UNIVERSITEIT GENT

FACULTY OF ENGINEERING AND ARCHITECTURE

A fully parameterized Virtual Coarse-Grained Reconfigurable Arrays for HPC Applications

Amit Kulkarni (Amit.Kulkarni@UGent.be)
Elias Vansteenkiste and Dirk Stroobandt
Ghent University - Belgium

Andreas Brokalakis and Antonios Nikitakis
Synilixis Solutions Ltd - Greece

EXTRA
www.extrahpc.eu

IPDPS 2016 CHICAGO

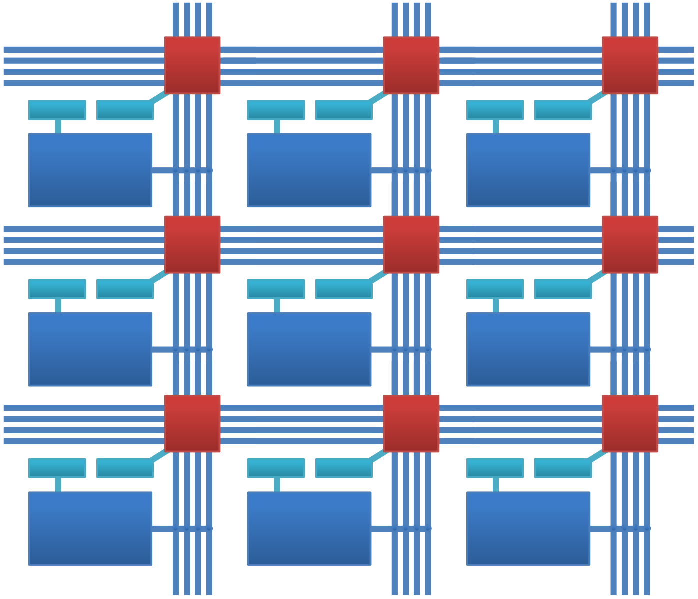
RAW 2016

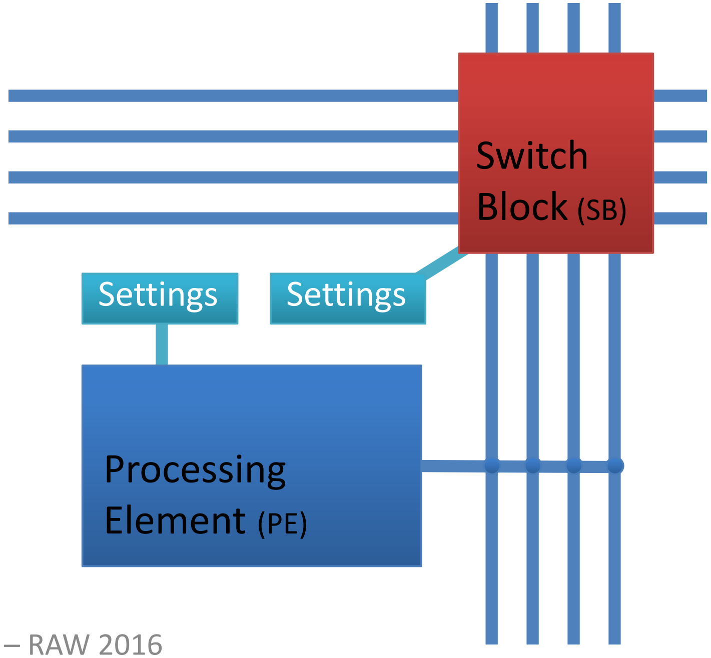
Virtual Coarse-Grained Reconfigurable Array (VCGRA)

CGRA: Domain-specific architecture

Virtual CGRA: CGRA implemented on FPGA

Example: CGRA of mathematical operators to compute mathematical functions





Why use a VCGRA?

CGRA versus processor:

- Better performance, power efficiency, …

CGRA versus FPGA/ASIC:

- Higher abstraction level for programmers
- Higher abstraction level for development tools (faster P&R tools)

  Lower development cost

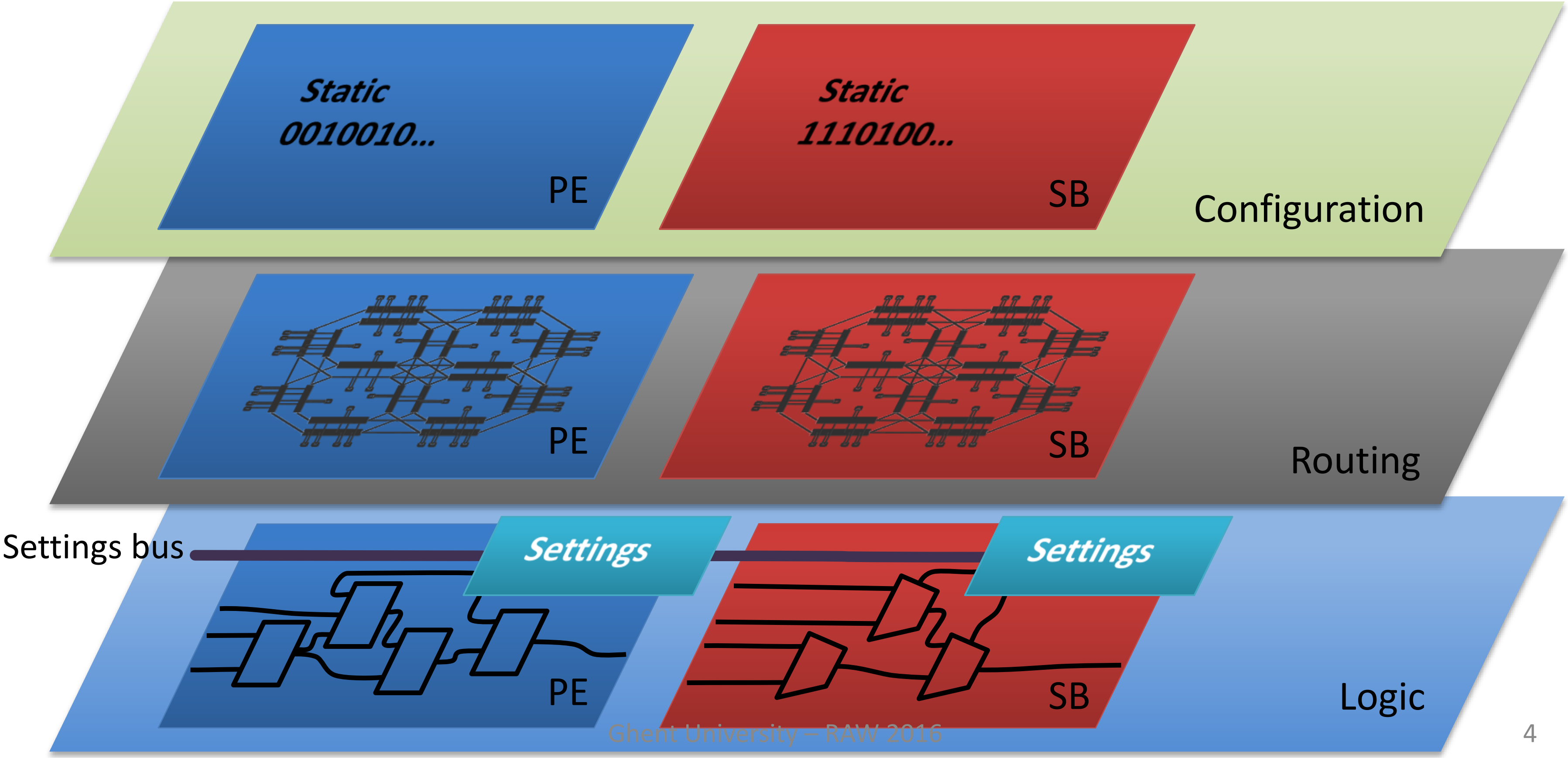
Implemented on FPGA:

- Advantages of (customized) CGRA without cost of ASIC

Conventional implementation of PEs and SBs on FPGA

Static FPGA-configuration

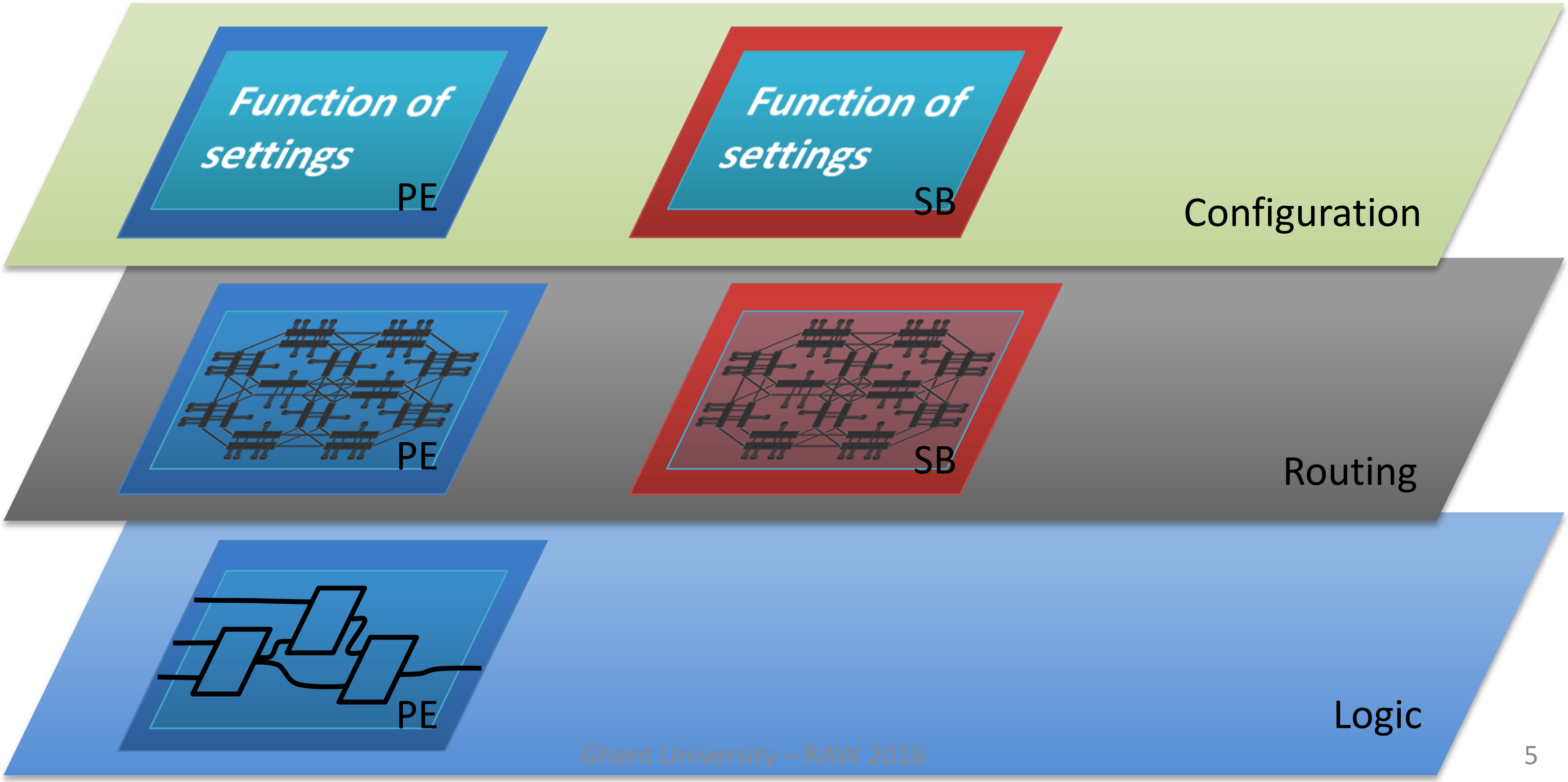
Application updated by writing settings registers



Implementation of PEs and SBs using Parameterized Configurations

Configuration is function of settings

Application updated by reconfiguring FPGA



Function of settings

PE

Function of settings

SB

Configuration

PE

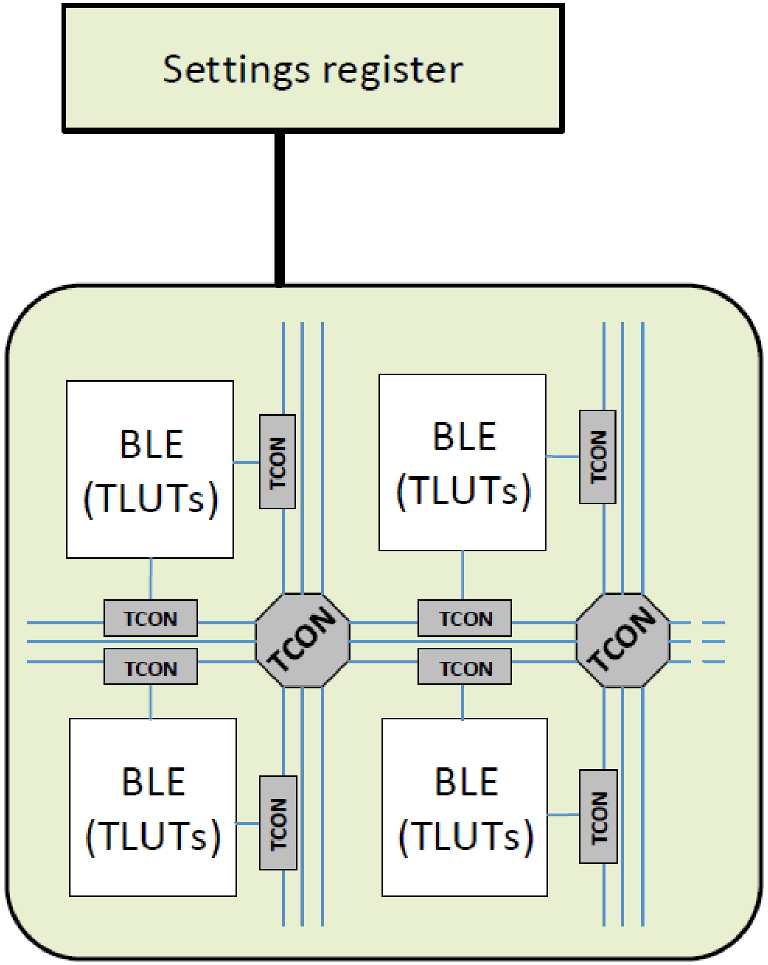
SB

Routing

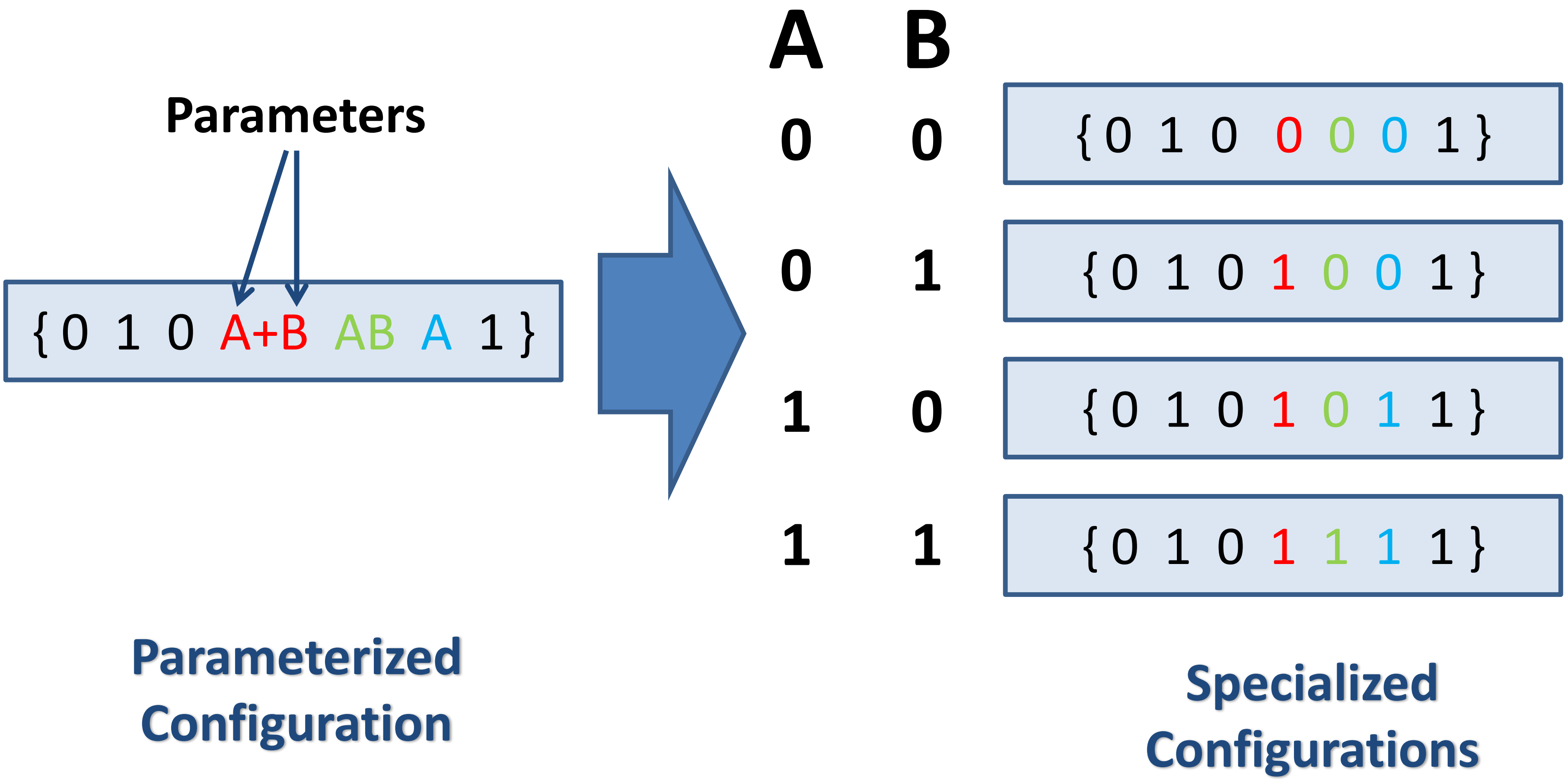
PE

Logic

Parameterizing the intra-connects of the PE



Parameterized Configuration



Parameters

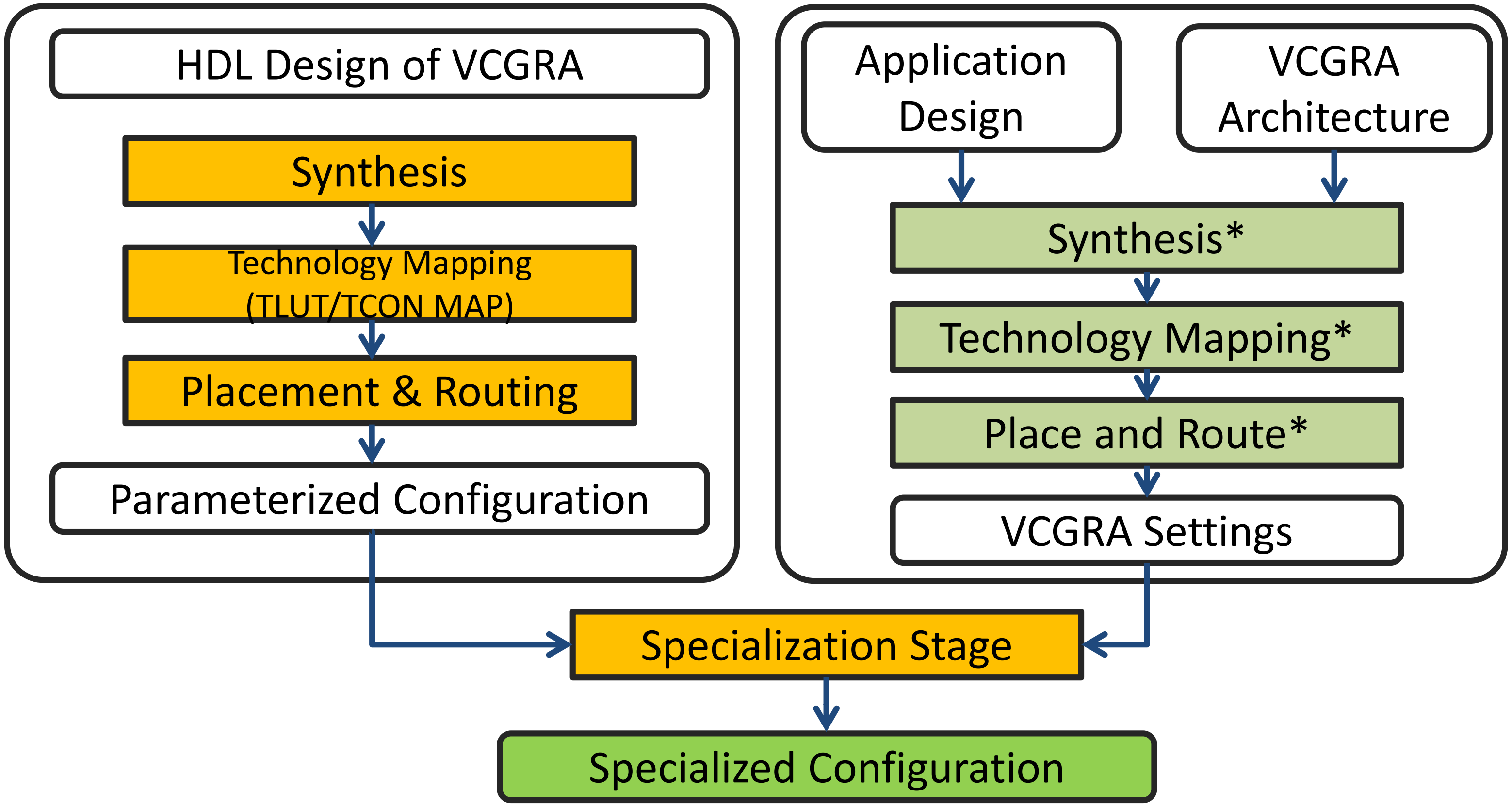
{0 1 0 A+B AB A 1}

Parameterized Configuration

A	B	
0	0	{0 1 0 0 0 0 1}
0	1	{0 1 0 1 0 0 1}
1	0	{0 1 0 1 0 1 1}
1	1	{0 1 0 1 1 1 1}

Specialized Configurations

Parameterized Configurations VCGRA tool flow



*Usually custom synthesis, mapping and P&R tools

Why (not) use Parameterized Configurations?

Lower implementation overhead of VCGRAs

Computing intensive FPGA tool flow outside application development cycle

Slower reconfiguration of VCGRA

No cycle-by-cycle context switching

Tools currently work for academic FPGA architectures and (with limitations) for Xilinx FPGAs

HPC: Retinal Vessel Segmentation

Resource utilization and PaR results of a PE (MAC) operator

VCGRA	LUTs (TLUTs)	TCONs	Logic depth level
Conventional	2522 (0)	0	36
Fully Parameterized	1802 (*526*)	*568*	*33*

30% fewer slices, 31% fewer intra-connects, lower delay

4x4 VCGRA Grid	Inter-Network	Settings Register
Conventional	41	25
Fully Parameterized	*0*	*0*

- No logic resources (LUTs and FFs) used for switch blocks
- No settings registers and shift-register chain

Conclusion

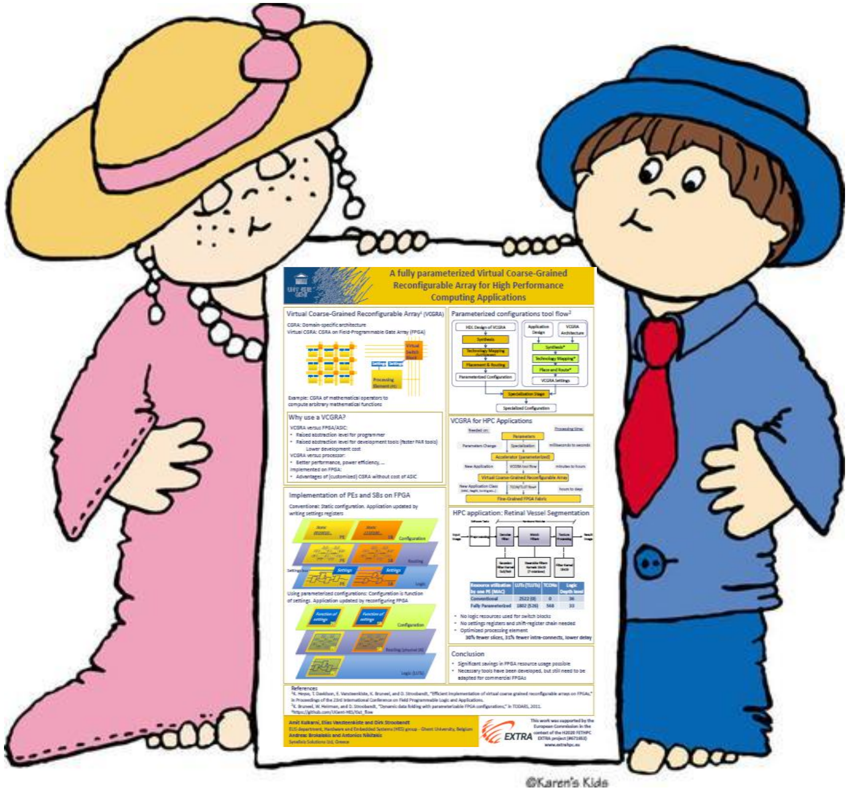
Applied the technique of Parameterized configurations to VCGRAs

Significantly more efficient implementation of VCGRAs on FPGA possible

Necessary algorithms have been developed, but tools still need to be adapted for commercial FPGAs

Thank you

- For more info please visit my poster



A fully parameterized Virtual Coarse-Grained Reconfigurable Array for High Performance Computing Applications

Virtual Coarse-Grained Reconfigurable Array[1] (VCGRA)

CGRA: Domain-specific architecture
Virtual CGRA: CGRA on Field-Programmable Gate Array (FPGA)

Example: CGRA of mathematical operators to compute arbitrary mathematical functions

Why use a VCGRA?

VCGRA versus FPGA/ASIC:

- Raised abstraction level for programmer
- Raised abstraction level for development tools (faster PAR tools)
  Lower development cost

VCGRA versus processor:

- Better performance, power efficiency, ...

Implemented on FPGA:

- Advantages of (customized) CGRA without cost of ASIC

Implementation of PEs and SBs on FPGA

Conventional: Static configuration. Application updated by writing settings registers

Using parameterized configurations: Configuration is function of settings. Application updated by reconfiguring FPGA

Parameterized configurations tool flow[2]

VCGRA for HPC Applications

HPC application: Retinal Vessel Segmentation

Resource utilization by one PE (MAC)	LUTs (TLUTs)	TCONs	Logic Depth level
Conventional	2522 (0)	0	36
Fully Parameterized	1802 (526)	568	33

- No logic resources used for switch blocks
- No settings registers and shift-register chain needed
- Optimized processing element
  30% fewer slices, 31% fewer intra-connects, lower delay

Conclusion

- Significant savings in FPGA resource usage possible
- Necessary tools have been developed, but still need to be adapted for commercial FPGAs

References:
[1] K. Heyse, T. Davidson, E. Vansteenkiste, K. Bruneel, and D. Stroobandt, "Efficient implementation of virtual coarse grained reconfigurable arrays on FPGAs," in Proceedings of the 23rd International Conference on Field Programmable Logic and Applications.
[2] K. Bruneel, W. Heirman, and D. Stroobandt, "Dynamic data folding with parameterizable FPGA configurations," in TODAES, 2011.
[3] https://github.com/UGent-HES/tlut_flow

Amit Kulkarni, Elias Vansteenkiste and Dirk Stroobandt
ELIS department, Hardware and Embedded Systems (HES) group - Ghent University, Belgium
Andreas Brokalakis and Antonios Nikitakis
Synelixis Solutions Ltd, Greece

EXTRA
This work was supported by the European Commission in the context of the H2020 FETHPC EXTRA project (#671653)
www.extrahpc.eu

©Karen's Kids