Software Implementation of near-ML Soft-Output MIMO Detection

SDR'10

30 November – 3 December, 2010

Washington, DC

T. Cupaiuolo and D. Lo Iacono

Advanced System Technology, STMicroelectronics

Outline

- Reduced complexity MIMO detection: The LORD algorithm
- The Block Processing Engine (BPE) architecture
- Mapping of the LORD algorithm over the BPE
- Performance and synthesis results

MIMO detection

- Exhaustive-search Maximum-Likelihood (ML)
  - Find the transmitted sequence minimizing the square norm of the error matrix:

$$\overline{\mathbf{X}} = \arg\min_{\mathbf{X}} \left\| \mathbf{Y} - \sqrt{\frac{E_s}{T}}\mathbf{H}\mathbf{X} \right\|^2$$

  - For $\mathbf{M^2}$-QAM modulation it requires search of $\mathbf{M^{2T}}$ symbols (brute-force approach)
  - Rapidly unfeasible for growing $\mathbf{T}$ transmitting antennas

- Soft-output near-ML MIMO detection: the LORD algorithm
  - High performance gain over state-of-the-art detectors (ZF, MMSE)
  - The hard-output version has performance comparable with that of Sphere Decoder
  - Optimal (ML) max-log soft-output for $\mathbf{T=2}$, near optimal for $\mathbf{T>2}$
  - Candidate list set has linear instead of exponential dependency on $\mathbf{T}$

The LORD algorithm

A. Pre-processing

- Channel estimation matrix **H** decomposition using **H** = **QR** factorization
- Multiple **QR** decomposition (one for each **T**) to compute LLRs efficiently
- Enabling spatial Decision Feedback Equalization (DFE)
- For static channel, computed once per frame (latency with no impact on throughput)

B. Detection

- For each **T** and each **M²-QAM** symbol, compute **M²** Euclidean Distances (EDs) :

$$D_{ED}(\mathbf{x}) = \left\| \tilde{\mathbf{y}} - \mathbf{R}\mathbf{x} \right\|^2$$

C. Soft-output LLR generation

- For each bit $\mathbf{b_i}$ of the constellation, compute the difference between the minimum EDs evaluated over the two sets $S^+(\mathbf{b_i})$ and $S^-(\mathbf{b_i})$ defined by $\mathbf{b_i}=\mathbf{0}$ and $\mathbf{b_i}=\mathbf{1}$ respectively

$$L(b_i|\tilde{\mathbf{y}}) = \min_{x \in S^+} D_{ED}(x) - \min_{x \in S^-} D_{ED}(x)$$

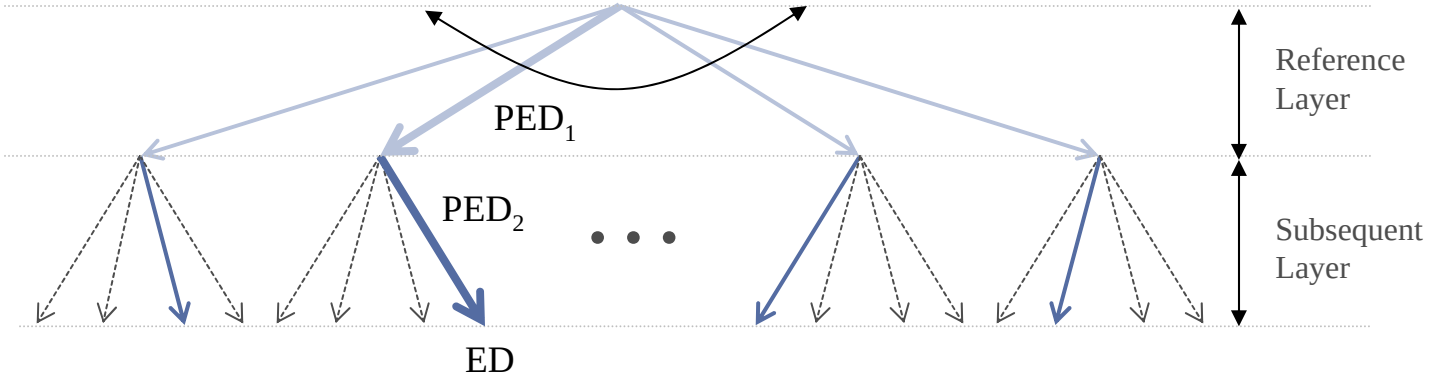
The LORD algorithm – Detection

For each transmitting antenna **T** and for each received symbol, compute **M^2** Euclidean Distances (EDs) to demodulate **M^2-QAM** symbols:

$$D_{ED}(\mathbf{x}) = \| \tilde{\mathbf{y}} - \mathbf{R}\mathbf{x} \|^2$$

For 2 transmitting antennas, it consists of computing:

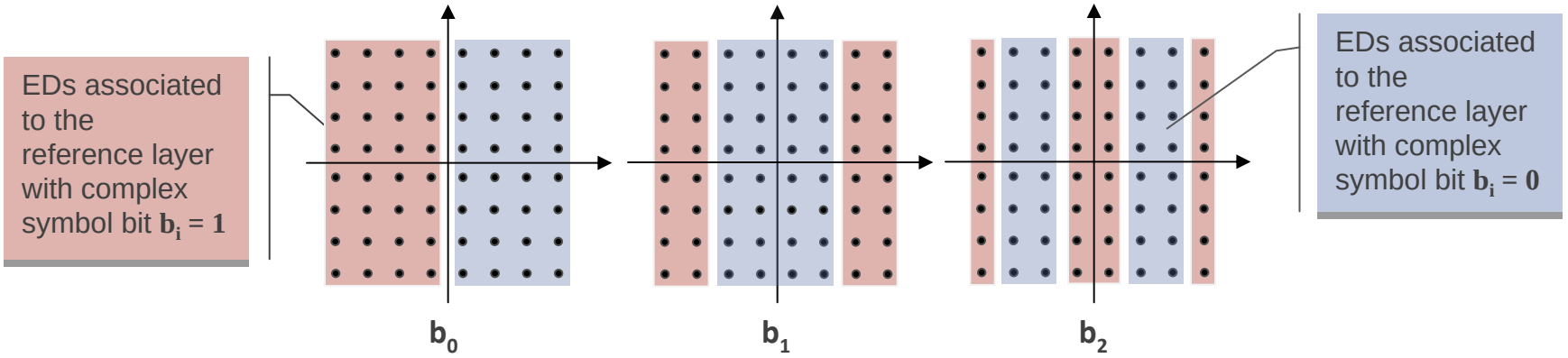
- Partial ED (PED) metrics of the reference layer (**PED_1**)
- Symbol estimate of the subsequent layer based on spatial DFE
- PED metric of the subsequent layer (**PED_2**)
- ED metric as sum of all the PEDs (**$ED = PED_1 + PED_2$**)



The LORD algorithm – Soft-output generation

For each bit $\mathbf{b_i}$ of the $\mathbf{M^2}$-QAM constellation:

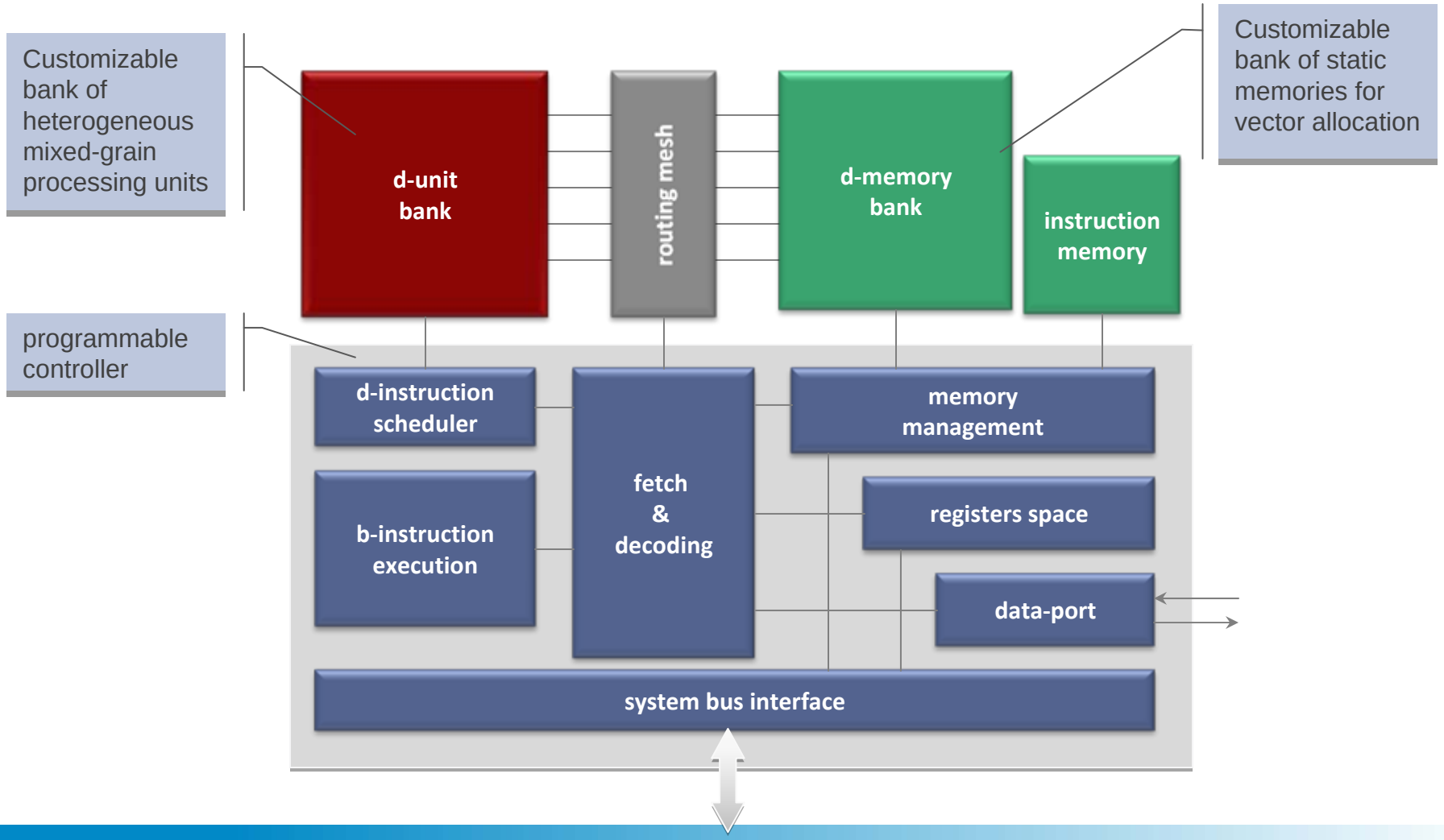
- Identify the two sets $\mathbf{S^+(b_i)}$ and $\mathbf{S^-(b_i)}$ corresponding to $\mathbf{b_i=0}$ and $\mathbf{b_i=1}$ respectively



- Subtract the two minimum EDs evaluated over the sets $\mathbf{S^+(b_i)}$ and $\mathbf{S^-(b_i)}$

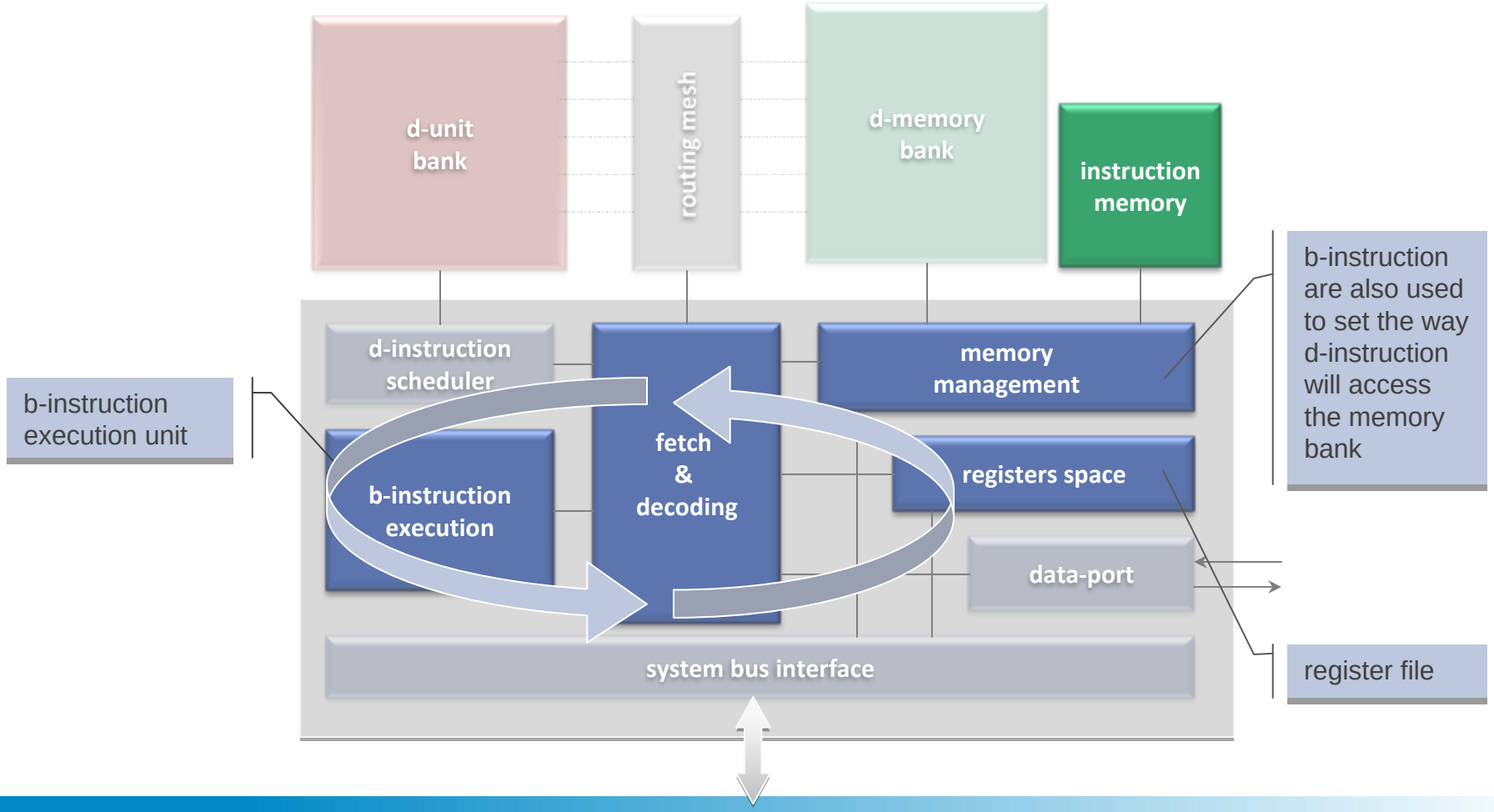
$$L(b_i|\tilde{y}) = \min_{x \in S^+} D_{ED}(x) - \min_{x \in S^-} D_{ED}(x)$$

Block Processing Engine template architecture



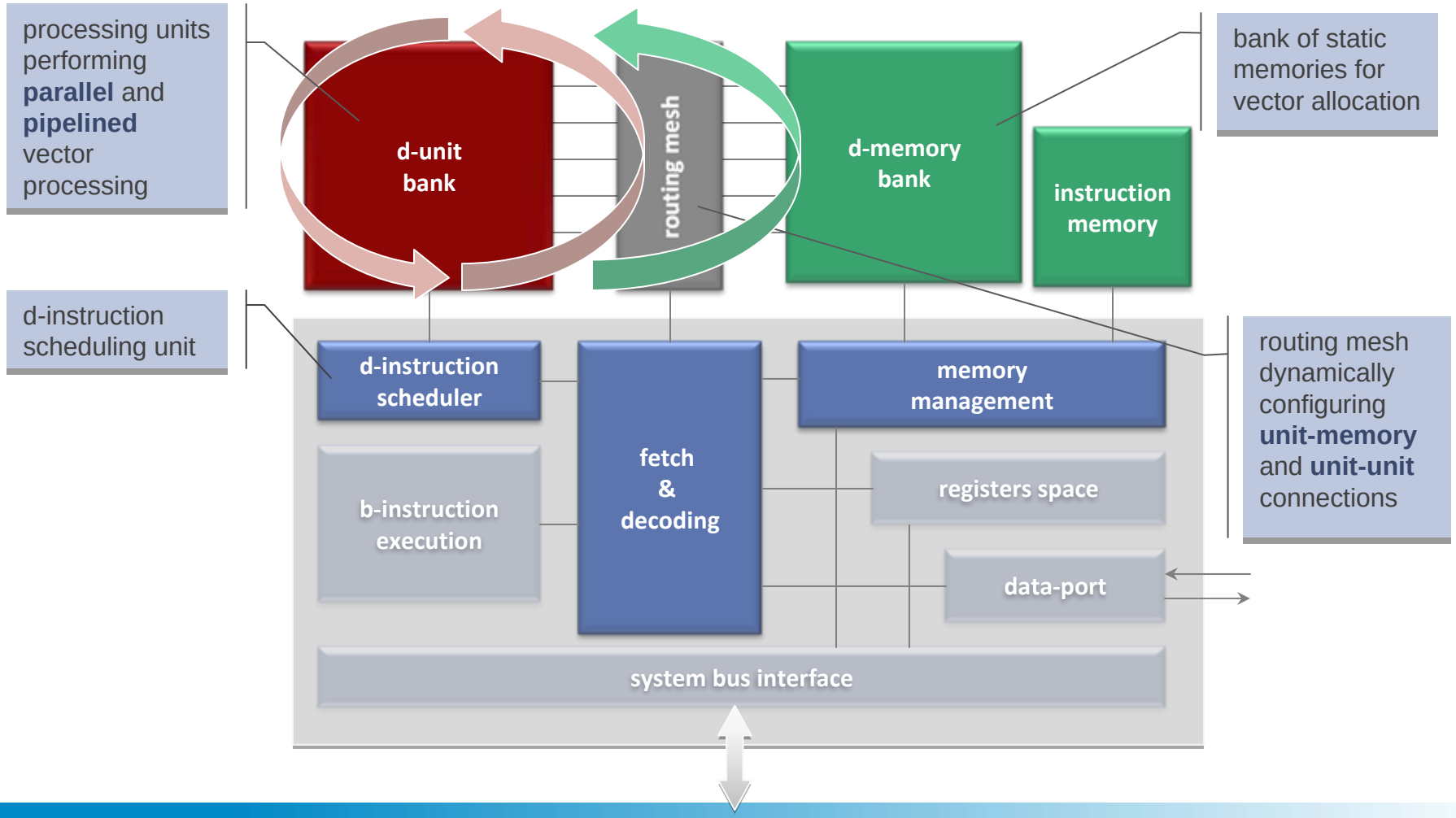
Flow-control: b-instruction

```
out = opcode(in0,in1,in2)
```

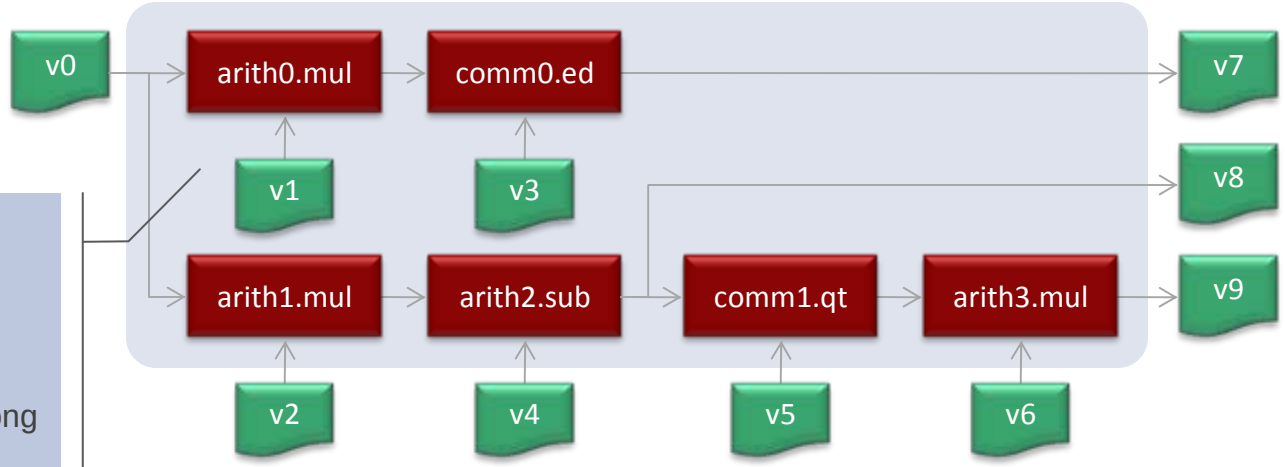


Vector processing: d-instruction

```
out = unit1.opcode(unit0,in0,in1)
```

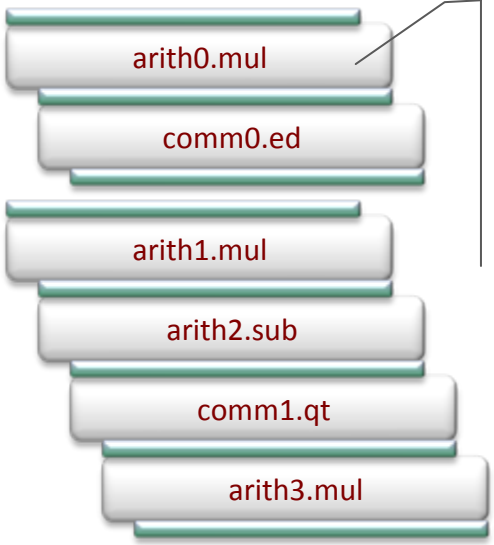


Block diagrams (macros)



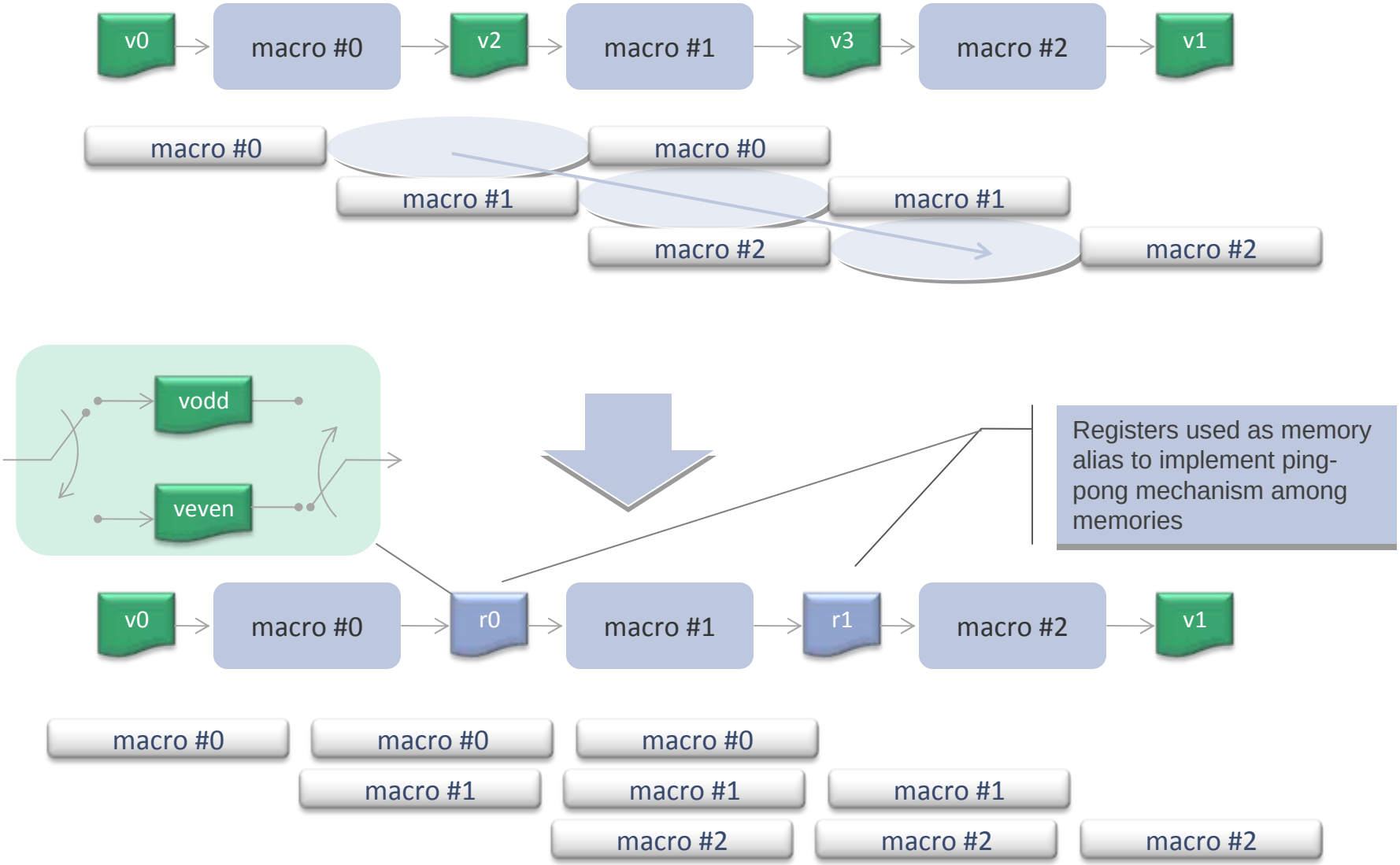
macro made by two **parallel** branches each performing **pipelined** processing among different units

```
v9 = arith3.mul(comm1,v6)
comm1.qt(arith2,v5)
v8 = arith2.sub(v4,arith1)
v7 = comm0.ed(arith0,v3)
arith0.mul(v0,v1);
arith1.mul(v0,v2)
```

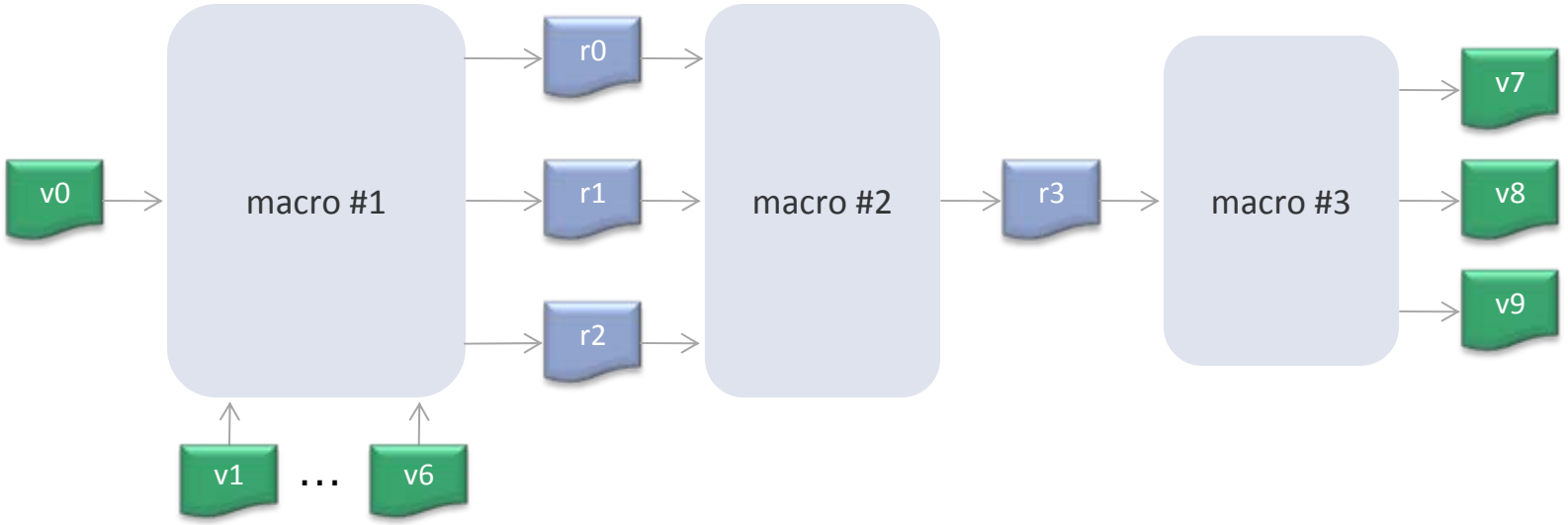


parallel and pipelined processing to reduce **execution time** and **memory accesses**

Pipeline of macros using memory alias



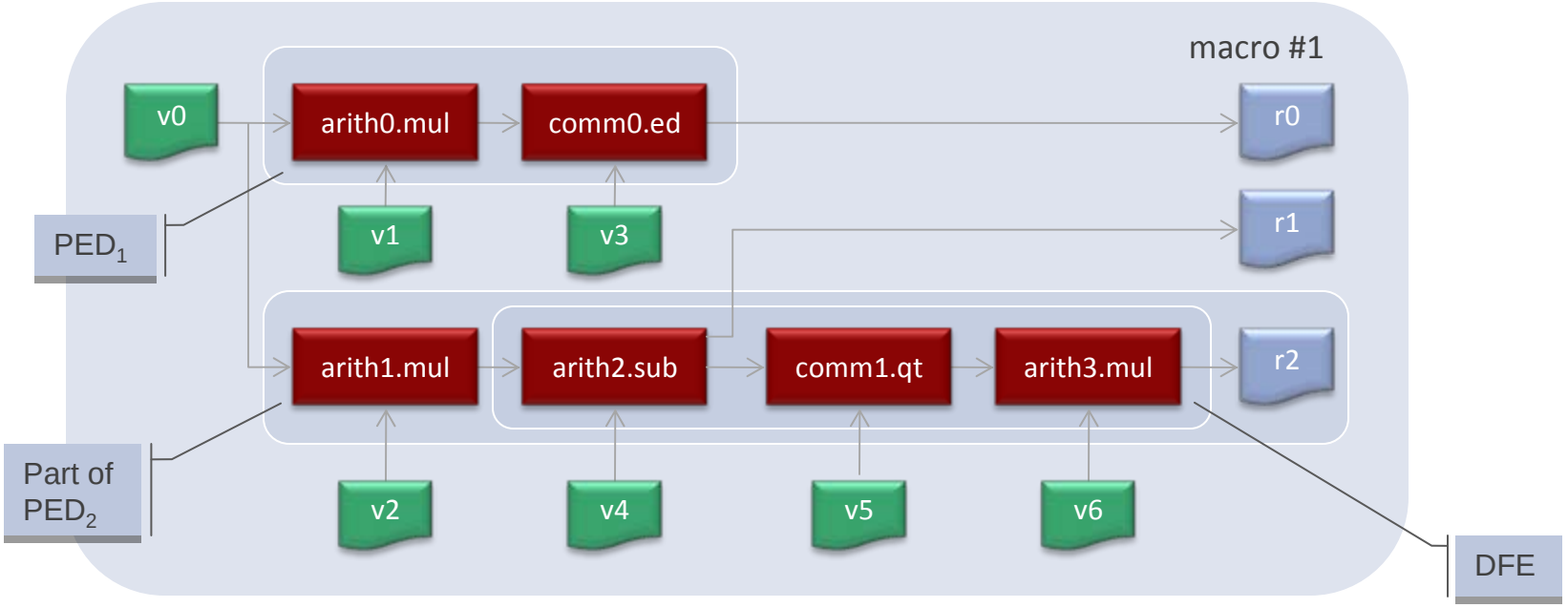
Mapping of the LORD algorithm: a 3-stage pipeline



- Detection
  1. Compute the PED of the reference layer (PED_1), the DFE and (part of) the PED for the subsequent layer (PED_2)
  2. Compute (rest of) PED_2 and the ED from the PED_1
- Soft-output generation
  3. Evaluate the LLRs

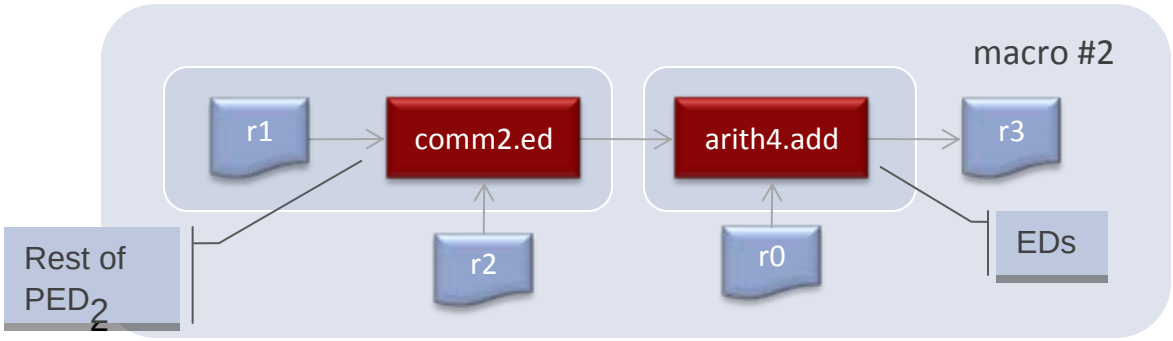
Mapping of the LORD algorithm: macro #1

- Compute the PED of the reference layer (PED_1)
- Compute the DFE and (part of) the PED for the subsequent layer (PED_2)

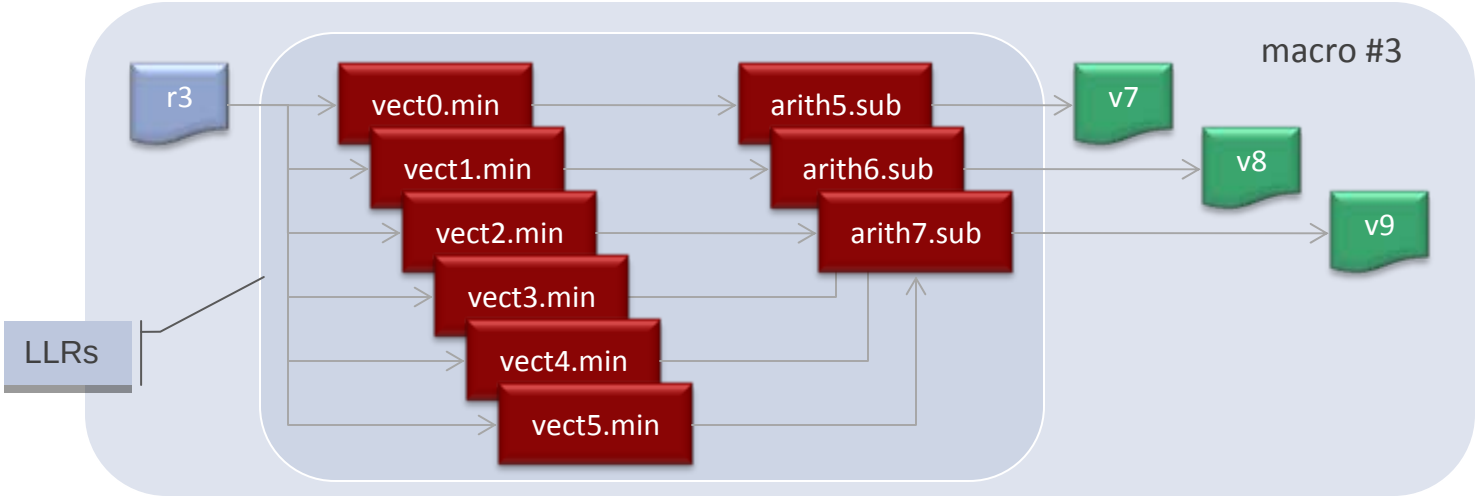


Mapping of the LORD algorithm: macro #2 and #3

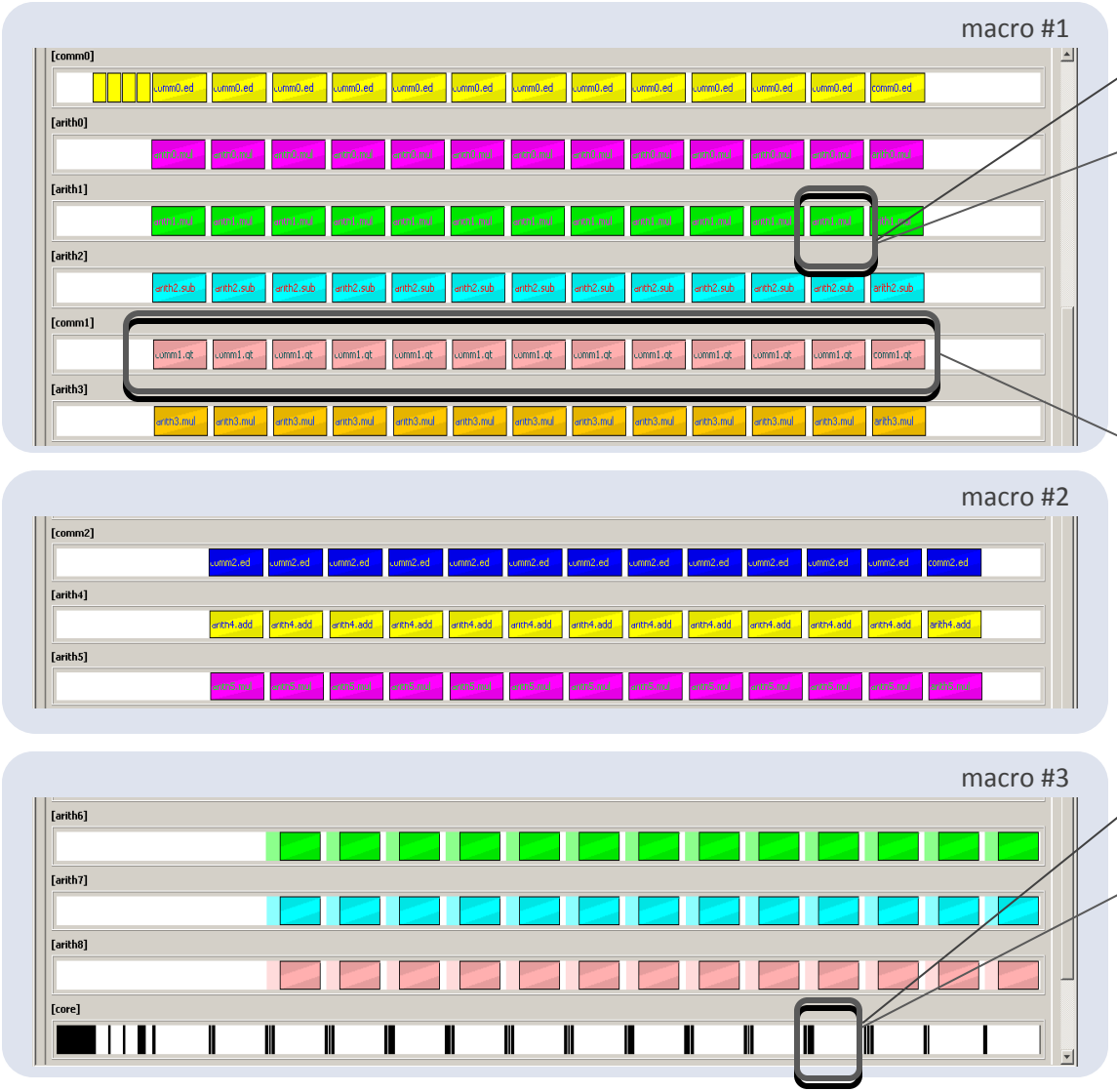
- Compute the PED_2 and the ED from the PED_1 of macro #1



- Generate the LLRs using the EDs computed by macro #2



Mapping of the LORD algorithm: pipeline and timing



vector instruction

Vector length depends on the number of carriers processed with a single instruction
(4 in this case).

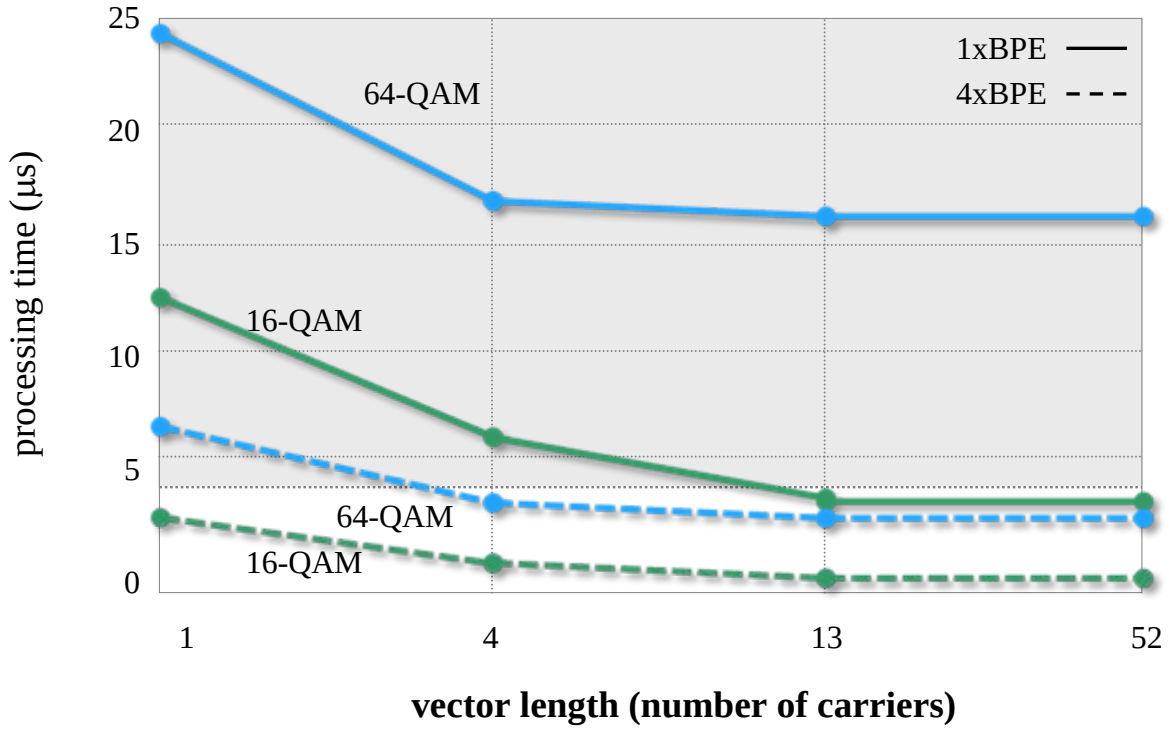
Loop on 13 vector instructions for a total of 52 IEEE 802.11n OFDM symbol carriers

flow-control instruction

For adequate vector lengths the execution of a bunch of basic instructions does not affect total execution time

Performance of LORD on the BPE

- Processing time for one IEEE 802.11n OFDM symbol @400MHz



4µs OFDM symbol duration for IEEE 802.11n

- Real-time capabilities for both 16-QAM and 64-QAM can be achieved using a cluster of 4xBPEs with minimum vector length of 4 carriers

Synthesis results

	BPE	4× BPE
Technology	STMicroelectronics CMOS 65nm	
Area	0.9mm^2	3.6mm^2
Clock	400MHz	
Detector Type	near-ML SO	
Max theoretical Gops	12	48
Near-ML detector Gops	9.6	38.4
Utilization (%)	80	
Throughput [Mbit/s] @ 16-QAM	98	392
Throughput [Mbit/s] @ 64-QAM	38	152