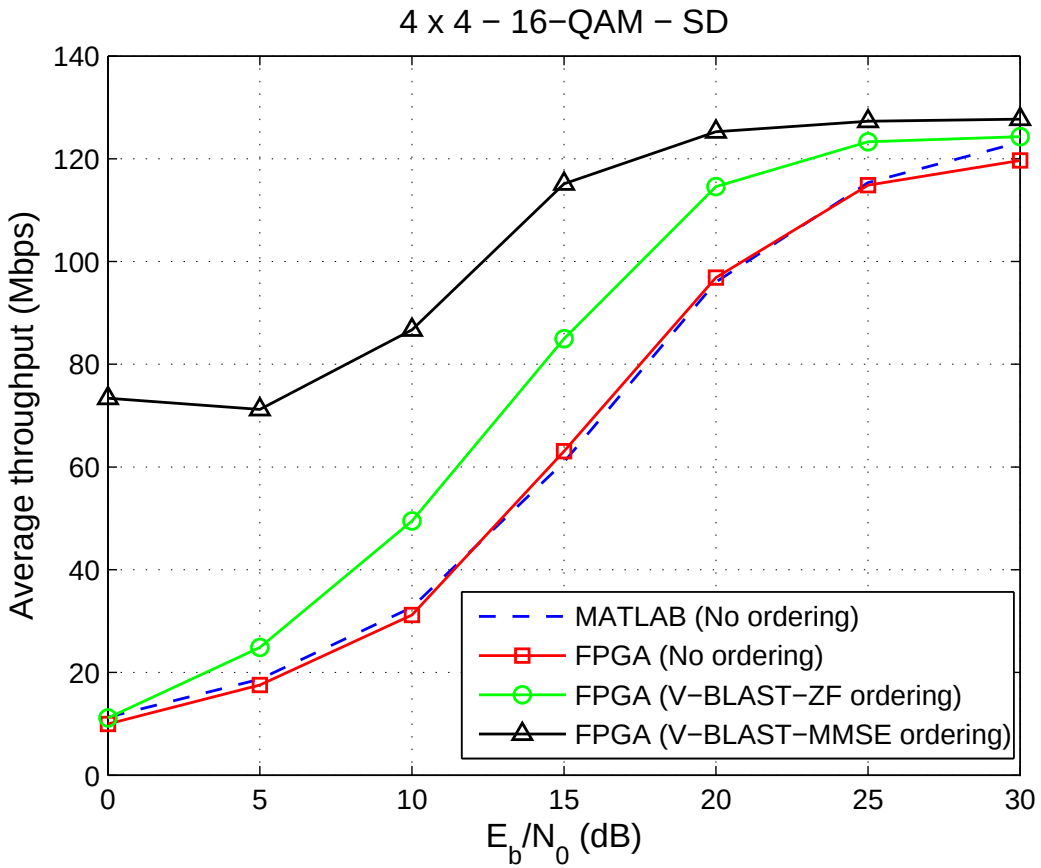


FPGA Throughput Performance (SD)

- Average throughput depends on the clock frequency and the number of cycles per detection.

$$Q_{avg} = 4 \cdot M \cdot \log_2 P \cdot f_{clock} \,/\, C_{avg} \quad \text{(Mbps)}$$

- $f_{clock} = 50$ MHz, $C_{min} = 25$ cycles $\rightarrow Q_{max} = 128$ Mbps



Fixed-Sphere Decoder (FSD)

- Fixed-Sphere Decoder Algorithm
- Simulation Results
- FPGA Implementation
- FPGA Results

1. L. G. Barbero and J. S. Thompson, “A Fixed-Complexity MIMO Detector Based on the Complex Sphere Decoder,” to appear in *IEEE Workshop on Signal Processing Advances for Wireless Communications (SPAWC ’06)*, Cannes, France, July 2006.
2. L. G. Barbero and J. S. Thompson, “Performance Analysis of a Fixed-Complexity Sphere Decoder for MIMO Systems,” in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP ’06)*, vol. 4, Toulouse, France, May 2006, pp. 557–560.
3. L. G. Barbero and J. S. Thompson, “Rapid prototyping of a Fixed-Throughput Sphere Decoder for MIMO Systems,” in *Proc. IEEE International Conference on Communications (ICC ’06)*, Istanbul, Turkey, June 2006.
4. L. G. Barbero and J. S. Thompson, “FPGA Design Considerations in the Implementation of a Fixed-Throughput Sphere Decoder for MIMO Systems,” to appear in *Proc. IEEE International Conference on Field Programmable Logic and Applications (FPL ’06)*, Madrid, Spain, August 2006.

Fixed-Sphere Decoder Motivation

- Disadvantages of the sphere decoder:
  - $\Rightarrow$ Its **variable complexity** poses a problem in actual communication systems where data needs to be processed in a fixed number of operations.
  - $\Rightarrow$ Its **sequential search** results in a hardware implementation that is not fully pipelined, affecting the achievable throughput.

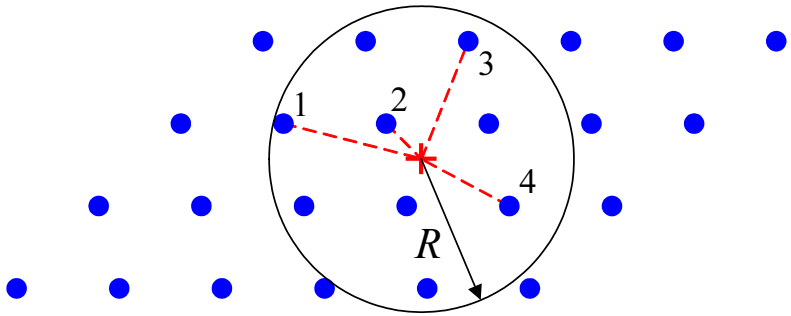
Fixed-Sphere Decoder Motivation

- Disadvantages of the sphere decoder:
  - $\Rightarrow$ Its **variable complexity** poses a problem in actual communication systems where data needs to be processed in a fixed number of operations.
  - $\Rightarrow$ Its **sequential search** results in a hardware implementation that is not fully pipelined, affecting the achievable throughput.
- Modifications of the sphere decoder to marginally reduce the average complexity require additional operations or calculation of limiting thresholds.
  - $\Rightarrow$ It results in a more complex hardware implementation without achieving a fixed complexity.
- The K-best lattice decoder (based on the sequential M-algorithm) provides a fixed complexity.
  - $\Rightarrow$ It does not take into consideration the MIMO system model resulting in a considerably high complexity.

Fixed-Sphere Decoder

- Approximate maximum-likelihood performance with fixed complexity.
- Search, independently of the noise level, over only a fixed number of lattice points $\mathbf{Hs}$, generated by a subset $\mathcal{S} \subset \mathcal{C}$, around the received point $\mathbf{r}$.

$$\hat{\mathbf{s}}_{\mathbf{fsd}} = \arg\{\min_{\mathbf{s} \in \mathcal{S}} \|\mathbf{r} - \mathbf{Hs}\|^2\}$$

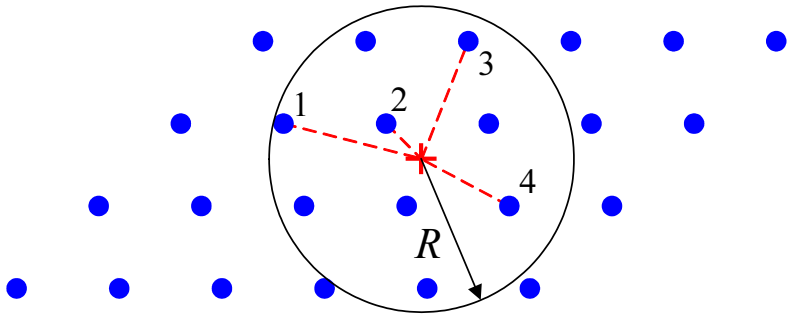


Fixed-Sphere Decoder

- Approximate maximum-likelihood performance with fixed complexity.
- Search, independently of the noise level, over only a fixed number of lattice points $\mathbf{Hs}$, generated by a subset $\mathcal{S} \subset \mathcal{C}$, around the received point $\mathbf{r}$.

$$\hat{\mathbf{s}}_{\mathbf{fsd}} = \arg\{\min_{\mathbf{s} \in \mathcal{S}} \|\mathbf{r} - \mathbf{Hs}\|^2\}$$

- In order not to have the complexity of the maximum-likelihood detector, $\mathcal{S}$ needs to be a very small subset of $\mathcal{C}$.
- How do we select the M-dimensional points to be checked?



Fixed-Sphere Decoder Algorithm (1)

- From the sphere decoder, the points to be considered per level ($i = M, \ldots, 1$) are determined by

$$|s_i - z_i|^2 \leq \frac{T_i}{u_{ii}^2}$$

where

$$T_i = R^2 - \sum_{j=i+1}^{M} u_{jj}^2 |s_j - z_j|^2$$

Fixed-Sphere Decoder Algorithm (1)

- From the sphere decoder, the points to be considered per level ($i = M, \ldots, 1$) are determined by

$$|s_i - z_i|^2 \leq \frac{T_i}{u_{ii}^2}$$

where

$$T_i = R^2 - \sum_{j=i+1}^{M} u_{jj}^2 |s_j - z_j|^2$$

- From the definition of T_i, we have

$$T_M \geq T_{M-1} \geq \cdots \geq T_1$$

- The diagonal elements of $\mathbf{U}$, u_{ii}, are such that $2u_{ii}^2$ are real-valued and have a Chi-square (χ^2) distribution with $2(N - i + 1)$ degrees of freedom and $\mathsf{E}[u_{ii}^2] = N - i + 1$, with $i = 1, \ldots, M$.

$$\mathsf{E}[u_{MM}^2] < \mathsf{E}[u_{M-1M-1}^2] < \cdots < \mathsf{E}[u_{11}^2]$$

Fixed-Sphere Decoder Algorithm (2)

- Therefore, we can write

$$\mathsf{E}[\frac{T_M}{u_{MM}^2}] > \mathsf{E}[\frac{T_{M-1}}{u_{M-1M-1}^2}] > \cdots > \mathsf{E}[\frac{T_1}{u_{11}^2}]$$

- If we define n_i as the number of points (i.e. candidates) s_i at level i that satisfy $|s_i - z_i|^2 \leq \frac{T_i}{u_{ii}^2}$, from the previous equation, we obtain

$$\mathsf{E}[n_M] \geq \mathsf{E}[n_{M-1}] \geq \cdots \geq \mathsf{E}[n_1]$$

Fixed-Sphere Decoder Algorithm (2)

- Therefore, we can write

$$\mathsf{E}[\frac{T_M}{u_{MM}^2}] > \mathsf{E}[\frac{T_{M-1}}{u_{M-1M-1}^2}] > \cdots > \mathsf{E}[\frac{T_1}{u_{11}^2}]$$

- If we define n_i as the number of points (i.e. candidates) s_i at level i that satisfy $|s_i - z_i|^2 \leq \frac{T_i}{u_{ii}^2}$, from the previous equation, we obtain

$$\mathsf{E}[n_M] \geq \mathsf{E}[n_{M-1}] \geq \cdots \geq \mathsf{E}[n_1]$$

$\Rightarrow$ The FSD assigns a fixed number of candidates, n_i with $1 \leq n_i \leq P$, to be searched per level.

  - More candidates in the first levels due to interference from the other levels.
  - The decision-feedback equalization performed on z_i and the increase in $\mathsf{E}[u_{ii}^2]$ reduce the number of candidates in the last levels.

$\Rightarrow$ The total number of M-dimensional points checked is $N_{\mathcal{S}} = \prod_{i=1}^{M} n_i$.

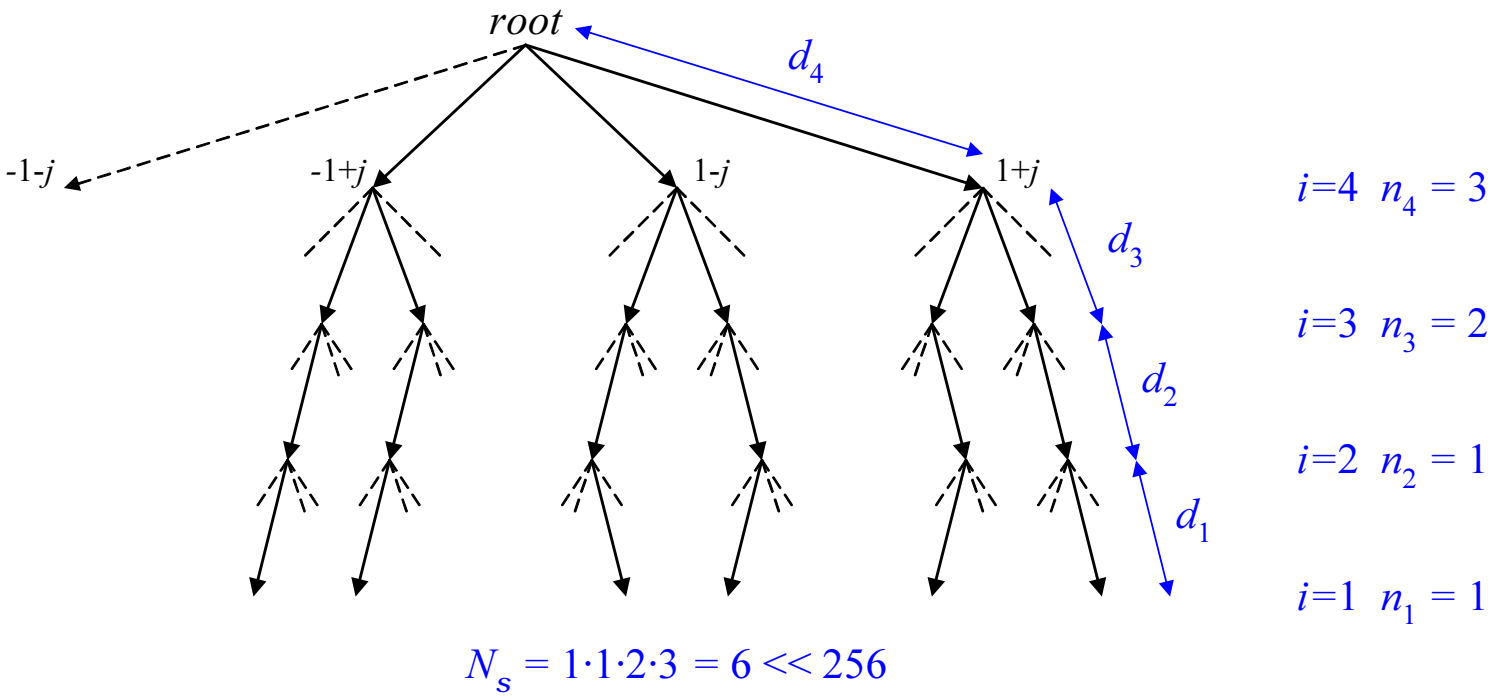
  - Objective: approximate ML performance with $N_{\mathcal{S}} \ll P^M$.

Parallelism with a Tree Search

- Conceptually, the FSD is equivalent to performing a tree search following predefined paths and selecting the path with the smallest metric as the solution.
- The n_i candidates on each level i are selected according to increasing distance to z_i, following the Schnorr-Euchner enumeration.

Parallelism with a Tree Search

- Conceptually, the FSD is equivalent to performing a tree search following predefined paths and selecting the path with the smallest metric as the solution.
- The n_i candidates on each level i are selected according to increasing distance to z_i, following the Schnorr-Euchner enumeration.
- *Hypothetical example*: 4×4 system with 4-QAM modulation ($P^M = 256$) where the number of points per level $\mathbf{n}_{\mathcal{S}} = (n_1, n_2, n_3, n_4)^T = (1, 1, 2, 3)^T$.



FSD Ordering of the Channel Matrix

- Determine the detection ordering of $\hat{s}_i$ according to the distribution of candidates, $\mathbf{n}_{\mathcal{S}}$.

$$\mathsf{E}[n_M] \geq \mathsf{E}[n_{M-1}] \geq \cdots \geq \mathsf{E}[n_1]$$

where at any level

$$\max\{n_i\} = P$$

FSD Ordering of the Channel Matrix

- Determine the detection ordering of $\hat{s}_i$ according to the distribution of candidates, $\mathbf{n}_{\mathcal{S}}$.

$$\mathsf{E}[n_M] \geq \mathsf{E}[n_{M-1}] \geq \cdots \geq \mathsf{E}[n_1]$$

  where at any level

$$\max\{n_i\} = P$$

FSD ordering

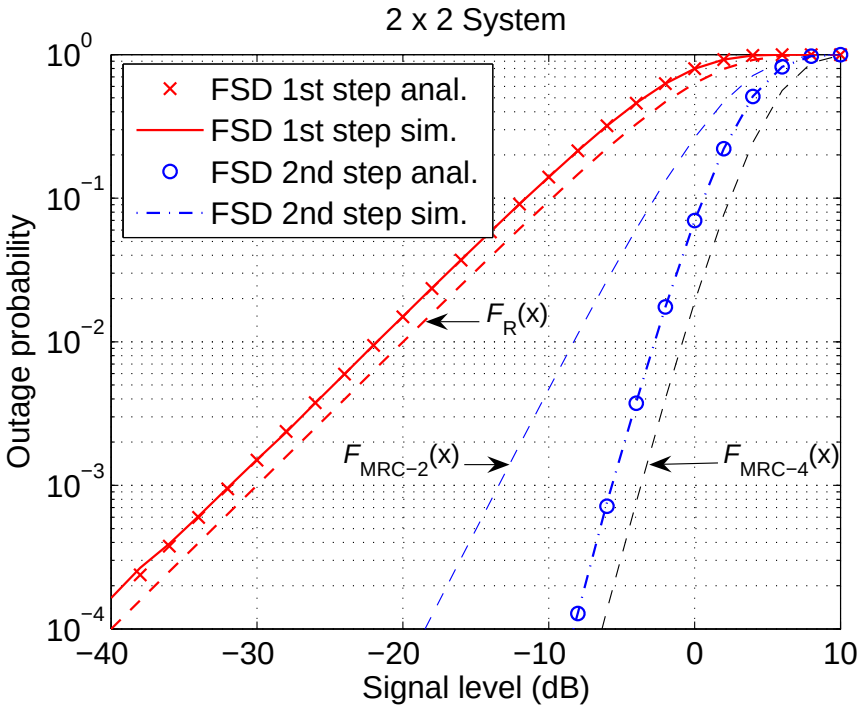
- The columns of the channel matrix are ordered according to the norm of the rows of the pseudoinverse (post-detection noise amplification), in an iterative fashion, as follows;
  - If $n_i = P$, the signal $\hat{s}_k$ with the largest noise amplification is selected.
  - If $n_i < P$, the signal $\hat{s}_k$ with the smallest noise amplification is selected.
- The SINR could also be used as a metric $\rightarrow$ it would require an estimate of the noise level at the receiver.

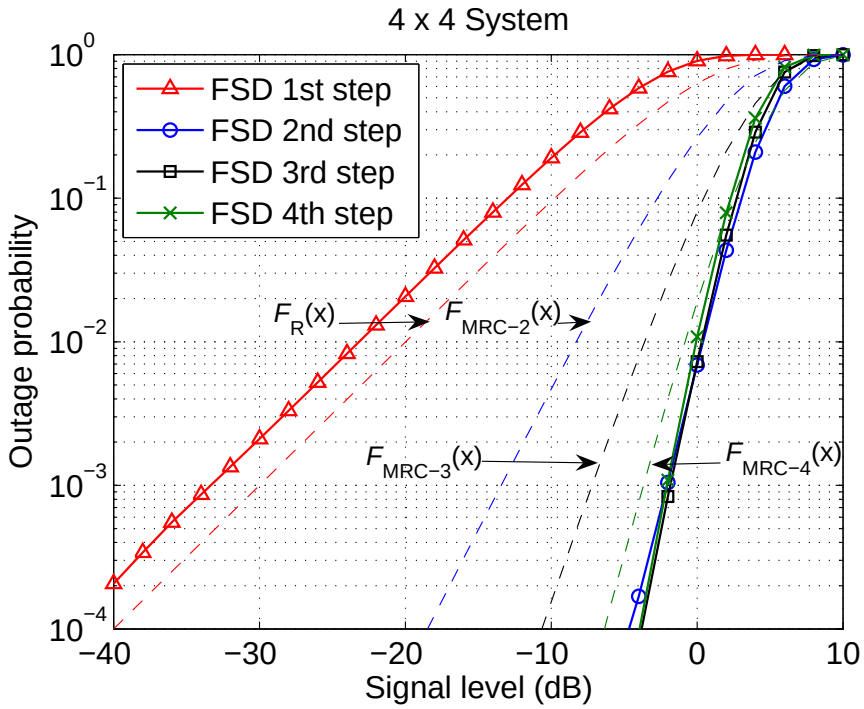
FSD Distribution of Points

- The FSD ordering has a **determinant** effect in the distribution of points $\mathbf{n}_{\mathcal{S}}$.
- Crucial to understand the effect the FSD ordering has on the channel matrix.
- No analytical study of the FSD ordering seems to be feasible when $M > 2$.

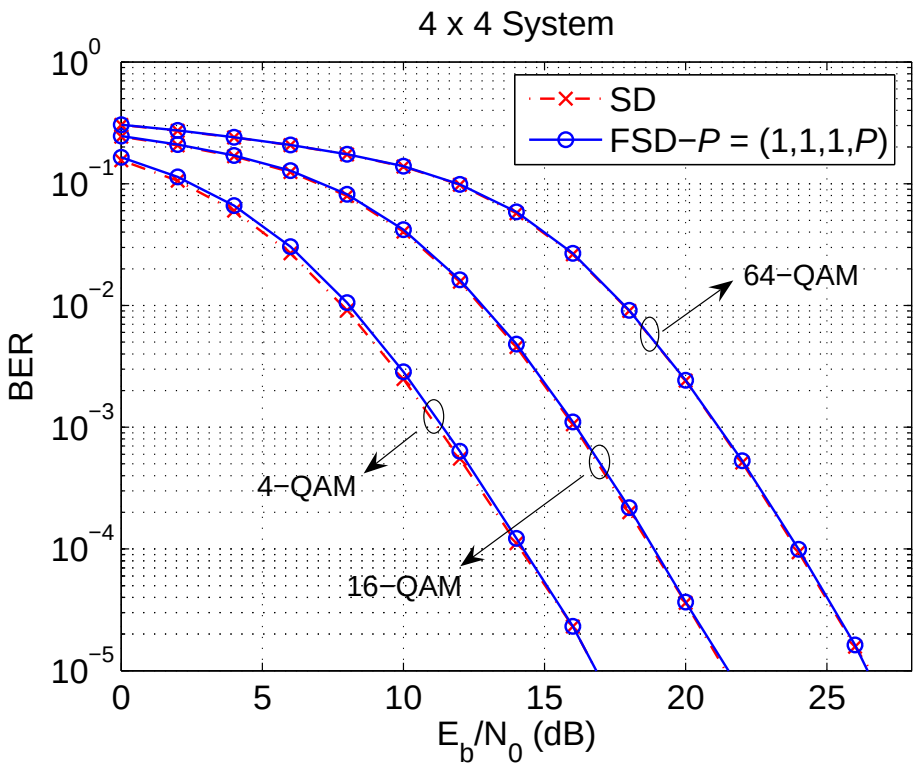
FSD Distribution of Points

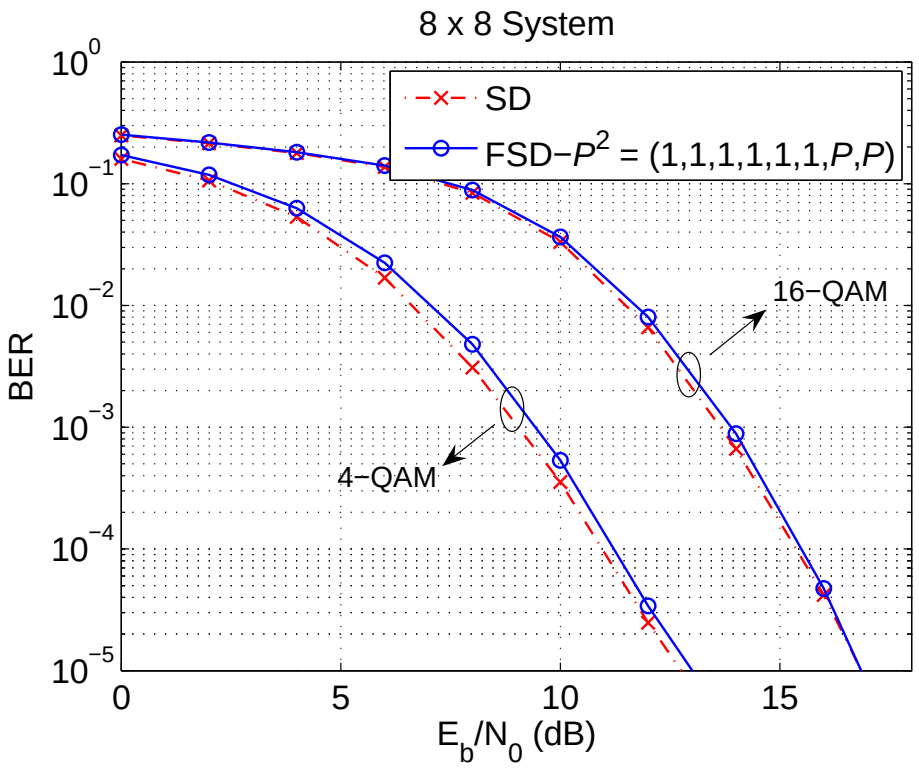
- The FSD ordering has a **determinant** effect in the distribution of points $\mathbf{n}_{\mathcal{S}}$.
- Crucial to understand the effect the FSD ordering has on the channel matrix.
- No analytical study of the FSD ordering seems to be feasible when $M > 2$.
- Effect of the FSD ordering on the outage probability of the signals $\hat{s}_i$





Fixed-Sphere Decoder Performance





$$\mathbf{n}_{\mathcal{S}} = (1,1,1,P)^T$$

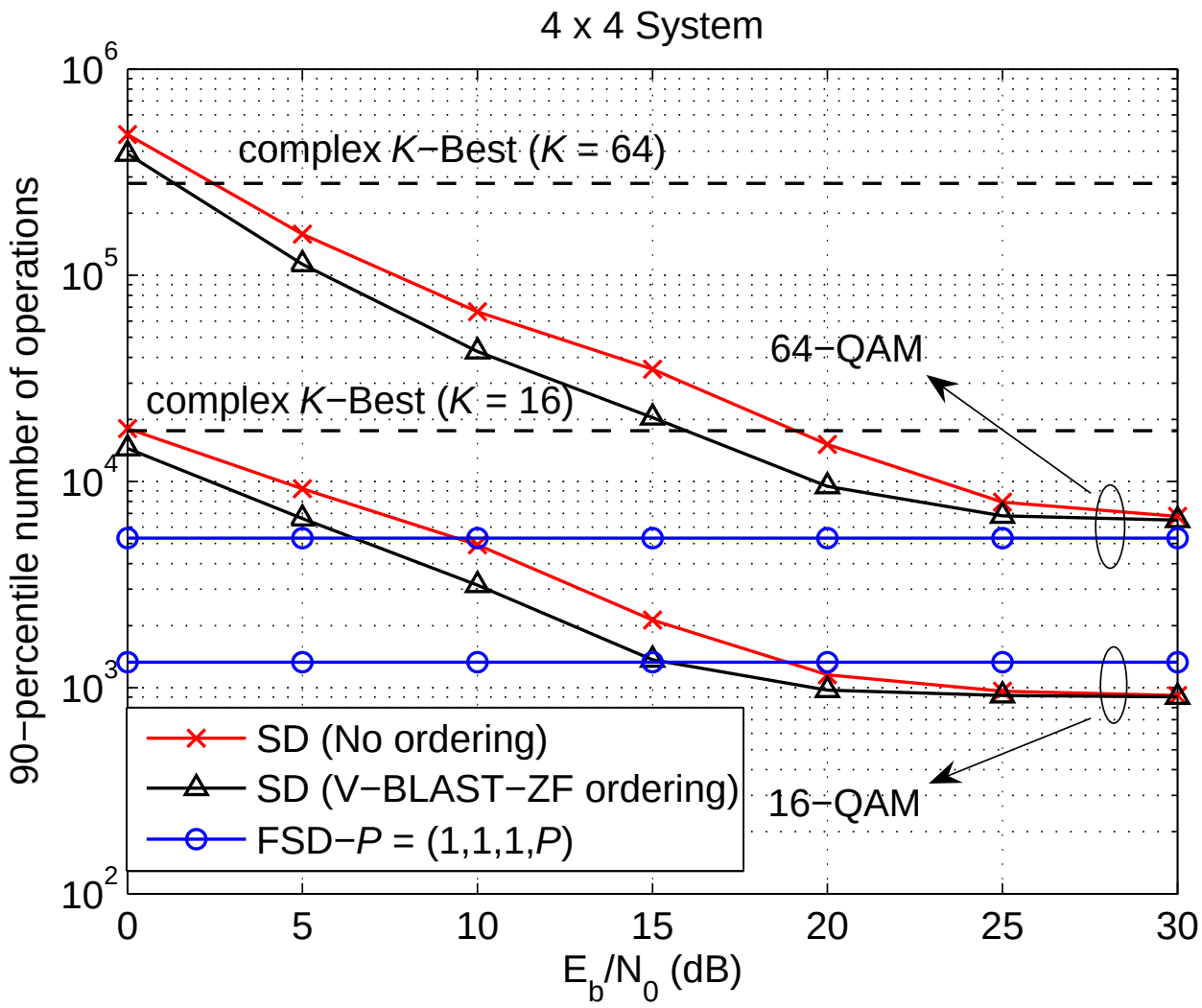
$$N_{\mathcal{S}} = \prod_{i=1}^{4} n_i = P \ll P^4$$

$$\mathbf{n}_{\mathcal{S}} = (1,1,1,1,1,1,P,P)^T$$

$$N_{\mathcal{S}} = \prod_{i=1}^{8} n_i = P^2 \ll P^8$$

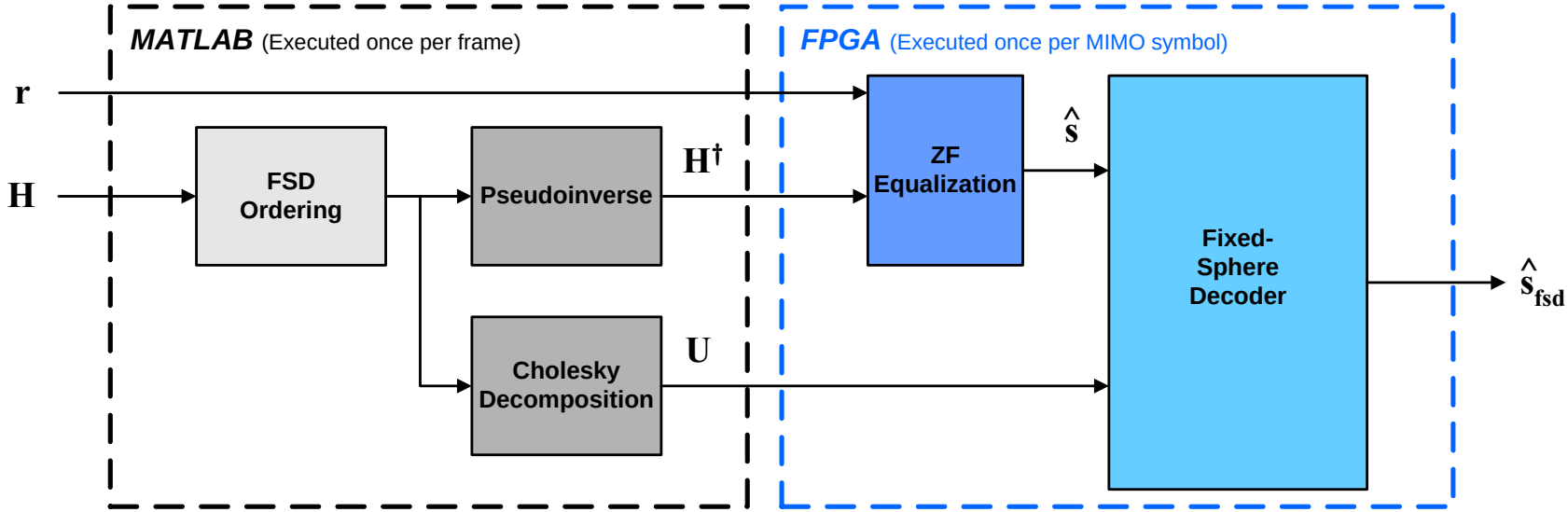
Fixed-Sphere Decoder Complexity

- 90-percentile to indicate the number of operations required in 90% of the cases.



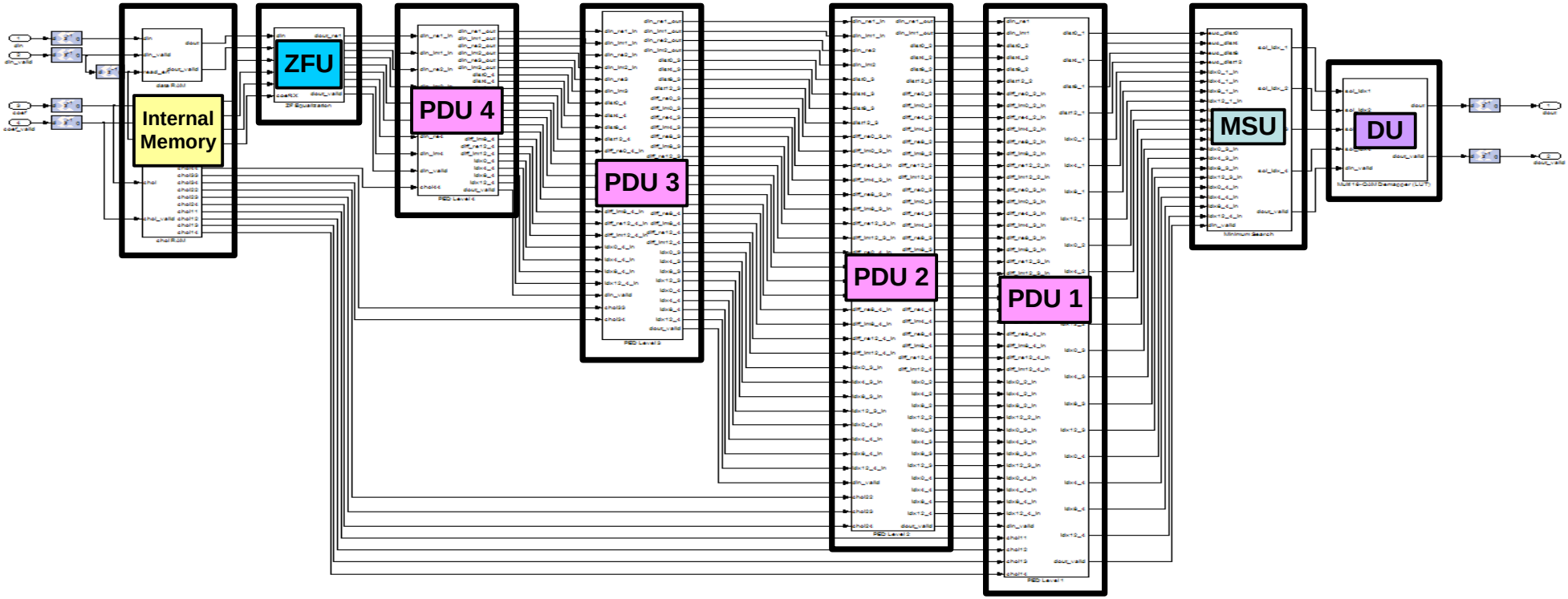
FPGA Implementation (FSD)

- FPGA performs the tree search of the fixed-sphere decoder.
- MATLAB performs parts of the algorithm that are required once per frame:
  - FSD ordering of the channel matrix
  - Pseudoinverse
  - Cholesky decomposition
- MATLAB and FPGA are synchronized through a Simulink memory interface.



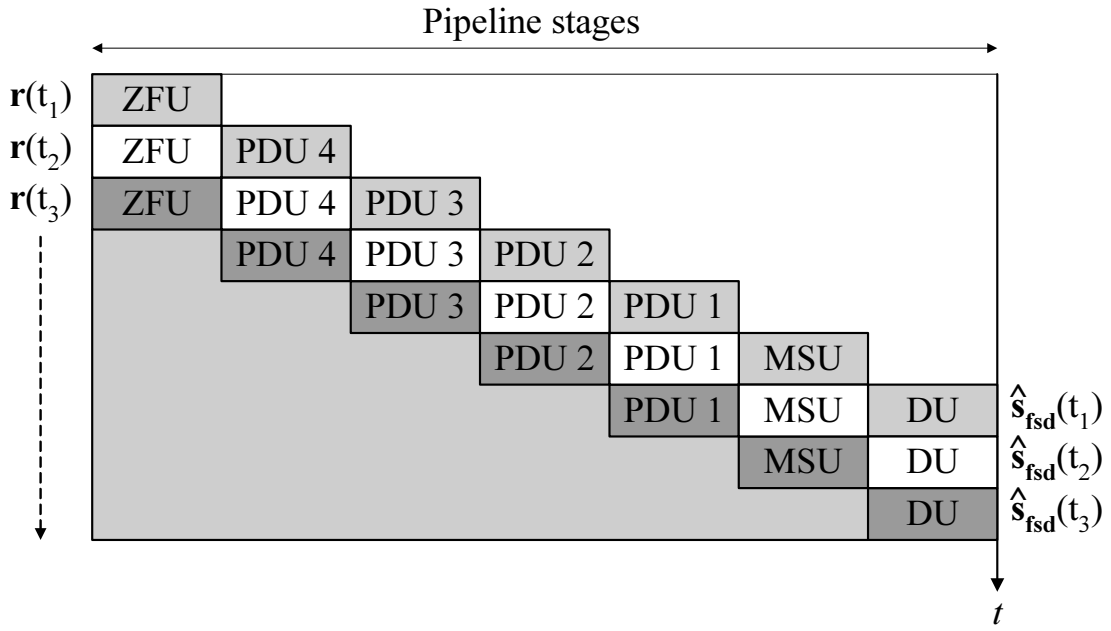
FPGA Architecture (FSD)

- 4×4 system with 16-QAM modulation



- ZFU: Zero Forcing Unit
- PDU i: Partial Distance Unit for level i
- MSU: Minimum Search Unit
- DU: Demapper Unit

FPGA Scheduling (FSD)



- The fixed structure of the FSD can be fully pipelined.
  1. Optimized use of the hardware resources.
  2. A MIMO symbol could be detected in every clock cycle.
  3. Throughput $\propto$ clock frequency.
  4. Pipeline registers can be introduced between blocks:

     increase in clock frequency $\rightarrow$ higher throughput.

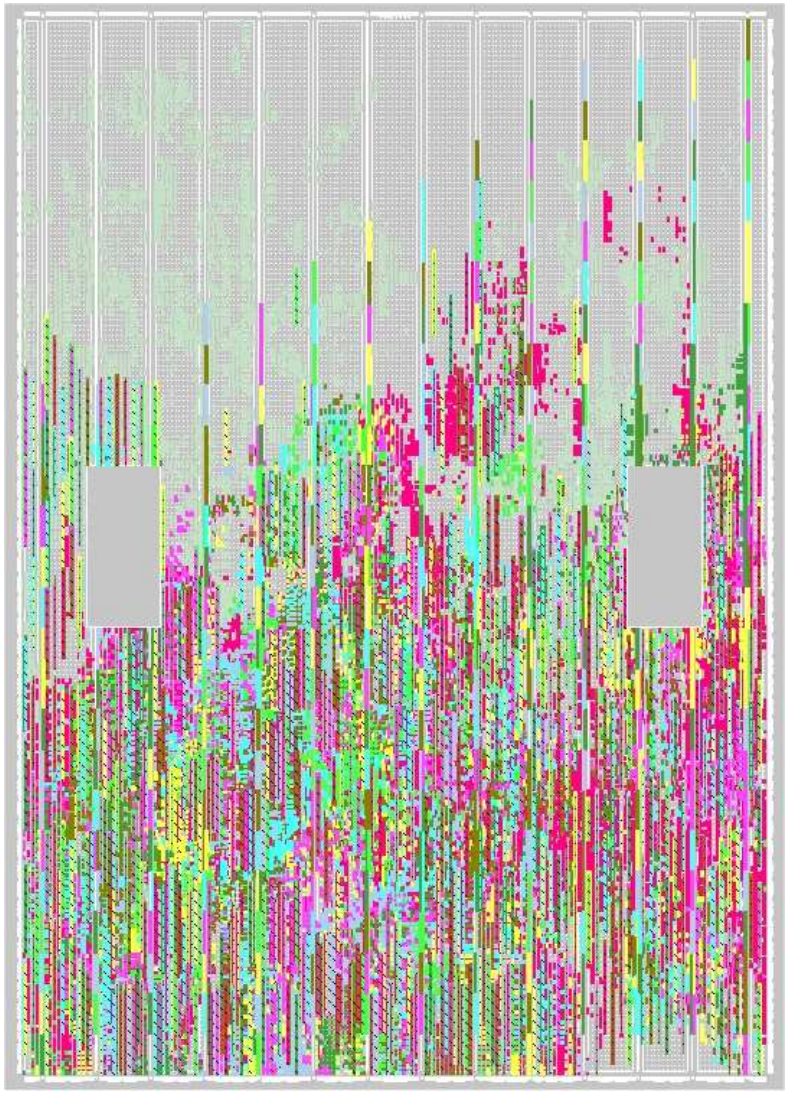
FPGA Results (FSD)

Xilinx XC2VP70 FPGA	4-SDs	**FSD**	**Diff.**
Number of slices (33,088)	64% (21,467)	**38% (12,721)**	**-40.7%**
Number of flip-flops (66,176)	26% (17,691)	**23% (15,332)**	**-13.3%**
Number of 4-input LUT (66,176)	54% (36,249)	**24% (16,119)**	**-55.5%**
Number of multipliers (328)	47% (156)	**48% (160)**	**+2.6%**
Number of block RAM (328)	55% (183)	**25% (82)**	**-55.2%**

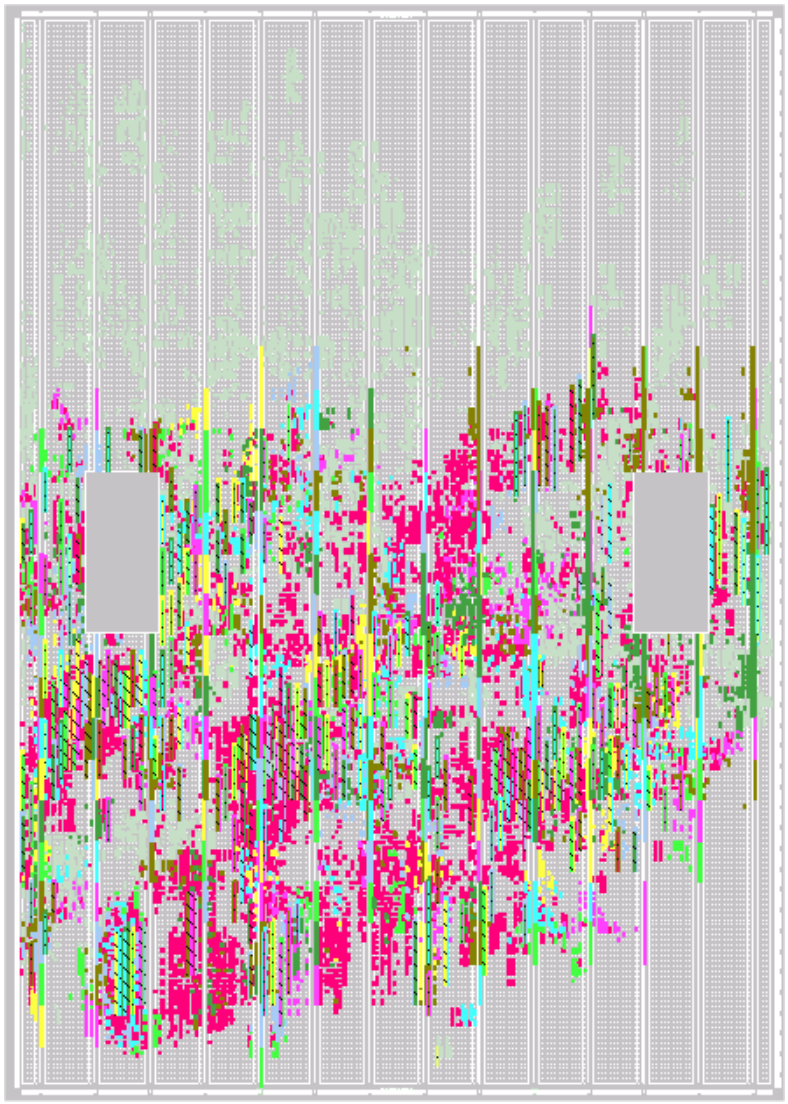
- Similar computational complexity (although better use of the multipliers).
- Flip-flops required to synchronize the different pipeline stages.
- LUTs considerably reduced given that the FSD requires less control logic.
- The FSD requires much less memory space for intermediate data storage.

Floorplan Comparison

4-SDs

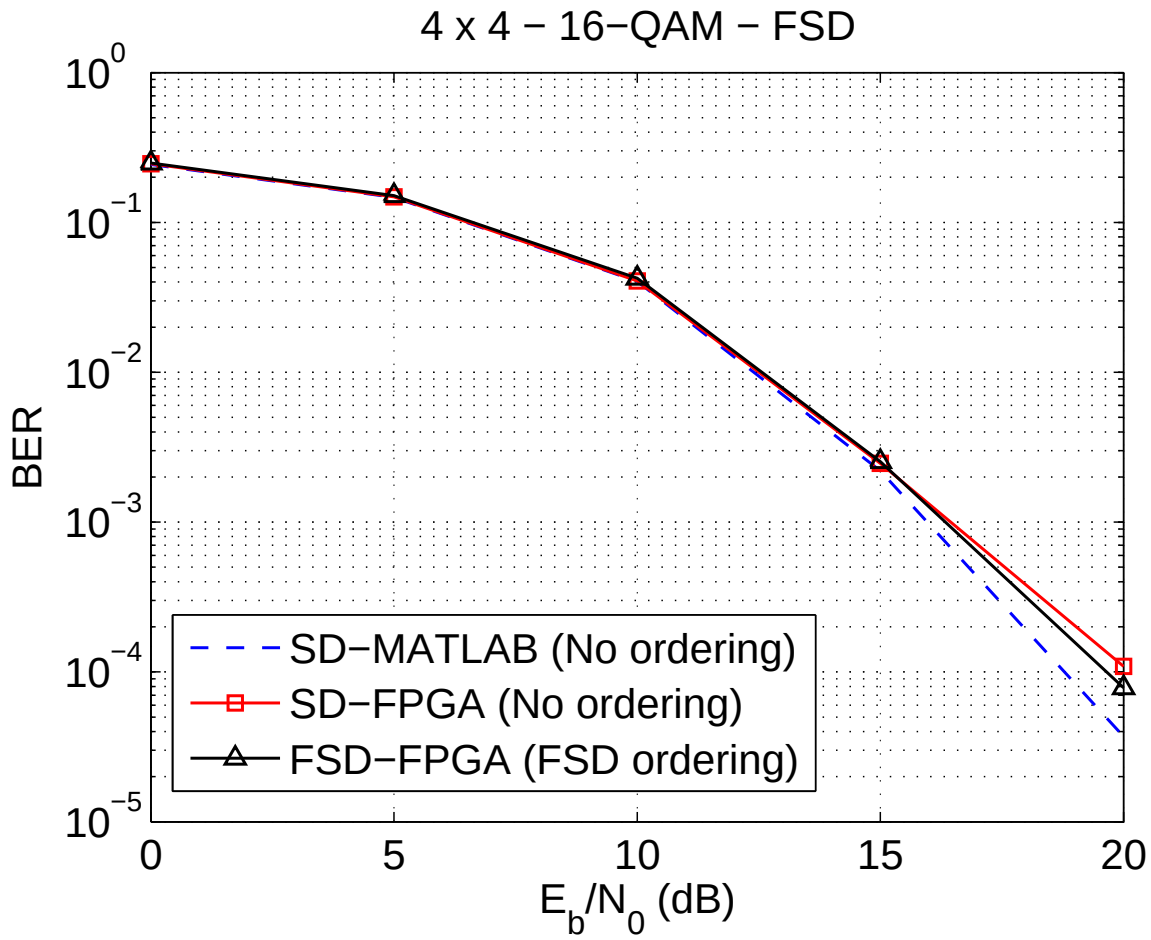


FSD



FPGA BER Performance (FSD)

- Monte Carlo simulations with real-time hardware co-simulation.
- 16 bits used for real and imaginary components quantization.

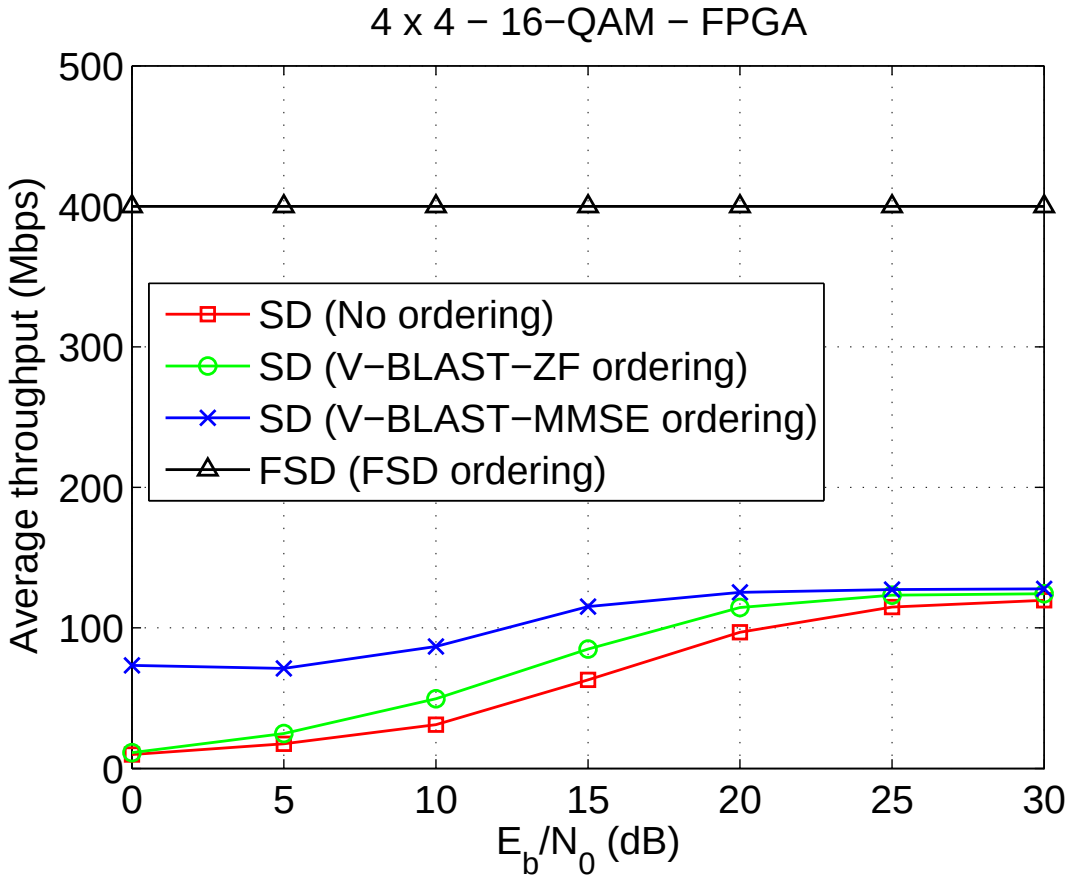


FPGA Throughput Performance (FSD)

- Throughput depends on the clock frequency and the number of cycles per detection.

$$Q = M \cdot \log_2 P \cdot f_{clock} \,/\, C \quad \text{(Mbps)}$$

- $f_{clock} = 100\,\text{MHz}$, $C = 4\,\text{cycles} \rightarrow Q = 400\,\text{Mbps}$



FSD / SD Comparison

	K-Best	SD 1	SD 2	**SD**	**FSD**
Hardware platform	ASIC	ASIC	ASIC	**FPGA**	**FPGA**
MIMO system	4×4	4×4	4×4	**4×4**	**4×4**
Modulation	16-QAM	16-QAM	16-QAM	**16-QAM**	**16-QAM**
Floating-point BER perfomance	quasi-ML	ML	close to ML	**ML**	**quasi-ML**
Clock frequency	100 MHz	51 MHz	71 MHz	**50 MHz**	**100 (150) MHz**
Throughput at E_b/N_0 = 20dB	53.3 Mbps (constant)	126 Mbps	253 Mbps	**114.5 Mbps**	**400 (600) Mbps (constant)**

Conclusion

- Sphere decoder is a promising algorithm for MIMO detection that can be implemented in real-time.
- However, the nature of the sphere decoder makes it difficult to achieve a high-throughput, highly-pipelined design.
- Fixed-sphere decoder that overcomes the two problems of the sphere decoder: its sequential nature and its variable complexity.
- FPGA rapid prototyping allows us to quickly analyze both algorithms from an implementation point of view.
- FPGA implementation of the FSD:
  - It uses less resources and achieves higher (constant) throughput that an equivalent SD on the same FPGA.
  - It has a considerably better performance than existing ASIC implementations of the SD.

Present Work

- Analytical study of the FSD ordering to better understand the effect the ordering has on the outage probability of the post-detection signals.
- Extension of the FSD to obtain soft-information for iterative detection and decoding in turbo-MIMO systems
- FPGA implementation of a so-called list-FSD.
- Collaboration with the University of Mondragon (Spain) to initially integrate the original sphere decoder in a complete real-time MIMO transceiver (invited paper to special session in MIMO testbeds in EUSIPCO 06).

Thank you!

References

[1] G. J. Foschini, “Layered space-time architecture for wireless communication in a fading environment when using multi-element antennas,” *Bell Labs Technical Journal*, pp. 41–59, Oct. 1996.

[2] T. Kaiser, A. Wilzeck, M. Berentsen, and M. Rupp, “Prototyping for MIMO systems: An overview,” in *Proc. 12th European Signal Processing Conference (EUSIPCO ’04)*, Vienna, Austria, Sept. 2004.

[3] A. Adjoudani, E. C. Beck, A. P. Burg, G. M. Djuknic, T. G. Gvoth, D. Haessig, S. Manji, M. A. Milbrodt, M. Rupp, D. Samardzija, A. B. Siegel, T. Sizer, C. Tran, S. Walker, S. A. Wilkus, and P. W. Wolniansky, “Prototype experience for MIMO BLAST over third-generation wireless systems,” *IEEE J. Select. Areas Commun.*, vol. 21, no. 3, pp. 440–451, Apr. 2003.

[4] P. Murphy, F. Lou, A. Sabharwal, and J. P. Frantz, “An FPGA based rapid prototyping platform for MIMO systems,” in *Proc. 37th Asilomar Conference on Signals, Systems and Computers*, vol. 1, Pacific Grove, CA, USA, Nov. 2003, pp. 900–904.

[5] A. van Zelst and T. Schenk, “Implementation of a MIMO OFDM based wireless LAN system,” *IEEE Trans. Signal Processing*, vol. 52, no. 2, pp. 483–494, Sept. 2004.

[6] E. Viterbo and J. Boutros, “A universal lattice code decoder for fading channels,” *IEEE Trans. Inform. Theory*, vol. 45, no. 5, pp. 1639–1642, July 1999.

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

[8] M. O. Damen, H. E. Gamal, and G. Caire, “On maximum-likelihood detection and the search for the closest lattice point,” *IEEE Trans. Inform. Theory*, vol. 49, no. 10, pp. 2389–2402, Oct. 2003.

[9] P. W. Wolniansky, G. J. Foschini, G. D. Golden, and R. A. Valenzuela, “V-BLAST: An architecture for realizing very high data rates over the rich-scattering wireless channel,” in *Proc. URSI International Symposium on Signals, Systems and Electronics (ISSSE ’98)*, Atlanta, GA, USA, Sept. 1998, pp. 295–300.

[10] K. wai Wong, C. ying Tsiu, R. S. kwan Cheng, and W. ho Mow, “A VLSI architecture of a K-best lattice decoding algorithm for MIMO channels,” in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS ’02)*, vol. 3, Scottsdale, AZ, USA, May 2002, pp. 273–276.

[11] Z. Guo and P. Nilsson, “A VLSI architecture of the Schnorr-Euchner decoder for MIMO systems,” in *Proc. IEEE 6th Circuits and Systems Symposium on Emerging Technologies: Frontiers of Mobile and Wireless Communication*, vol. 1, Shanghai, China, June 2004, pp. 65–68.

[12] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bölcskei, “VLSI implementation of MIMO detection using the sphere decoding algorithm,” *IEEE J. Solid-State Circuits*, vol. 40, no. 7, pp. 1566–1577, July 2005.

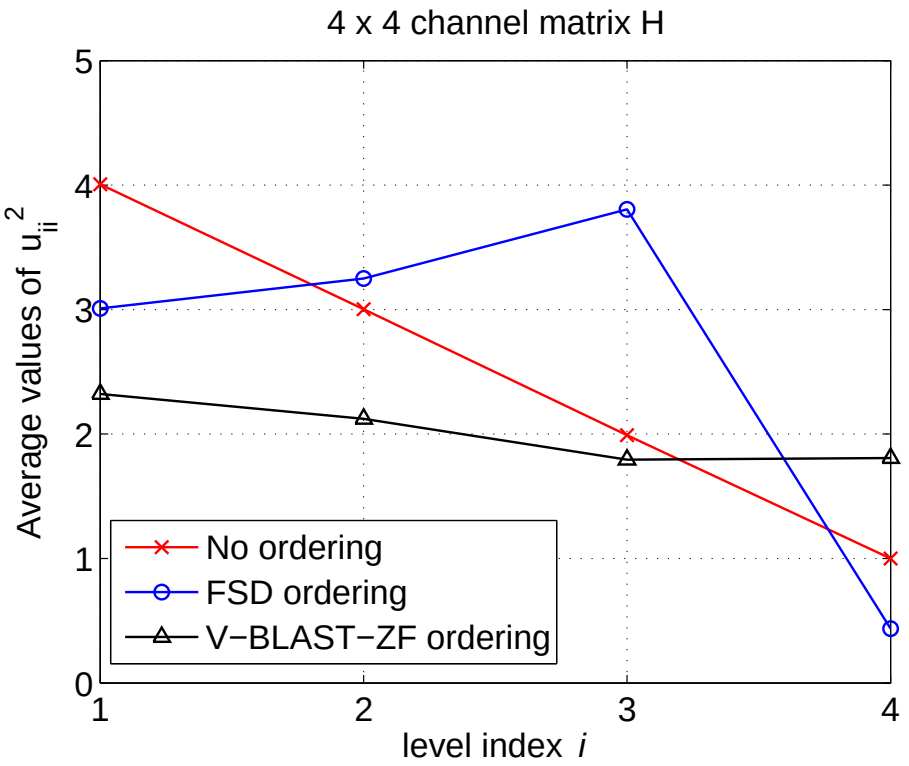
FPGA Clock Frequency (SD)

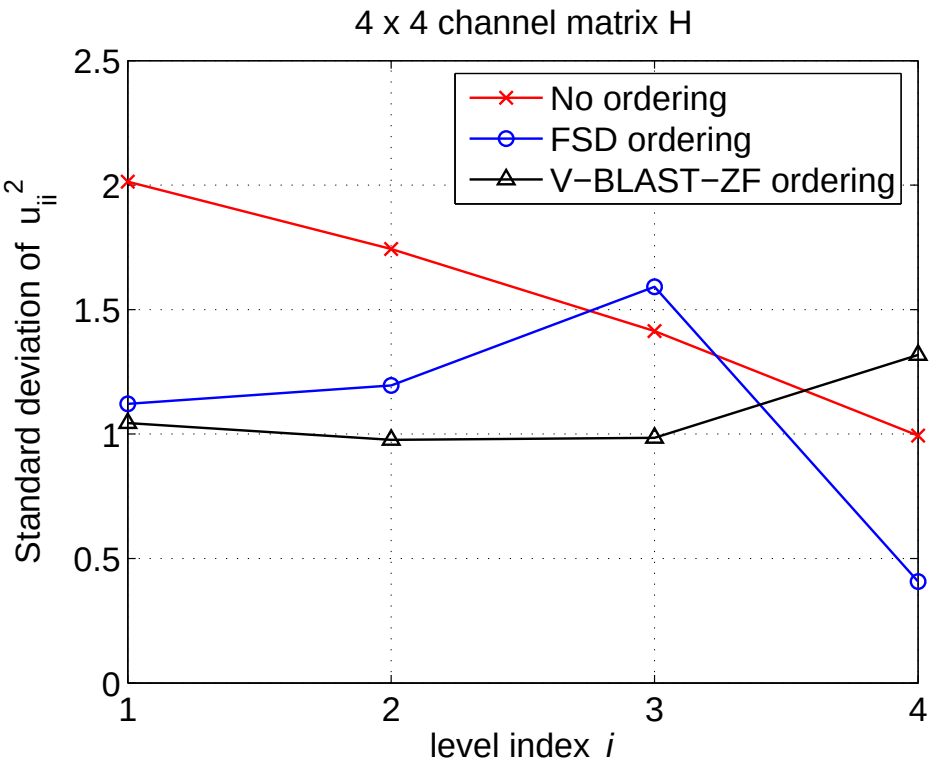
- Increasing the clock frequency f_{clock} does not result in a direct increment in the throughput.
- Due to the sequential nature of the sphere decoder, the number of cycles C also increases.

f_{clock} (MHz)	C_{min} (cycles)	Q_{max} (Mbps)
50	25	128
54.18	25	138.70
68.46	37	118.41
99.39	53	120.02

Effect of the FSD Ordering

- Squared-diagonal elements of the upper-triangular matrix $\mathbf{U}$ in a 4×4 system where $\mathbf{H} \sim \mathcal{CN}(\mathbf{0}, \mathbf{I}_N \otimes \mathbf{I}_M)$.





FSD / K-Best Decoder

- They both achieve quasi-ML performance with fixed complexity.
- Performance degradation compared to ML at BER = 10^{-3} in a 4×4 system when $N_{\mathcal{S}} = K = P$.

Modulation	FSD	K-Best
16-QAM	0.06 dB	0.015 dB
64-QAM	0.03 dB	0.05 dB

Reducing the Number of Multipliers

- The FSD for a 4×4 system with 16-QAM uses 160 multipliers (limiting factor).
- Reduce the number of multipliers without affecting the performance (only in fully-pipelined designs):
  - A complex multiplication requires 4 multipliers and 2 adders (latency = 2 cycles)

$$(a + jb)(c + jd) = (ac - bd) + j(bc + ad)$$

  - It can be rewritten to require 3 multipliers and 5 adders (latency = 3 cycles)

$$(a + jb)(c + jd) = [a(c - d) + d(a - b)] + j[b(c + d) + d(a - b)]$$

- A different metric can be used to further reduce the number of multipliers with a non-negligible effect on the performance (Manhattan distance).
- Applying those two methods to the FSD:

160 mult. $\rightarrow$ 132 mult. $\rightarrow$ 100 mult.

FPGA Results (FSD-16 and FSD-64)

Xilinx XC2VP70 FPGA	FSD-16	FSD-64
MIMO system	4×4	4×4
Modulation	16-QAM	64-QAM
Number of slices (33,088)	38% (12,721)	62% (20,580)
Number of flip-flops (66,176)	23% (15,332)	40% (26,372)
Number of 4-input LUT (66,176)	24% (16,119)	41% (27,643)
Number of multipliers (328)	48% (160)	92% (304)
Number of block RAM (328)	25% (82)	26% (88)
Number of cycles C	4	8
Clock frequency $f_{clock}(MHz)$	100	100
Throughput Q (Mbps)	400	300

Design Optimizations (FSD-64)

Xilinx XC2VP70 FPGA	FSD-64A	FSD-64B	FSD-64C	optimized FSD-64B
Number of slices	62%	65%	65%	74%
Number of flip-flops	40%	47%	48%	60%
Number of 4-input LUT	41%	45%	48%	47%
Number of multipliers	92%	76%	57%	76%
Number of block RAM	26%	26%	26%	26%
f_{clock} (MHz)	100	100	100	150
Q (Mbps)	300	300	300	450
Initial latency (cycles)	62	66	66	78

V-BLAST-MMSE Ordering

- The performance degradation is due to the fact that the ML solution obtained is the solution to

$$\min_{\mathbf{s}} \|\tilde{\mathbf{r}} - \tilde{\mathbf{H}}\mathbf{s}\|^2$$

where

$$\tilde{\mathbf{H}} = \begin{bmatrix} \mathbf{H} \\ \sigma\sqrt{M}\mathbf{I}_M \end{bmatrix}, \tilde{\mathbf{r}} = \begin{bmatrix} \mathbf{r} \\ \mathbf{0}_{M1} \end{bmatrix}$$

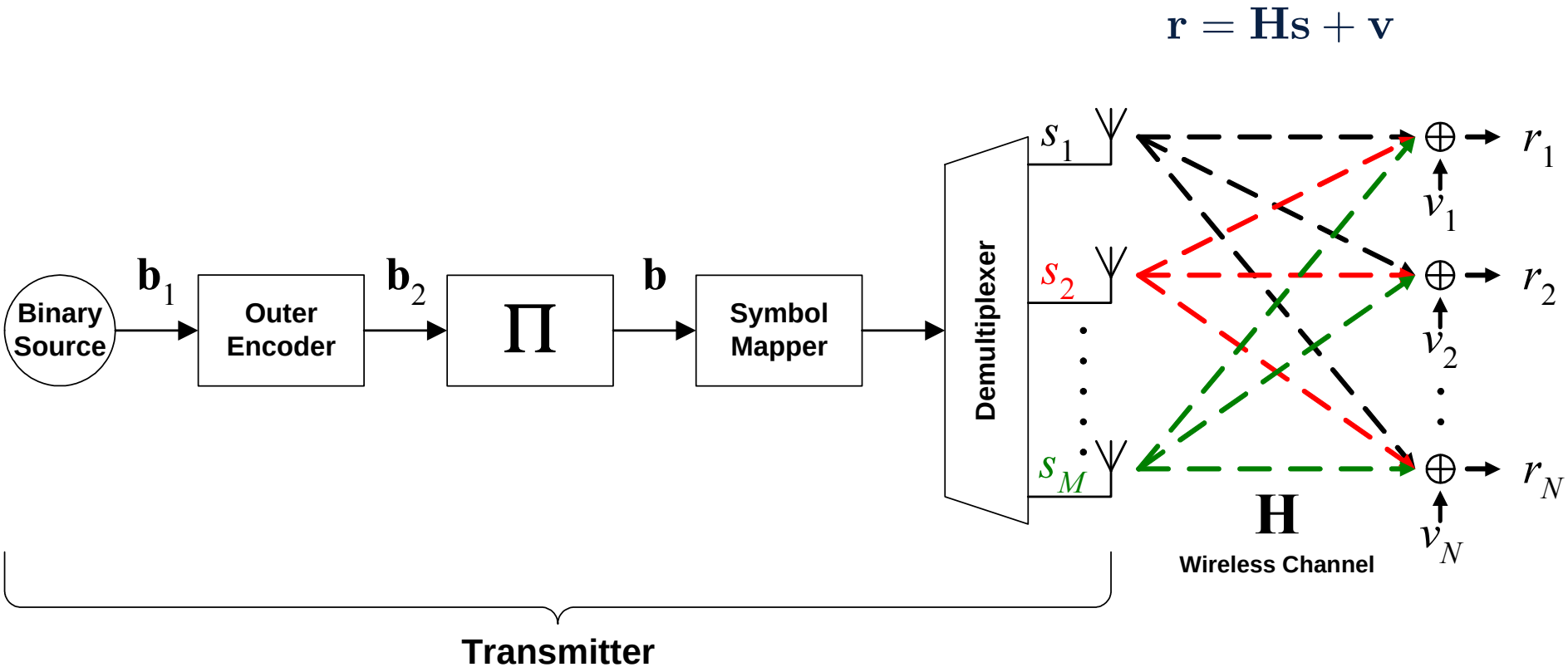
with $\mathbf{I}_M$ the $M \times M$ identity matrix and $\mathbf{0}_{M1}$ the $M \times 1$ 0-vector.

- The (SC) is written as

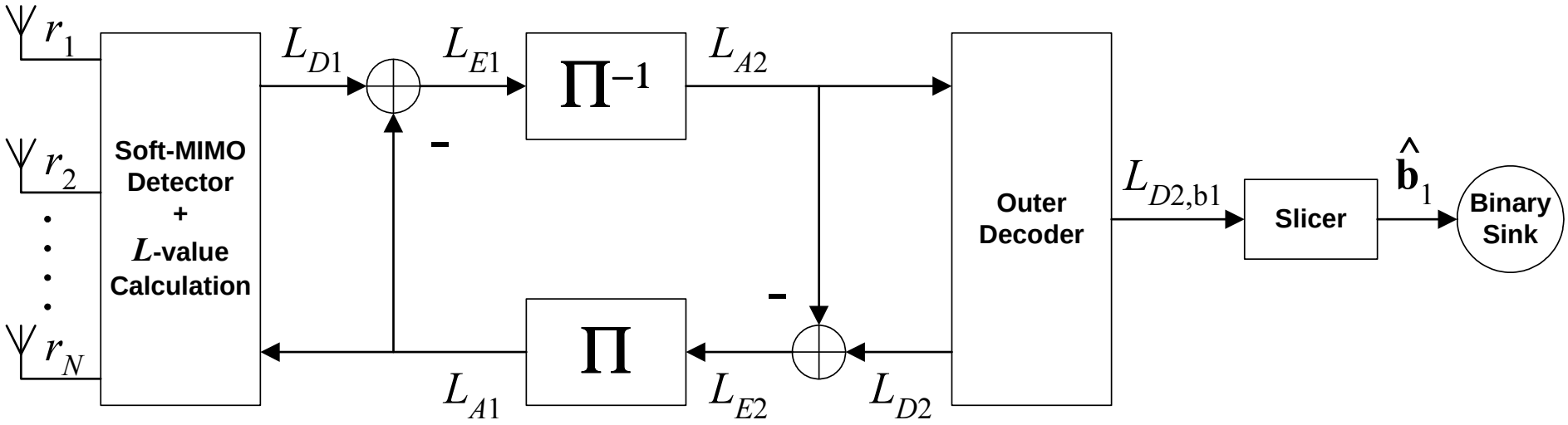
$$\hat{\mathbf{s}}_{\mathbf{ml-mmse}} = \arg\{\min_{\mathbf{s}} \|\tilde{\mathbf{U}}(\mathbf{s} - \hat{\mathbf{s}}_{\mathbf{mmse}})\|^2 \leq R^2\}$$

  - $\tilde{\mathbf{U}}$: $M \times M$ upper triangular matrix, Cholesky decomposition of Gram matrix $\tilde{\mathbf{G}} = \tilde{\mathbf{H}}^H\tilde{\mathbf{H}}$
  - $\hat{\mathbf{s}}_{\mathbf{mmse}} = \tilde{\mathbf{H}}^\dagger\mathbf{r}$
  - $\tilde{\mathbf{H}}^\dagger = (\tilde{\mathbf{H}}^H\tilde{\mathbf{H}})^{-1}\tilde{\mathbf{H}}^H$: pseudoinverse of $\tilde{\mathbf{H}}$

Transmitter Architecture



Receiver Architecture



- L-value: log-likelihood ratio (LLR)

$$\underbrace{L_{D1}(b_k|\mathbf{r})}_{a-posteriori\ \text{info}} = \ln\frac{P[b_k=+1|\mathbf{r}]}{P[b_k=-1|\mathbf{r}]} = \underbrace{L_{A1}(b_k)}_{a-priori\ \text{info}} + \underbrace{L_{E1}(b_k|\mathbf{r})}_{\text{extrinsic info}}$$

List Fixed-Sphere Decoder

- The LFSD is based on the original FSD and is denoted as LFSD-$N_{\mathcal{S}_e}/N_{\mathcal{L}}$.
  1. **Search stage**: search through lattice points generated by an *extended* subset $\mathcal{S}_e \subset \mathcal{C}$.
  2. **Sort and select stage**: it generates a list $\mathcal{L}$ of the $N_{\mathcal{L}}$ candidates with the smallest metric ($N_{\mathcal{L}} \leq N_{\mathcal{S}_e}$).

$$L_{E1}(b_k|\mathbf{r}) \approx \frac{1}{2} \max_{\mathbf{b}\in\mathcal{L}\cap\mathbb{B}_{k,+1}} \left\{ \frac{-\|\mathbf{r}-\mathbf{Hs}\|^2}{\sigma^2} + \mathbf{b}_{[k]}^T \mathbf{L}_{A1,[k]} \right\} - \frac{1}{2} \max_{\mathbf{b}\in\mathcal{L}\cap\mathbb{B}_{k,-1}} \left\{ \frac{-\|\mathbf{r}-\mathbf{Hs}\|^2}{\sigma^2} + \mathbf{b}_{[k]}^T \mathbf{L}_{A1,[k]} \right\}$$

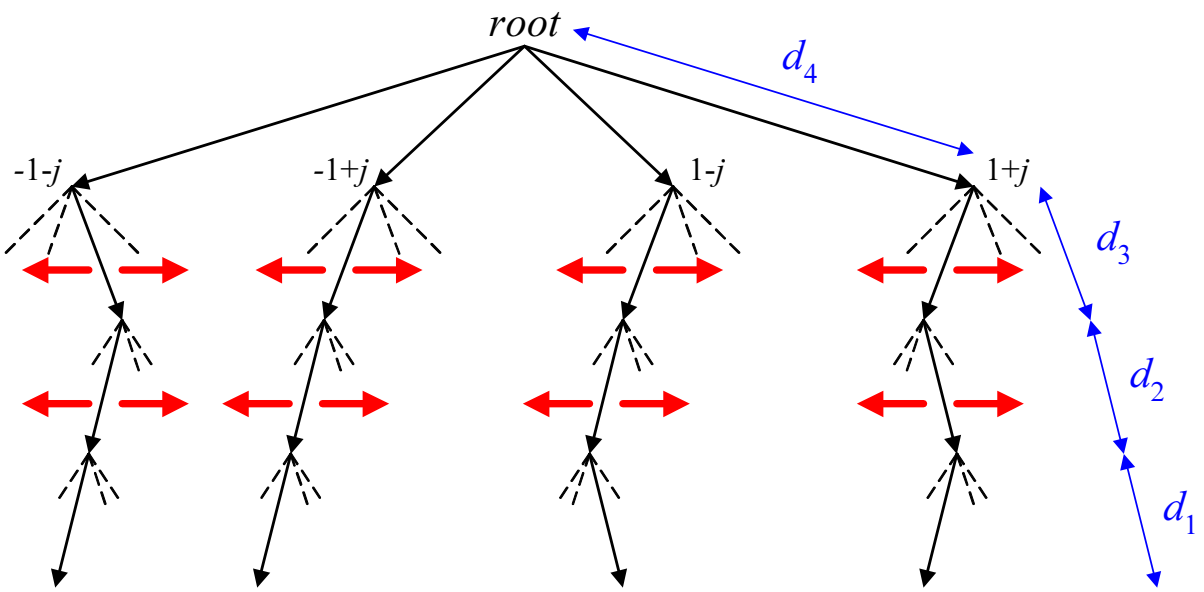
$\mathcal{L} \cap \mathbb{B}_{k,+1}$ denotes the subgroup of vectors of $\mathcal{L}$ that have $b_k = +1$

- The subset $\mathcal{S}_e$ is an extension of $\mathcal{S}$ to improve the quality of the candidates from a soft-output perspective.

LFSD Distribution of Points

- The distribution of points $\mathbf{n}_{\mathcal{S}_e}$ is obtained taking as a starting point the distribution of points $\mathbf{n}_{\mathcal{S}}$ (equivalent uncoded FSD).
- Gradually increase the number of points n_i on the levels $\{i \mid n_i < P\}$ until we reach $N_{\mathcal{S}_e}$.

$$n_i^{(k+1)} = 2 \cdot n_i^{(k)}$$



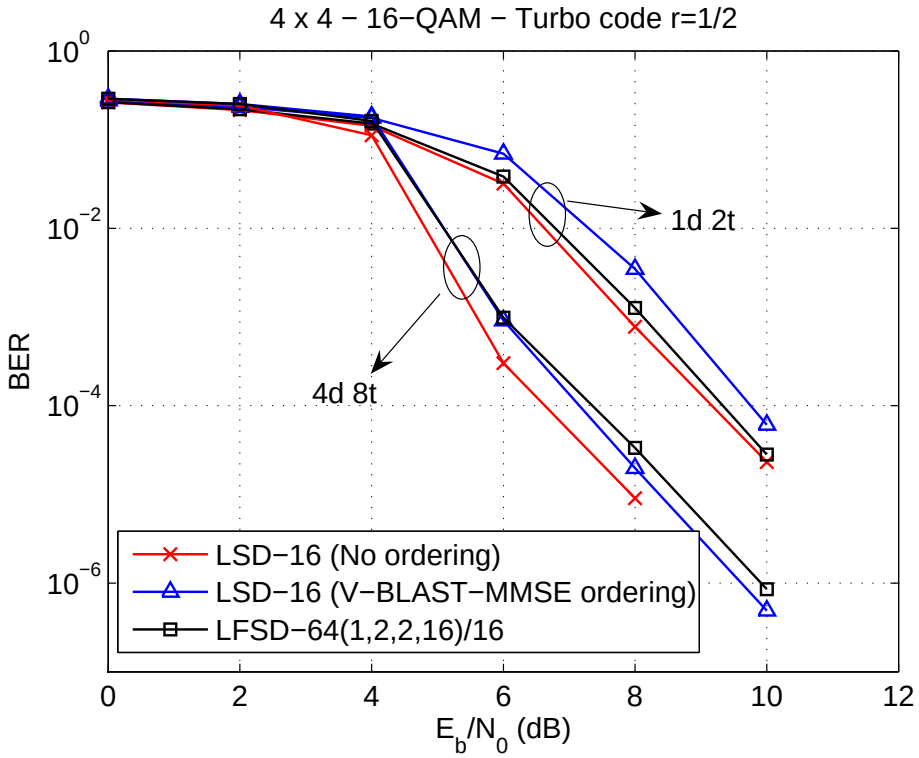
	FSD	LFSD
i=4	$n_4 = 4$	$n_4 = 4$
i=3	$n_3 = 1$	$n_3 = 2$
i=2	$n_2 = 1$	$n_2 = 2$
i=1	$n_1 = 1$	$n_1 = 1$

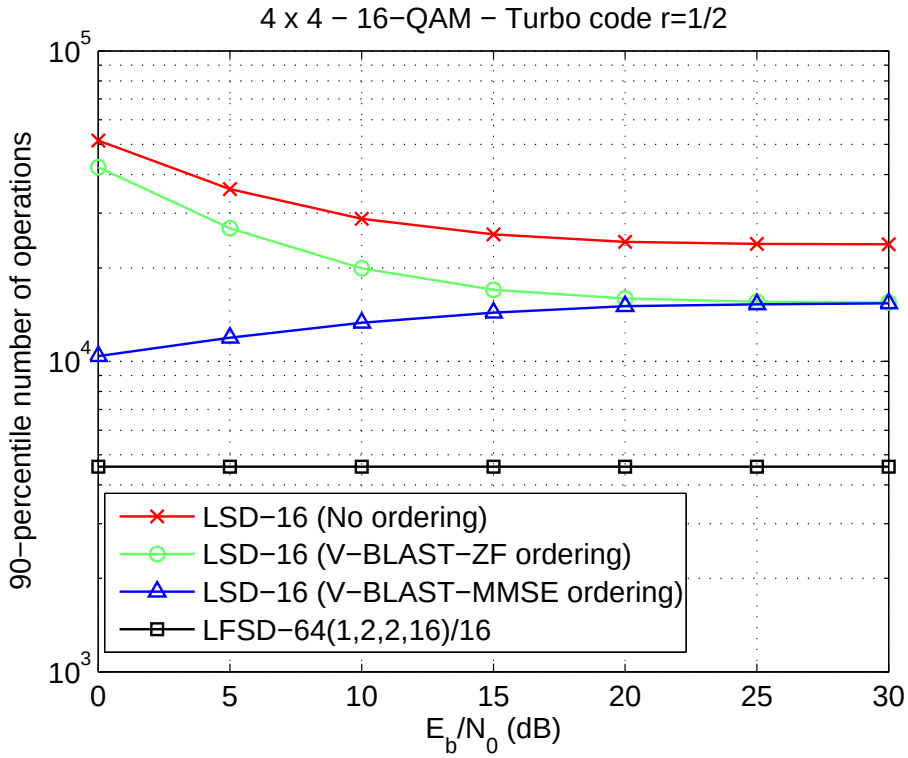
$N_s = 1 \cdot 1 \cdot 1 \cdot 4 = 4 << 256$

$N_{s_e} = 1 \cdot 2 \cdot 2 \cdot 4 = 16 << 256$

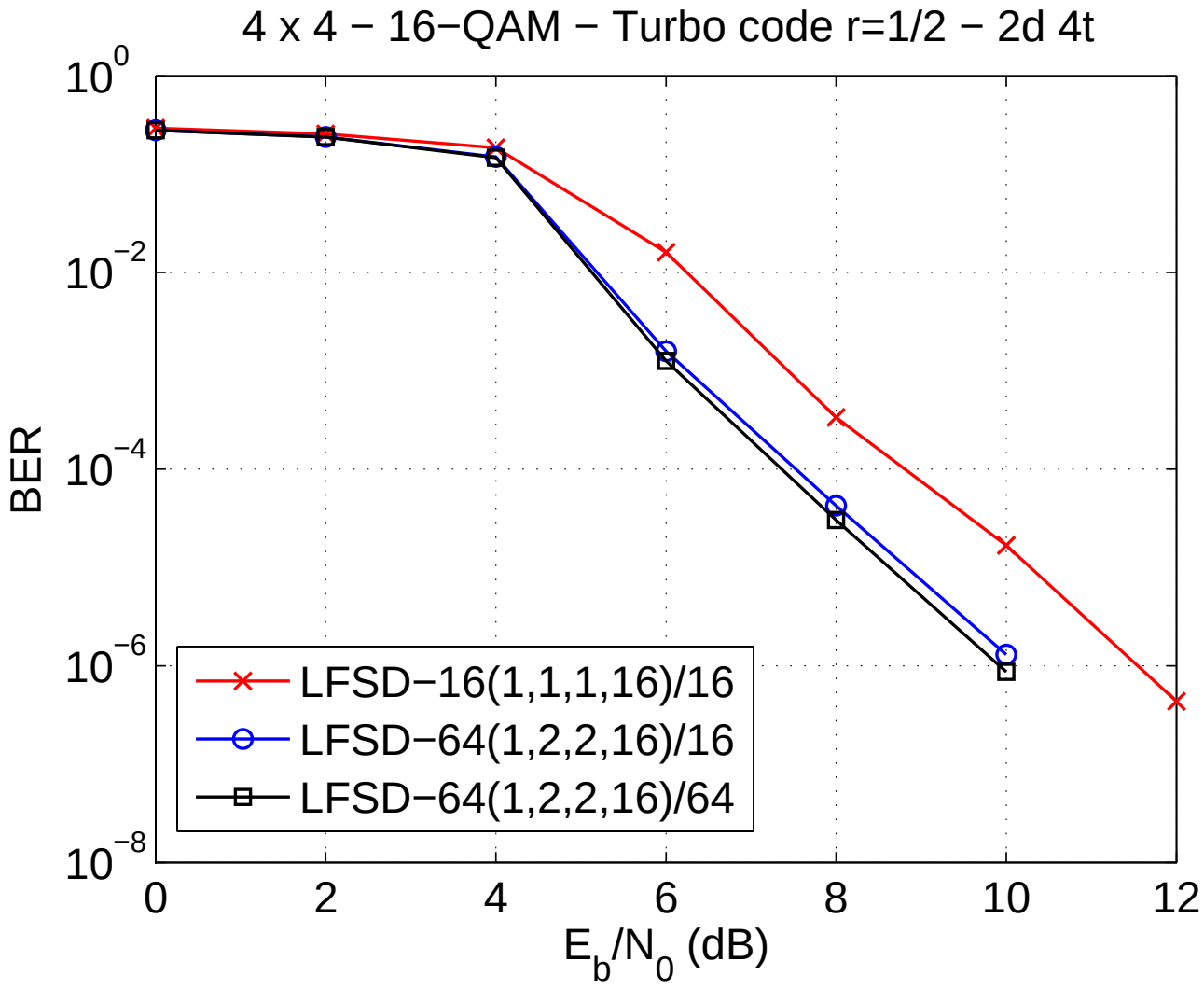
LFSD Performance

- 4×4 system with 16-QAM modulation.
- Parallel concatenated turbo code $r = 1/2$ of memory 2 ($G_1(D) = 1 + D + D^2$, $G_2(D) = 1 + D^2$).
- Frames of $K_b = 8192$ bits with $K_{ch} = 16$ symbols transmitted per antenna and channel realization.
- LFSD-64/16 with distribution of points $\mathbf{n}_{\mathcal{S}_e} = (1, 2, 2, 16)^T$.

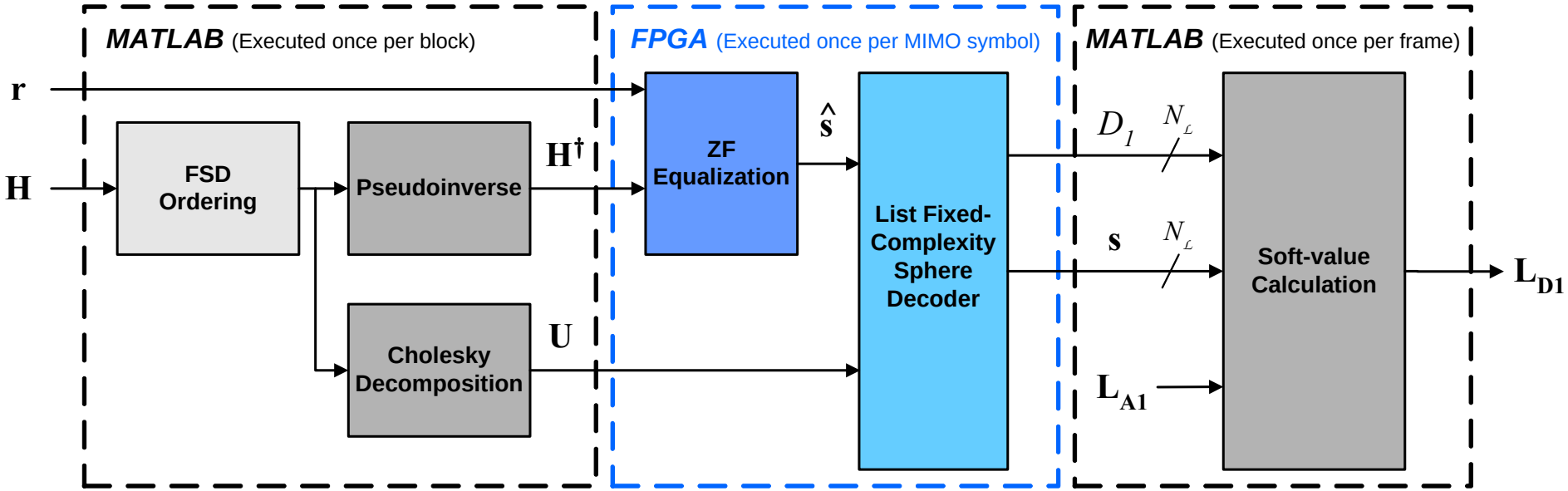




LFSD Performance (2)



FPGA Implementation (LFSD)



FPGA Results (LFSD)

- 4×4 system with 16-QAM modulation.

Xilinx XC2VP70 FPGA	FSD-16	**LFSD-64/16**
Number of slices (33,088)	38% (12,721)	**96% (31,960)**
Number of flip-flops (66,176)	23% (15,332)	**79% (52,719)**
Number of 4-input LUT (66,176)	24% (16,119)	**58% (38,995)**
Number of multipliers (328)	48% (160)	**54% (180)**
Number of block RAM (328)	25% (82)	**15% (52)**
Number of cycles C	4	**8**
Clock frequency $f_{clock}(MHz)$	100 (150)	**100 (150)**
Throughput Q (Mbps)	400 (600)	**200 (300)**

FPGA BER Performance (LFSD)

- Monte Carlo simulations with real-time hardware co-simulation.
- 16 bits used for real and imaginary components quantization.