Dynamic Partial Reconfiguration of Customized Single-Row Accelerators

Nuno Paulino, João Canas Ferreira, João M.P. Cardoso
nmcp@fe.up.pt
REC 2019, 14-15 February 2019

SPeCS
Special-Purpose Computing Systems, languages and tools

Motivation and Objectives

Embedded applications → Demanding performance requirements

- Possible solution: design of specialized circuits for execution of *"hot"* kernels

Issue: Manual hardware design error-prone and laborious

- Solution provided by previous work
  - Detection of CDFGs from simulated execution of embedded applications
  - Generation of **tailored Loop Accelerator** for embedded system
  - Geometric mean speedups between 1.47x and 18.98x for 24 benchmarks
  - **Issue:** *for too many loop kernels → large area required & drops in frequency*

Approach

Use Dynamic Partial Reconfiguration → **Reduce required area/resources**

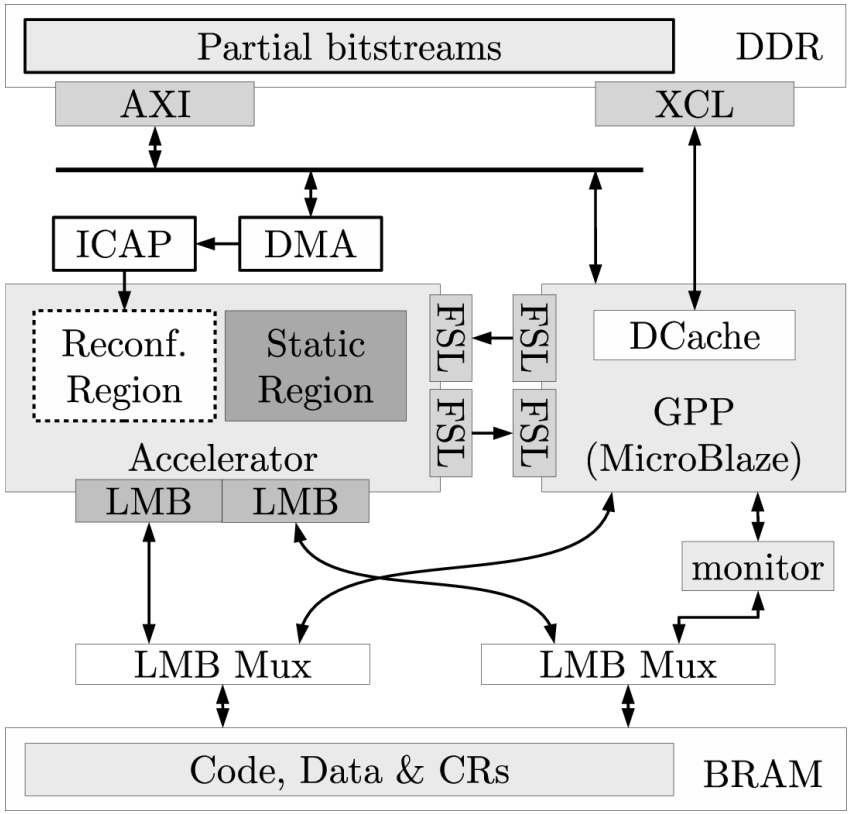
Approach & System Architecture

General Approach

1) Detect loop kernels (MegaBlocks [1])
2) Generate Loop Accelerator instance
   - Multi-config with or w/use of **DPR**
3) Intercept kernel execution at runtime
4) Migrate execution to Loop Accelerator

System Architecture

- Single MicroBlaze main processor
  - Executes unmodifed binary code
- Customized Loop Accelerator (CLA)
  - Automatically generated from CDFGs
- Runtime migration of execution to CLA
- Shared local data/code memory



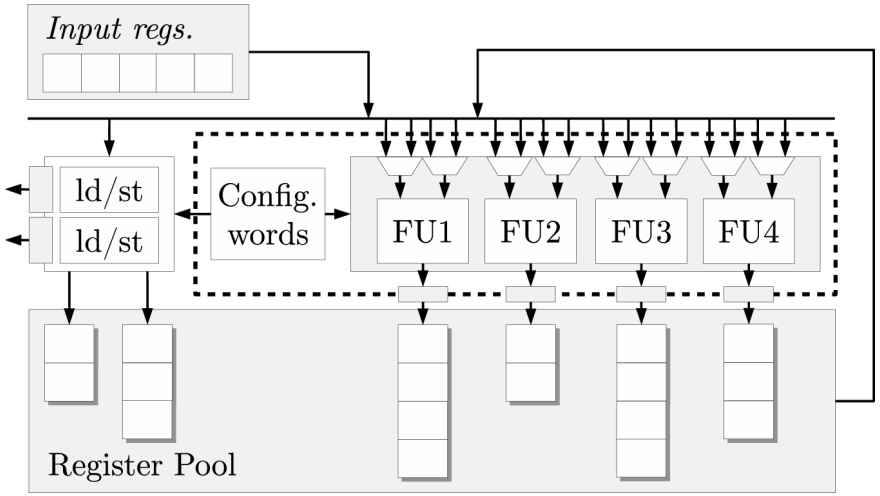
Loop Accelerator Architecture

- Customized Loop Accelerator (CLA)
  - Modulo-scheduled loop execution

- Reconfigurability
  - **Via DPR**
  - Via config. words & muxes

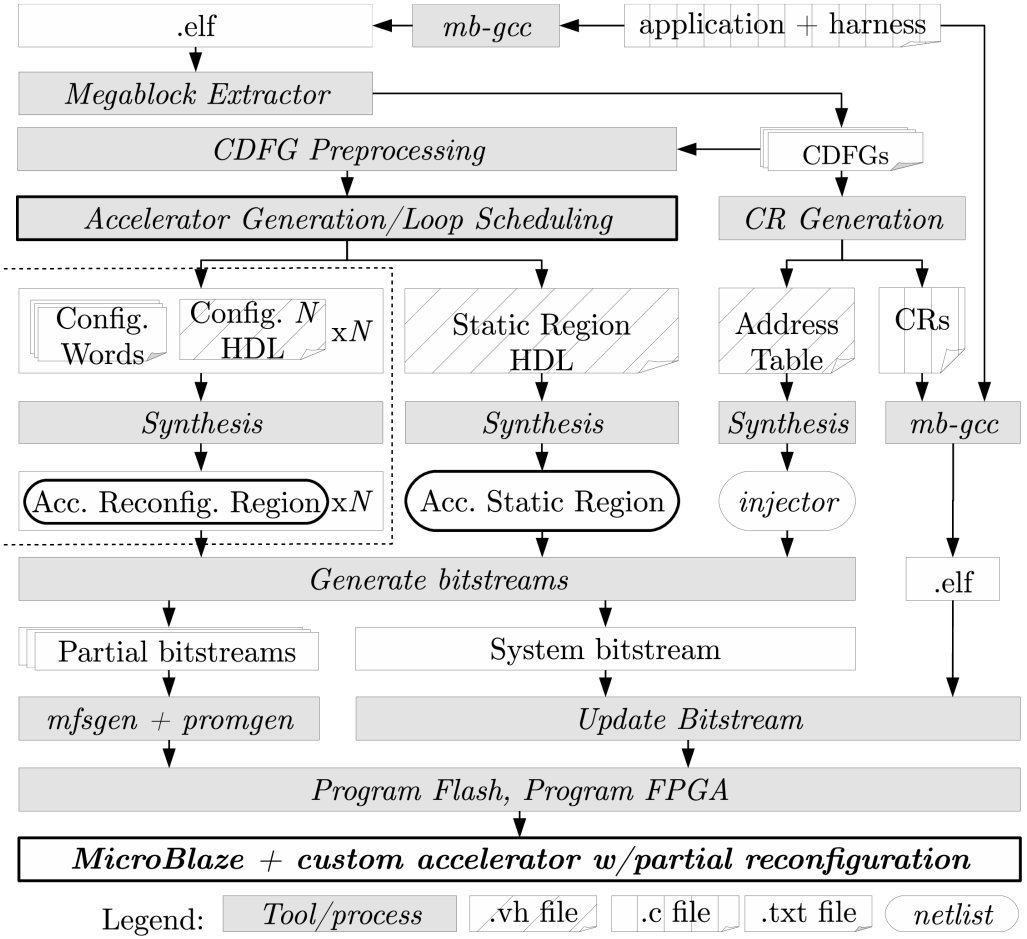
- Static Region
  - Input/output registers
  - Register pool
  - 2 LD/ST units

- Reconfigurable Region
  - Functional Units
  - Multiplexers
  - Configuration memory

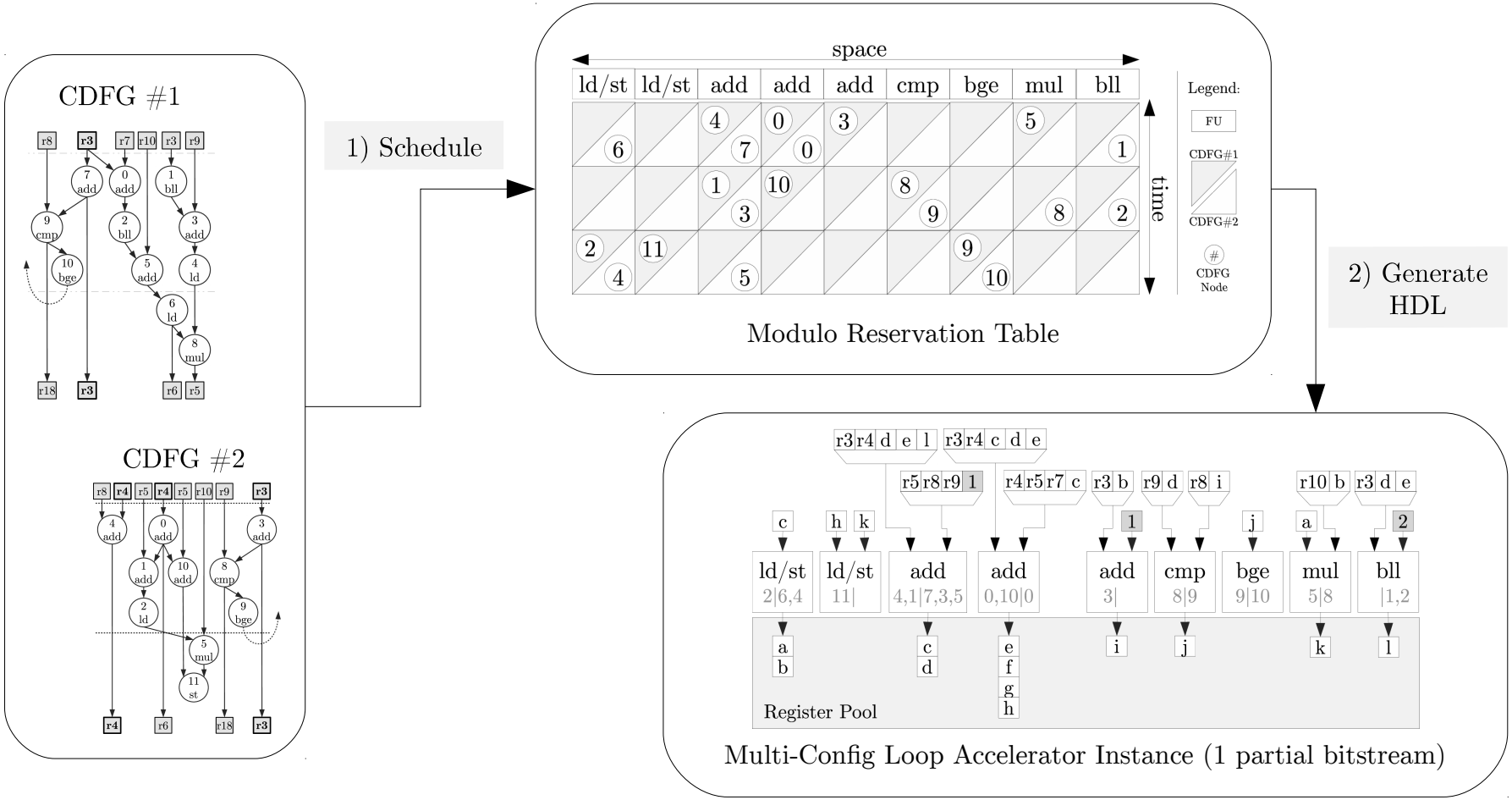


Toolflow

- Up to N reconf. region variants
  - From: 1 Variant with N configs
  - To: N Variants with 1 Config



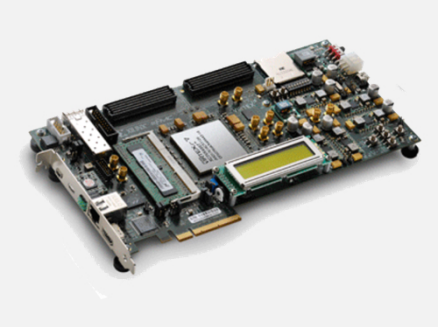
Toolflow - Example



Experimental Setup (1/2)

- Xilinx VC707 Evaluation Board
  - Virtex-7 VX485T FPGA

- 21 integer and floating-point loop kernels
  - Livermore Loops
  - Texas Instruments IMGLIB



- Three sets of kernel groups
  - Small → 2 to 3 loops → 8 groups
  - Medium → 5 to 6 loops → 4 groups
  - Large → 10 to 11 loops → 2 groups

- Each group was used to generate CLA instances tailored to execute the loops, by:
  1) Resorting to **a single partial bitstream** → Faster switching; more area required
  2) Resorting to **multiple bitstreams** → Area savings; frequency decrease less likely

- Each implementation required specifying an **area size** for the reconfigurable region
- All implementations targeted 100Mhz for clock signal driving the entire system

Experimental Setup (2/3)

Kernel groupings

- One CLA is generated per kernel group (e.g. *s1)*
- Average # insts: 35
- Average II: 7 clock cycles
- All kernels are scheduled **at their minimum II**, regardless of the use of DPR
- More kernels
  → greater area *if* DPR is not used

kernel	#insts	II	small	medium	big
quantize	10	3	s1	m1	b1
perimeter	21	3			
boundary	24	8			
conv3x3	64	10	s2		
sad16	13	2			
mad16	13	2			
sobel	40	5	s3	m2	
dilate	142	29			
erode	142	29			
innerprod	10	3	s4		
matmul	15	3			
hydro	17	3	s5	m3	b2
hydro2d	37	6			
innerprod_fp	10	4	s6		
matmul_fp	15	3			
intpredict	41	10	s7	m4	
diffpredict	44	10			
glinearrec	13	3			
cholesky	20	3	s8		
statefrag	42	5			
tridiag	12	3			

Experimental Setup (3/3)

Floorplanning for **static** and **reconfigurable** regions of the CLA

Seven area sizes for reconfigurable region

Area Size	Name	Reconfigurable Area	
		Slices	Circuit Area
Small	sml1	882	1.06%
	sml2	1248	1.53%
Medium	med1	1568	1.93%
	med2	2254	2.80%
Large	big1	3136	3.92%
	big2	5928	7.49%
	big3	9400	11.94%

- Two auxiliary floorplanning areas for static logic of CLA
  1) Place static logic on perimeter of reconfigurable logic
  2) Overlap BRAMs to reconfigurable logic

Each group of kernel loops can target any area size, with the ideal solution being that which allows for the least possible used area

Results – Area (1/2)

Kernel Set	Average Resource Usage of Allocated Area for **Single Partial Bitstream (No DPR)**		
	Slices / (%)	LUTs / (%)	Impl. Time (m)
Small set	1667 / (91%)	5760 / (78%)	93.7
Med. set	2669 / (97%)	9753 / (90%)	44.6
Big set	6395 / (83%)	18855 / (62%)	70.1

Kernel Set	Average Resource Usage of Allocated Area for **Multiple Partial Bitstreams (DPR)**		
	Slices / (%)	LUTs / (%)	Impl. Time (m)
Small set	1195 / (84%)	3825 / (69%)	57.0
Med. set	1660 / (91%)	5139 / (70%)	89.1
Big set	2073 / (92%)	6405 / (71%)	162.6

- The reserved area for the reconfigurable region of the CLA decreases considerably when resorting to DPR
  - For the big kernel set, the reserved area decreases from *big2/3* to *med2*

Results – Area (2/2)

Avg. Resource Requirements
Normalized to MicroBlaze Processor for:

A) Single Partial Bitstream

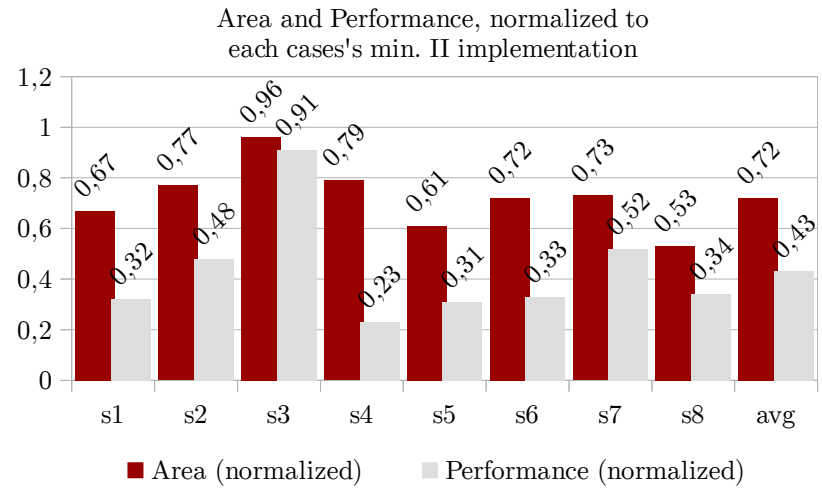
	Static Area			Reconfigurable Area		
	Slices	LUTs	FFs	Slices	LUTs	FFs
sml	0.40	0.16	0.87	1.32	3.25	0.05
med	0.77	0.19	1.55	2.01	5.50	0.07
big	1.56	0.45	3.03	**4.81**	10.64	0.09

B) Multiple Partial Bitstreams

	Static Area			Reconfigurable Area		
	Slices	LUTs	FFs	Slices	LUTs	FFs
sml	0.31	0.14	0.65	0.95	2.16	0.05
med	0.52	0.15	0.96	1.32	2.90	0.06
big	0.63	0.19	1.17	**1.64**	3.61	0.08

Increasing Initiation Interval to Reduce Area Usage

- Geo. mean speedup for minimum II ([2])
  - 6.61x for floating-point loops
  - 4.08x for integer loops
- Increasing II may allow for reutilization of Functional Units when modulo scheduling the kernel loops



Conclusion

- Customized Loop Accelerator
  - Mixed-grain reconfiguration via fast context switching and/or **Dynamic Partial Reconfiguration**

- **4.3x smaller** area using DPR vs. mux-based multi-config CLA
- Using DPR does not increase the II of each supported loop
- Decreasing the II of loop can lead to further reductions of area

- **Open issue:** for a set of loops, how many partial bitstreams to generated in order to minimize area without introducing penalizing overhead? I.e., how many circuits to generate, and which loops should each circuit support?

[1] *Bispo, J.; Cardoso, J.M.P., "On identifying and optimizing instruction sequences for dynamic compilation", in International Conference on Field-Programmable Technology (FPT) 2010*

[2] *N. M. C. Paulino, J. C. Ferreira and J. M. P. Cardoso, "Generation of Customized Accelerators for Loop Pipelining of Binary Instruction Traces", in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 25, no. 1, pp. 21-34, Jan. 2017.*

Thank You

Questions?