Fast Design Space Exploration using Vivado HLS: Non-Binary LDPC Decoders

Joao Andrade*, Nithin George†, Kimon Karras‡, David Novo†, Vitor Silva*, Paolo Ienne†, Gabriel Falcao*

* Instituto de Telecomunicações, Dept. Electrical and Computer Engineering, Univ. of Coimbra, Portugal
† École Polytechnique Fédérale de Lausanne (EPFL), School of Comp. and Comm. Sciences, Switzerland
‡ Xilinx Research Labs, Dublin, Ireland

Introduction: Non-binary LDPC Decoder on FPGAs

- ▶ We explore a complex error-correction signal processing algorithm:
  - ▷ **non-binary LDPC decoding** (FFT-SPA)

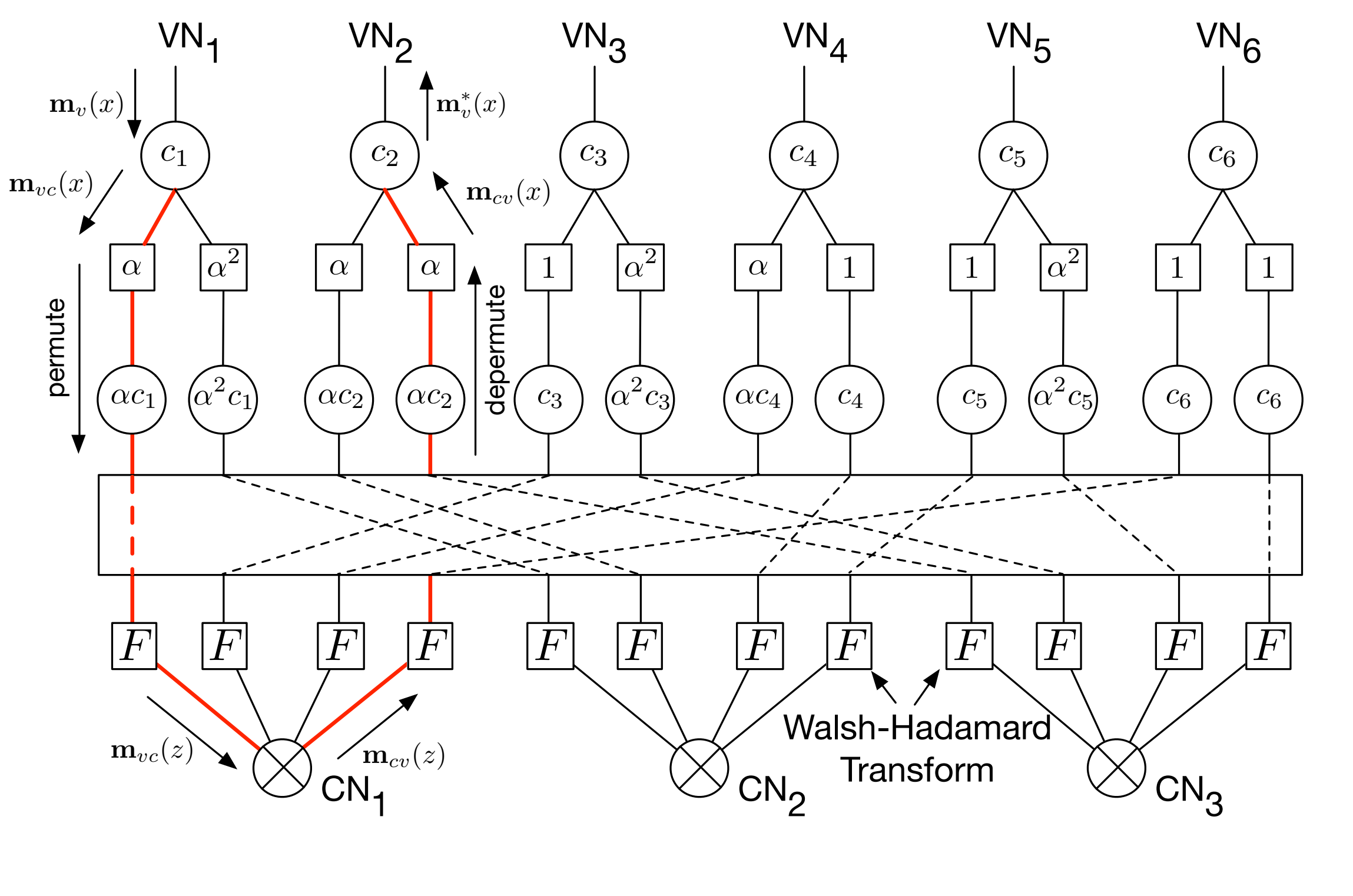

Fig. 1 Non-binary LDPC factor graph example and message-passing algorithm.

- ▶ We utilize a **high-level synthesis** tool to design an LDPC decoder FPGA accelerator
- ▶ **Vivado HLS** allows:
  - ▷ **fast design space exploration** via **directive optimizations**
  - ▷ **C/C++ code** as input for generating an FPGA accelerator

Proposed LDPC Decoder Accelerator

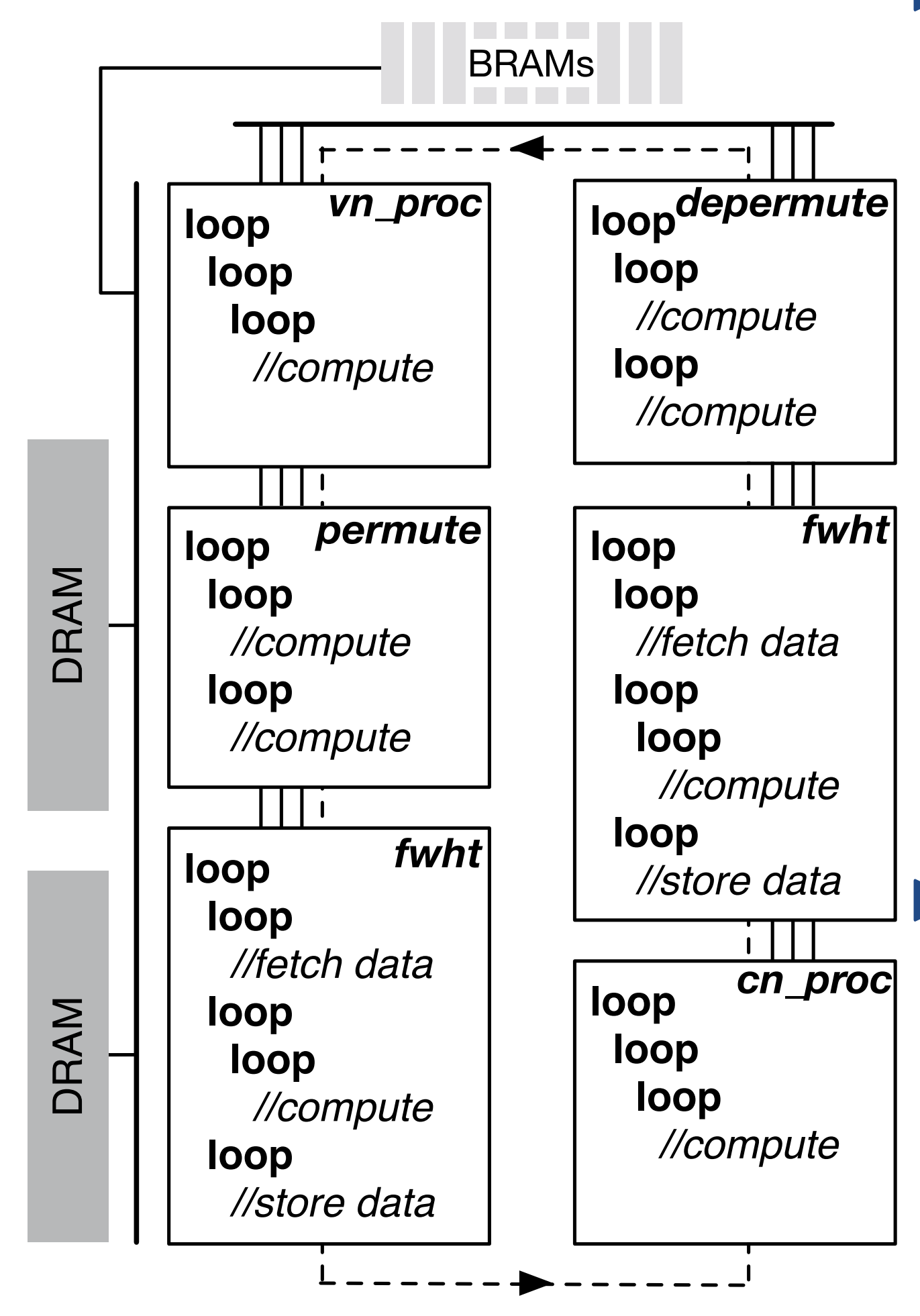
BRAMs
DRAM
DRAM
loop vn_proc loop loop //compute
loop depermute loop //compute loop //compute
loop permute loop //compute loop //compute
loop fwht loop //fetch data loop loop //compute loop //store data
loop fwht loop //fetch data loop loop //compute loop //store data
loop cn_proc loop loop //compute

Fig. 2 Non-binary LDPC decoder base solution block diagram.

- ▶ **LDPC decoder characteristics**
  - ▷ 3-dimensions of computation:
    - ▸ $N \times d/M \times d_c$ *probability mass functions (pmfs)*
    - ▸ 2^m probabilities per *pmf*
    - ▸ d_v/d_c *pmf* per node
    - ▸ 2^m is the Galois field dimension
  - ▷ each dimension is defined over a **computation loop**
- ▶ **Applied LDPC computation:**
  - ▷ Fast Walsh-Hadamard transform (`fwht`)
  - ▷ Hadamard products (`vn/cn_proc`)
  - ▷ Cyclic permutations (`(de)permute`)

- ▶ **Under the hood transformations:**
  - ▷ **3** different nested loop structures:
    - ▸ `cn_proc/vn_proc`: **3** loops triple-nested
    - ▸ `depermute/permute`: **2** loops double-nested
    - ▸ `fwht`: **5** loops triple-nested
  - ▷ no computation performed directly on DRAM data
    $\longrightarrow$ high bandwidth available but **high latency** of access
  - ▷ data **is moved to BRAM memory** for computation at prologue and **to DRAM memory** at epilogue

High-Level Architecture

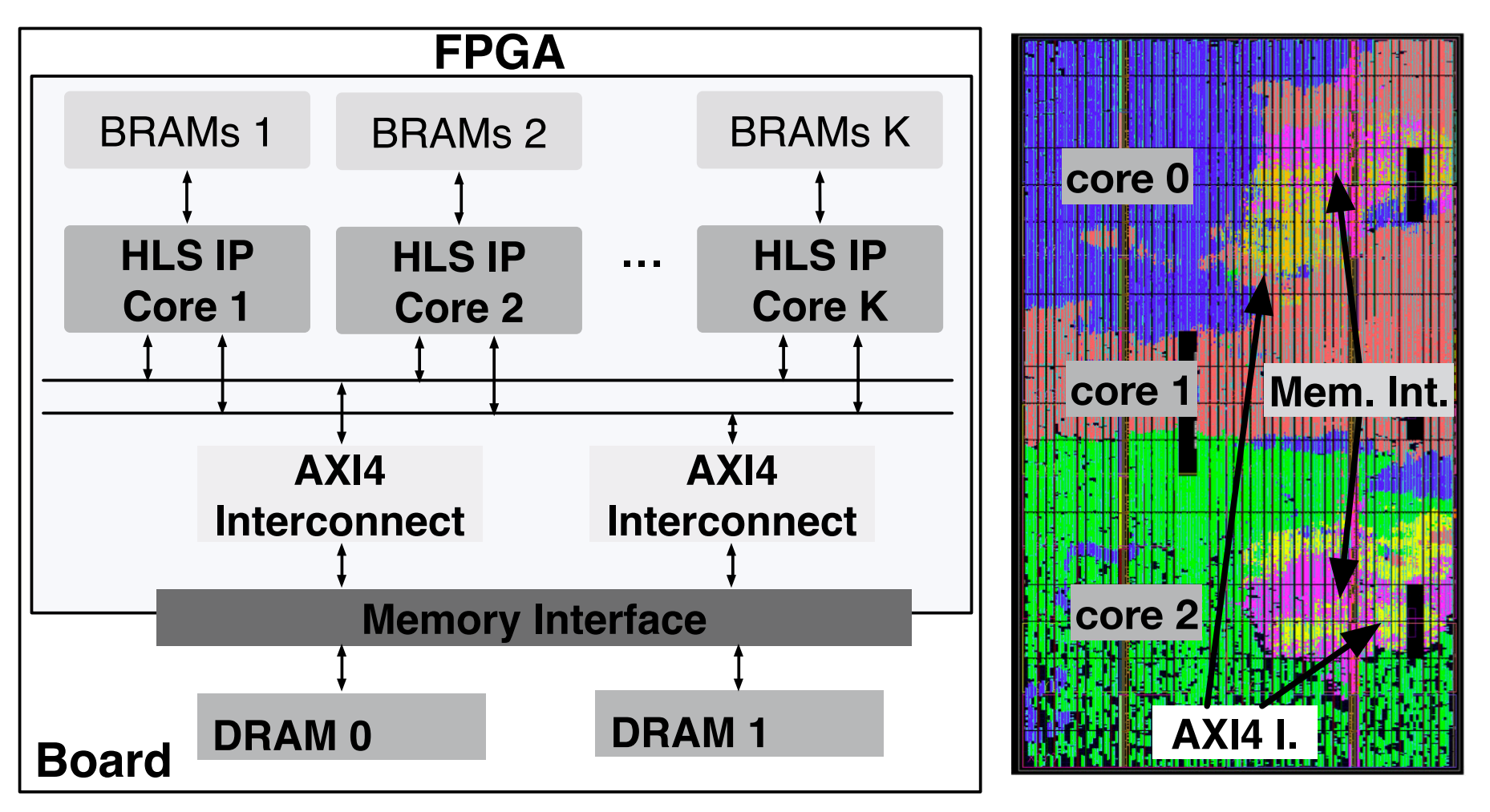

Fig. 3 High-level architecture and die shot with **3** decoders P&R'd.

- ▶ Vivado HLS exports an accelerator design as an IP-XACT **without external I/O**, clock interface or AXI4 data movers
  - ▷ **1 DRAM and AXI-M controllers** per SODIMM (2)
  - ▷ **1 port on AXI-M controllers** per accelerator instantiated (**K**)

Proposed Accelerator Optimizations

Table 2 Optimizations carried out for each solution.

Optimizations	I	II	III	IV	V	VI
Unrolling		✓	✓		✓	✓
Pipelining			✓			✓
Array partitioning				✓	✓	✓

- ▶ We combined the following optimizations to the **6** tested solutions:
  - ▷ loop unrolling (II, V)
  - ▷ loop pipelining (III, VI)
  - ▷ array partitioning (IV, V, VI)
- ▶ Opt. directives are not applied until **code refactoring** in some cases
- ▶ Every dimension where parallelism is exploited must be defined in its particular loop, otherwise unrolling or pipelining becomes unbearable to manage
  - ▷ in fact, some optimization configurations do not complete the C-synthesis
- ▶ **pipeline is targeted at II=1**
- ▶ **unrolling is complete**

Experimental Results: Latency vs. LUTs utilization

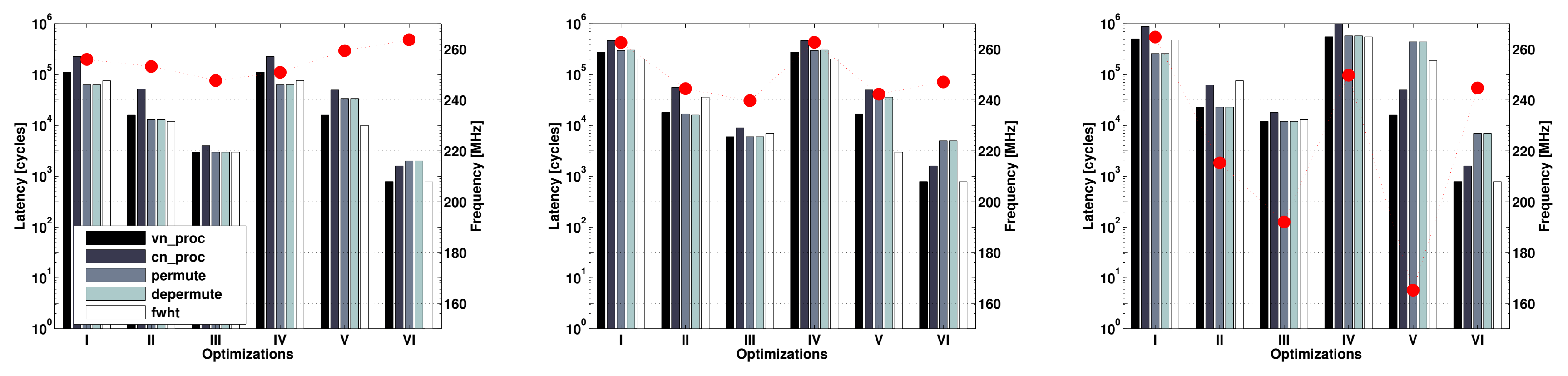

Fig. 4 Latency and clock frequency of operation of each LDPC accelerator solution for GF($\{2^2, 2^3, 2^4\}$).

- ▶ Applying the different optimizations we obtain a set of pareto points with tradeoffs in frequency and LUTs utilization:
  - ▷ providing more memory ports (higher bandwidth) is useful only if ALUs consume data
  - ▷ clock frequencies across the solutions can vary widely (**160∼260**) MHz
  - ▷ pipelining has diminishing returns in latency reduction (`depermute/permute`) for increasing Galois Field dimensions

Comparison with RTL-based Decoders

Table 1 Dec. throughput, FPGA util. and freq. of operation.

Decoder	m	K	LUT [%]	FF [%]	BRAM [%]	DSP [%]	Thr. [Mbit/s]	Clk [MHz]
This work	2	1	14	7	0.5	0.5	1.17	250
		14	80	35	6	6	14.54	219
	3	1	21	9	0.9	0.9	0.95	250
		6	81	34	5	5	4.81	210
	4	1	30	13	2	2	0.66	216
		3	73	32	5	5	1.85	201
Emden @ ISTCIIP'10	2		N/A				33.16	100
	4						13.22	
	8						1.56	
Zhang @ TCS–I'11	4		48 (Slices)		41	N/A	9.3	N/A
Boutillon, @ TCS–I'13	6		19	6	1	N/A	2.95	61
Scheiber @ ICECS'13	1		14 (Slices)		21	N/A	13.4	122
Andrade @ ICASSP'14	8	1	85 (LEs)		62	7	1.1	163

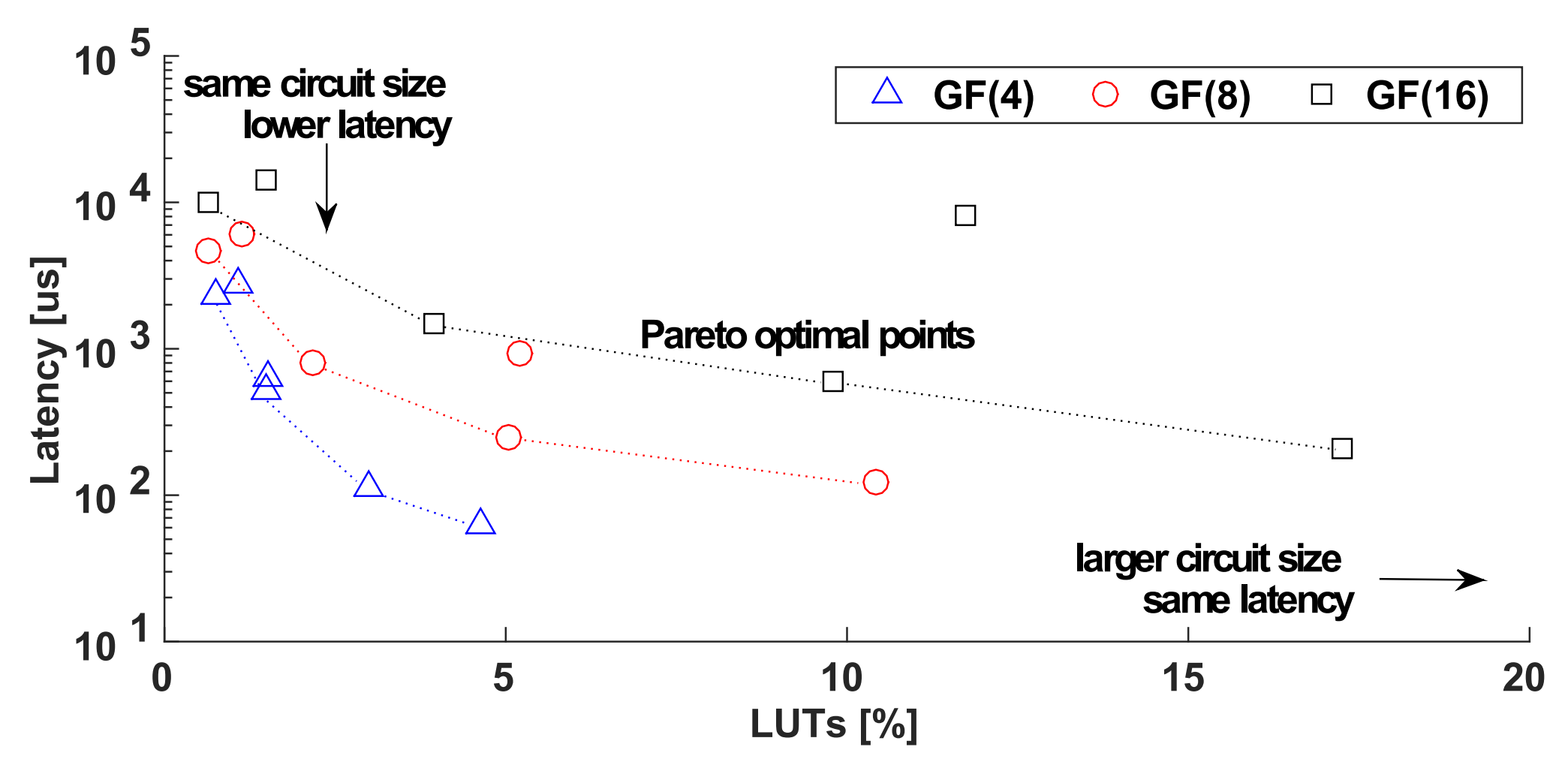

Fig. 5 Pareto and non-Pareto optimization points measured in latency (μs) vs. LUTs utilization (%).

- ▶ LUT utilization grows with the Galois Field dimension
  - ▷ Pareto points observed clearly illustrate the diminishing returns in the **latency for LUTs tradeoff**
- ▶ We can settle for the optimized solution VI and increase the number **K** of instantiated LDPC decoder accelerators on the high-level architecture
- ▶ RTL-based circuits still achieve higher performances but we reach quite close even though **HLS is being used**
  - ▷ **approx. 50% dec. throughput**
  - ▷ but only for several **K** instantiated decoders

Conclusions

- ▶ We show that combining the correct optimizations we are able to reach within **50% of RTL-based LDPC decoders**
- ▶ Programming language is the same but programming model is different
  - ▷ Code refactoring **is still required**
  - ▷ Exploited parallelism dimensions are exposed in proper loop structures
- ▶ By instantiating the accelerators in a suitable high-level architecture we are able to fit multiple accelerators **further elevating the parallelism level**

This work supported by the Portuguese Fundação para a Ciência e Tecnologia (FCT) under grants UID/EEA/50008/2013 and SFRH/BD/78238/2011.