

New Successes for Parameterized Run-time Reconfiguration

(or: use the FPGA to its true capabilities)

Prof. Dirk Stroobandt

Ghent University, Belgium

Hardware and Embedded Systems group

Outline

- What is Parameterized Run-time Reconfiguration?
- **Profiling** your applications for parameters
- Improvements in **reconfiguration procedure**
- **Routing** for parameterized reconfiguration
- Addressing conventional **modular reconfiguration**

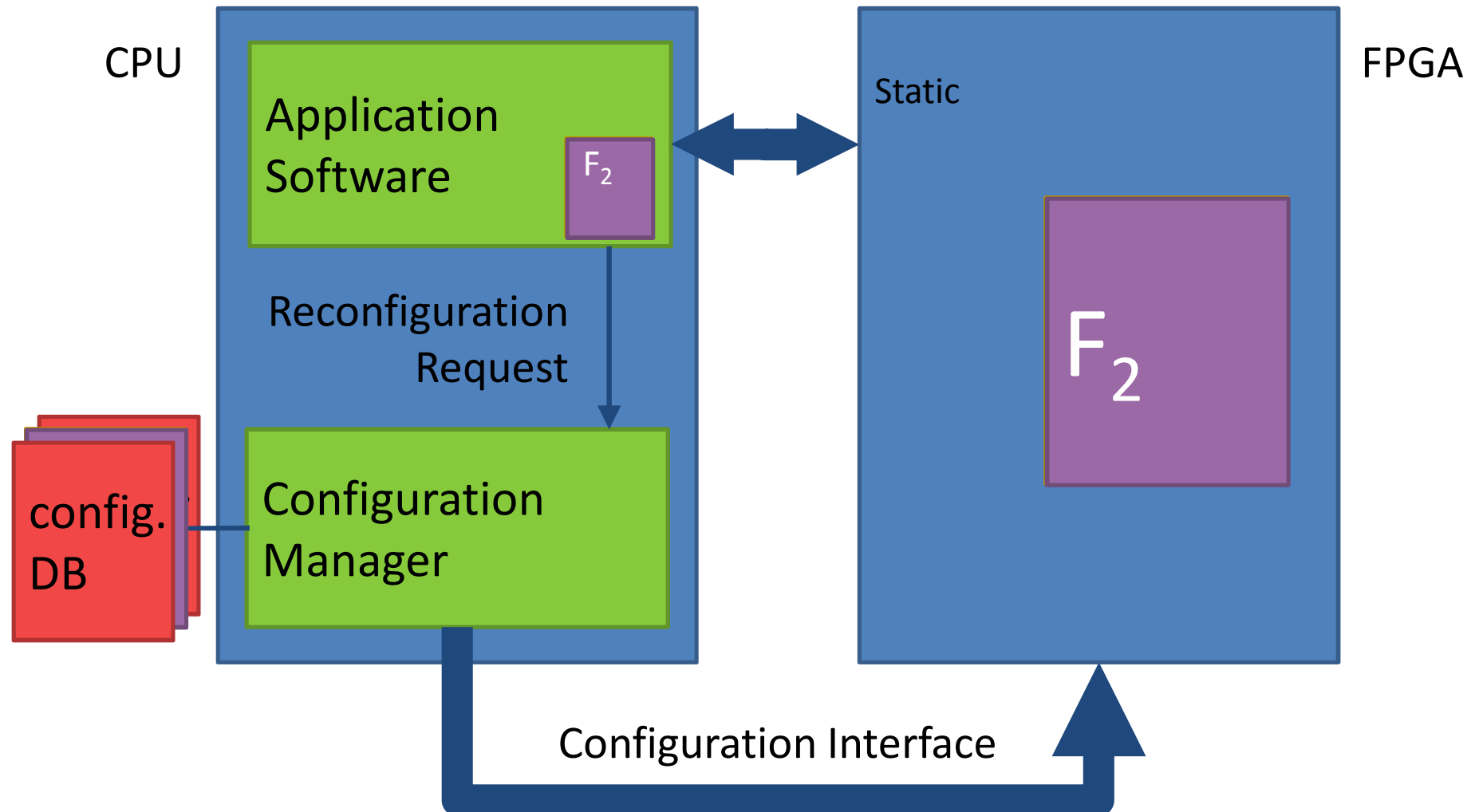
Outline

- **What is Parameterized Run-time Reconfiguration?**
- Profiling your applications for parameters
- Improvements in reconfiguration procedure
- Routing for parameterized reconfiguration
- Addressing conventional modular reconfiguration

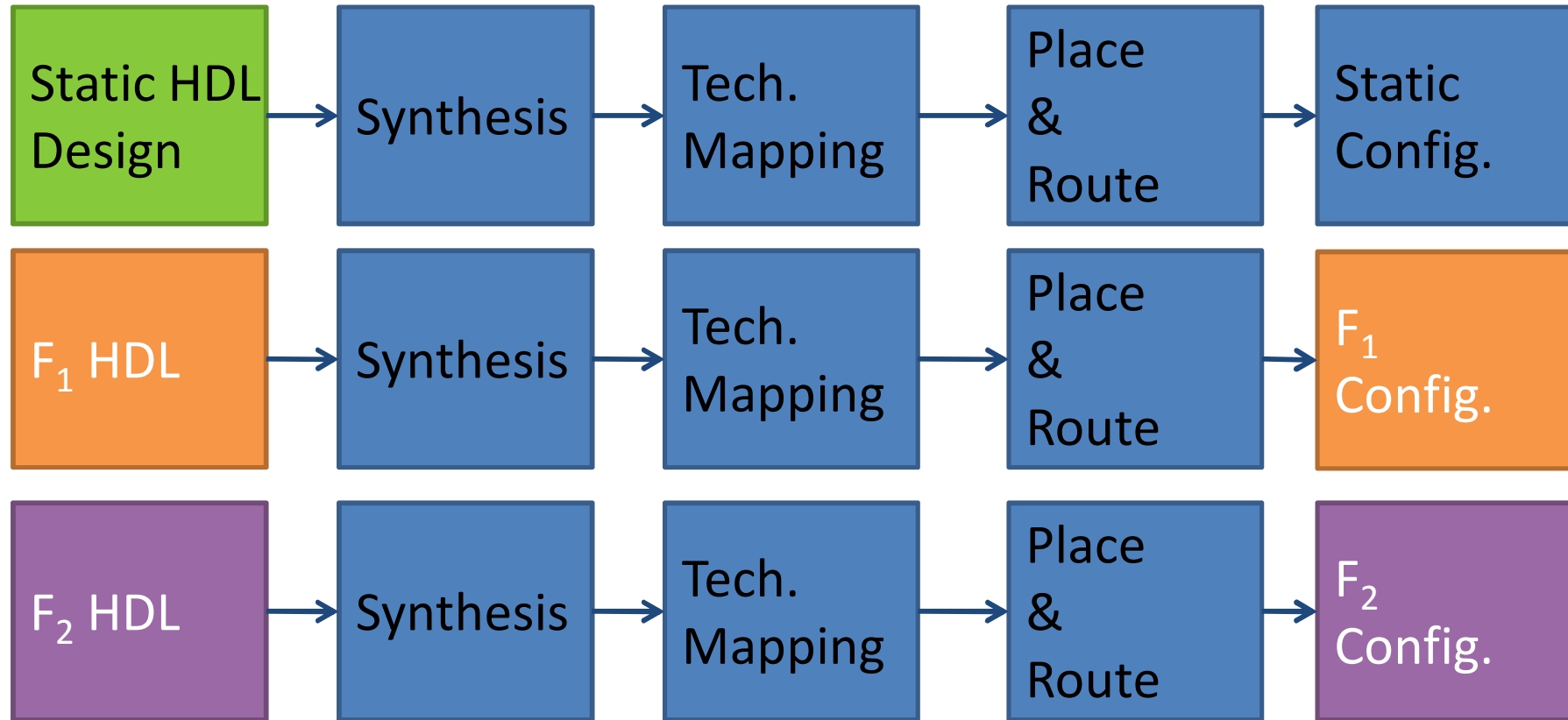
Run-Time Reconfiguration?

- Today: configurability on a **large time scale**
 - Prototyping
 - System update
 - ...
- We: configurability on a **smaller time scale**
 - Dynamic circuit specialization
 - Frequently changing (regular) inputs vs. infrequently changing **parameters**
 - Parameters trigger a reconfiguration (through **configuration manager**)
 - Goals:
 - Improve performance
 - Reduce area
 - Minimize design effort

Conventional Dynamic Reconfiguration



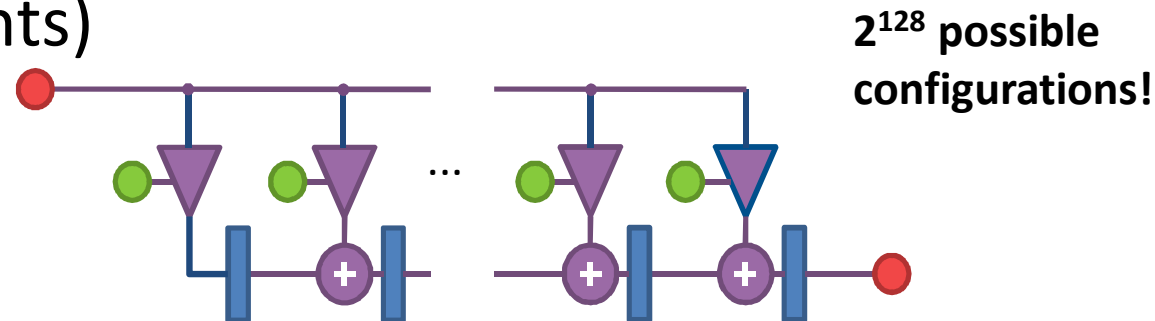
Conventional Tool Flow



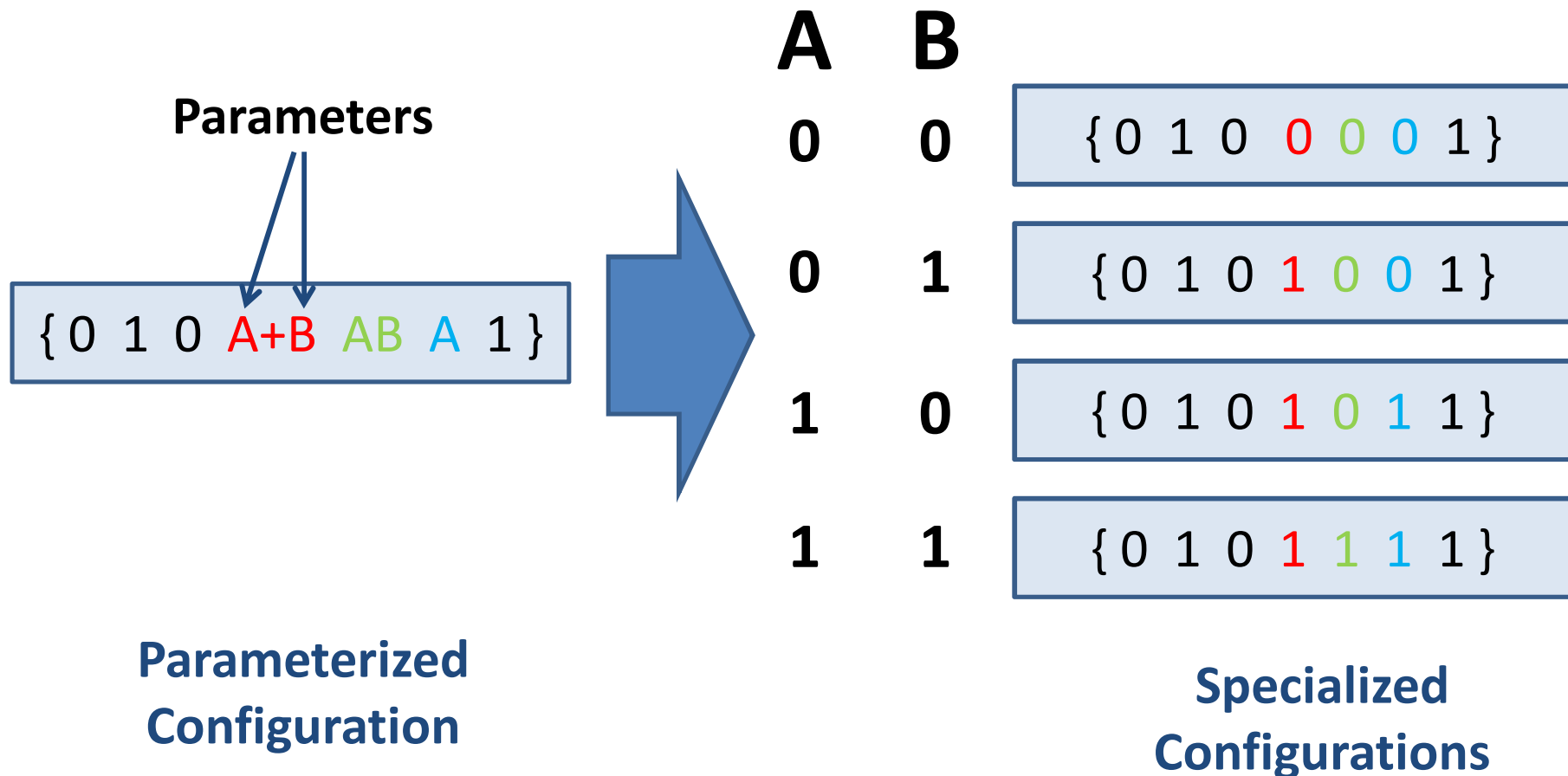
Dynamic Circuit

Specialization not feasible!

- Application where part of the input data changes infrequently
 - Conventional implementation (no reconfiguration):
Generic circuit, Store data in memory, Overwrite memory
 - Dynamic circuit specialization:
Reconfigure with configuration specialized for the data
- Example: Adaptive FIR filter (16-tap, 8-bit coefficients)

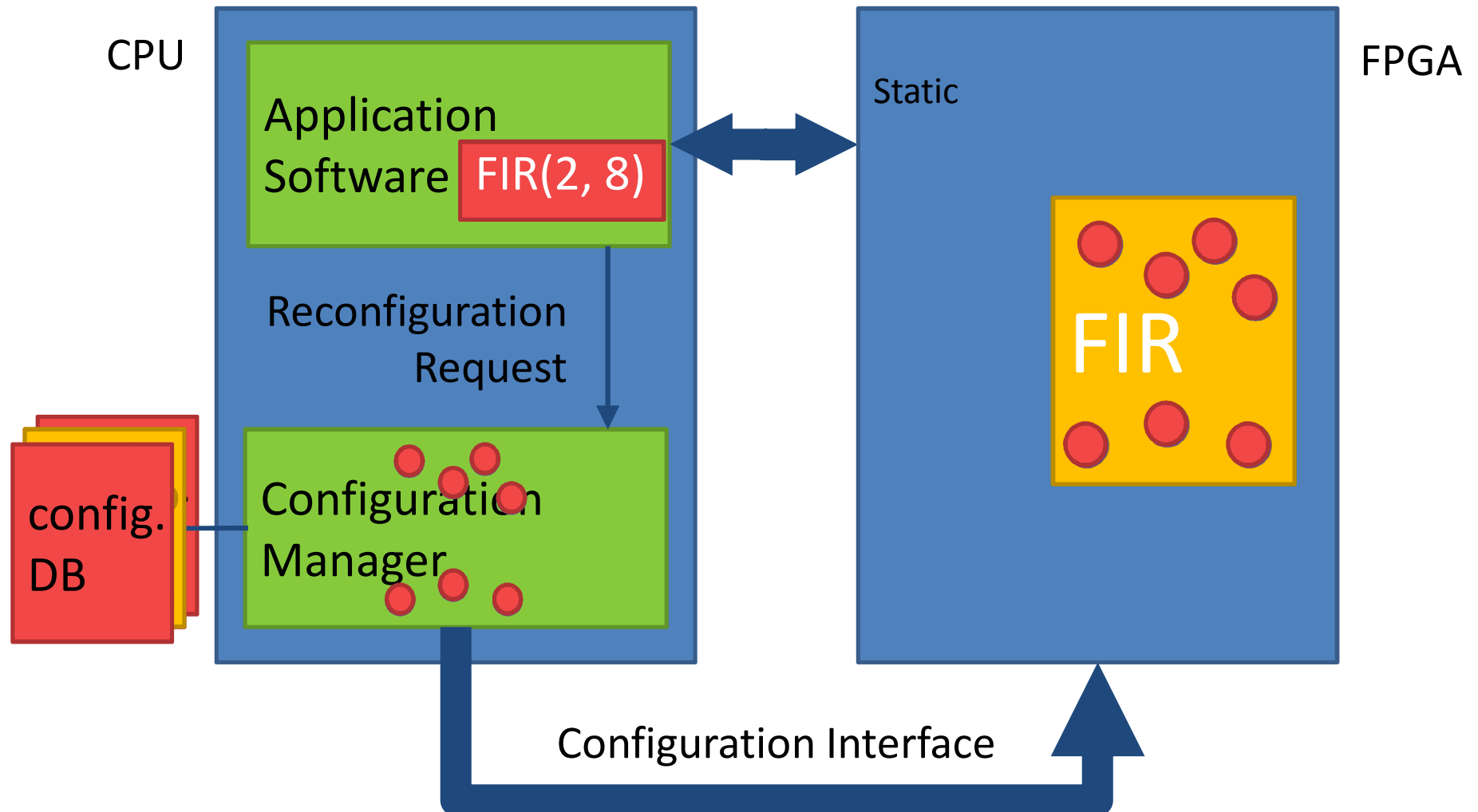


Our solution: Parameterized Configuration



* K. Bruneel and D. Stroobandt, "Automatic Generation of Run-time Parameterizable Configurations," FPL 2008.

Dynamic Circuit Specialization (micro-reconfiguration)



Outline

- What is Parameterized Run-time Reconfiguration?
- **Profiling** your applications for parameters
- Improvements in reconfiguration procedure
- Routing for parameterized reconfiguration
- Addressing conventional modular reconfiguration

Dynamic Circuit Specialization (DCS)

Identifying applications that might benefit from DCS is hard for the designer:

- Know the application very well
(What are the infrequently changing signals?)
- Be very familiar with Circuit Specialization
(What is the impact of choosing these parameters?)

 Requires a lot of low level work

In general, DCS results are **hard to predict** without actually making the DCS implementation

Dynamic Circuit Specialization (DCS)

Solution: Methodology for identifying DCS opportunities.

How?

- By comparing **all** DCS implementations of the same application

Two problems:

- How to compare different DCS implementations?
- Too many possible implementations (one for every possible set of parameters)

How to compare implementations?

Use the Functional Density as a measure for implementation efficiency.

$$FD = \frac{N}{T \cdot A}$$

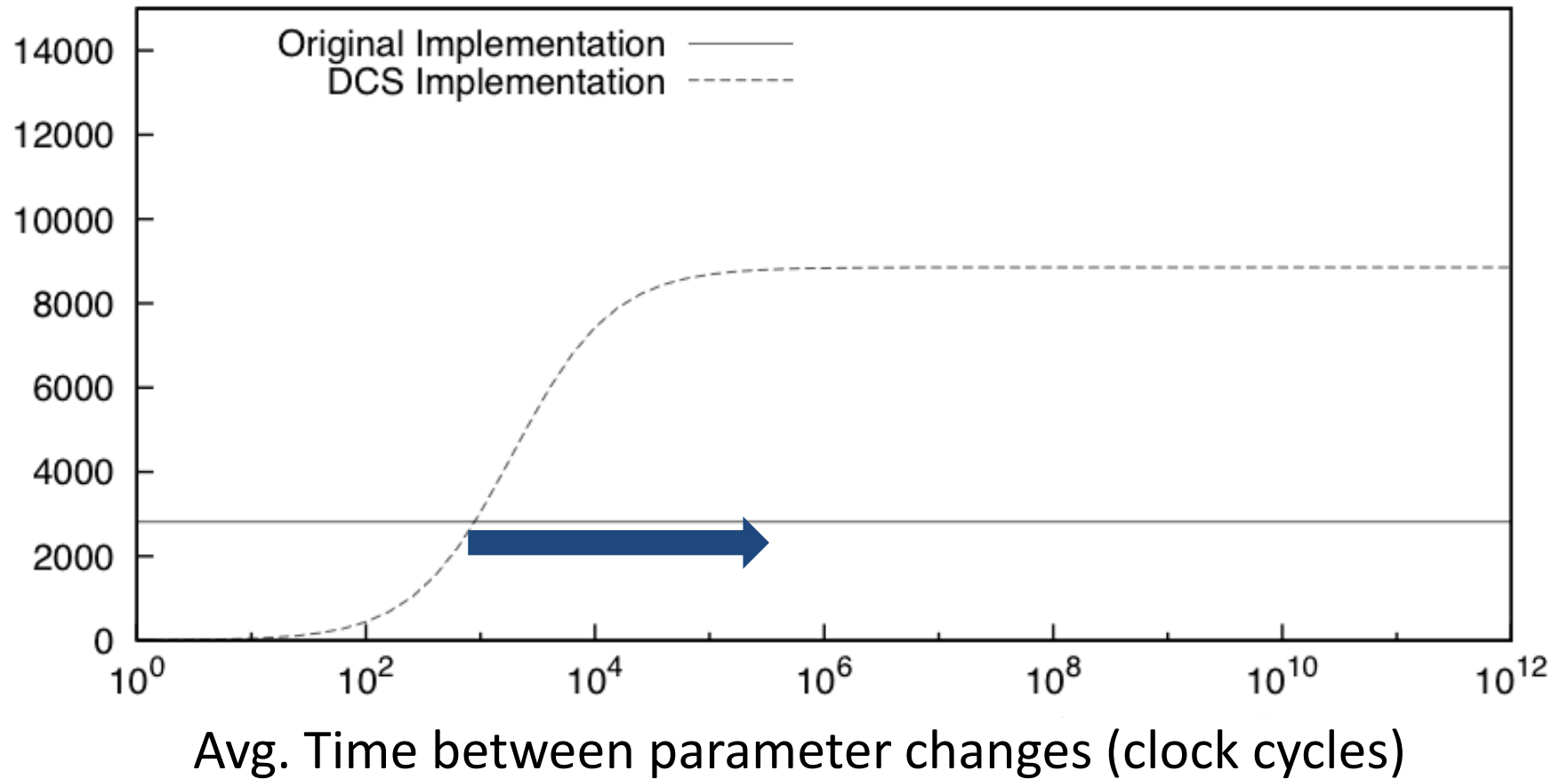
A: The area needed

T: The total execution time

N: The number of operations

*A. M. Dehon, Reconfigurable architectures for general- purpose computing, Massachusetts Institute of Technology, 1996.

Parameter Selection



Profiling the RTL

The DCS-RTL profiler, in three steps:

1. List parameter candidates and their dynamic behavior.
(Using a test bench with real-life data)
2. Reduce the number of parameter candidates
3. Calculate the functional density for each remaining parameter candidate

Execution Time

3 RTL-DCS Profiler implementations:

1. Exact FD calculation, no parameter pruning
2. Exact FD calculation, with parameter pruning
3. Estimated FD, without running Place and Route

Table I. Run times of all three RTL-DCS Profiler implementations

Design	Orig. Size (LUTs)	# cand.	after prun.	Total run time (h:m:s)			Run time Impr.
				Exact FD	Exact FD (prun.)	Est. FD	
16-tap FIR filter (8-bit)	2099	17	1	0:45:47	0:14:31	0:12:31	3.65x
32-tap FIR filter (8-bit)	4399	33	1	2:38:19	1:04:09	1:02:33	2.54x
16-tap FIR filter (16-bit)	8977	17	17	2:38:35	2:38:35	1:39:39	1.59x
32-tap FIR filter (16-bit)	17312	33	32	10:34:20	10:20:45	7:19:50	1.41x
RC6 encryption	2772	2	1	0:18:25	0:16:26	0:13:47	1.33x
RC6 decryption	3017	2	1	0:19:26	0:17:21	0:14:42	1.32x
Twofish 128	5491	48	14	2:51:58	0:56:41	0:22:19	7.70x
Twofish 192	6891	63	19	4:22:07	1:20:35	0:32:38	7.99x
Twofish 256	8270	78	22	5:56:24	1:40:47	0:44:34	8.03x
Pipelined AES	12958	15	5	2:18:58	0:57:20	0:29:23	5.19x

Test machine: Intel Core i7-3770 3.40GHz CPU with 32 GB of RAM

Target: Virtex-5 FPGA (XC5VFX70T-1FF1136)

Profiling Quality

Would DCS be beneficial?

Design	Calc. FD	Calc. FD w. Prun.	Est. FD
16-tap FIR (8 bit)	Yes	Yes	Yes
32-tap FIR (8 bit)	Yes*	Yes*	Yes*
16-tap FIR (16 bit)	Yes*	Yes*	Yes*
32-tap FIR (16 bit)	Yes*	Yes*	Yes*
RC6 encry.	Yes	Yes	Yes
RC6 decry.	Yes	Yes	Yes
Twofish 128	No	No	No
Twofish 192	No	No	No
Twofish 256	No	No	No
Pipelined AES	No	No	No

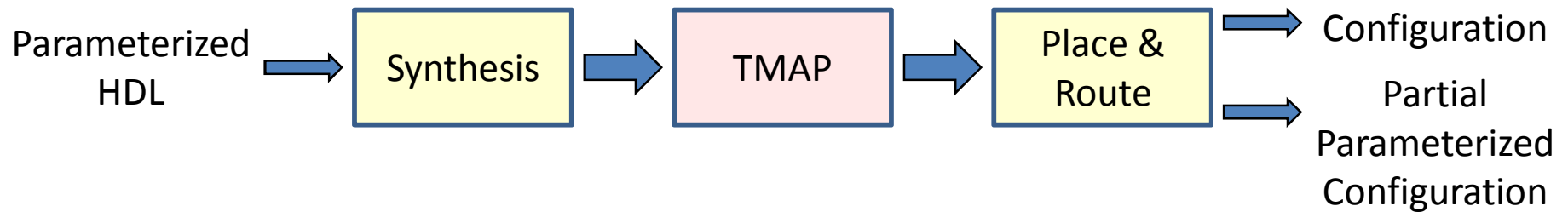
*SRL-reconfiguration is beneficial, HWICAP not

Outline

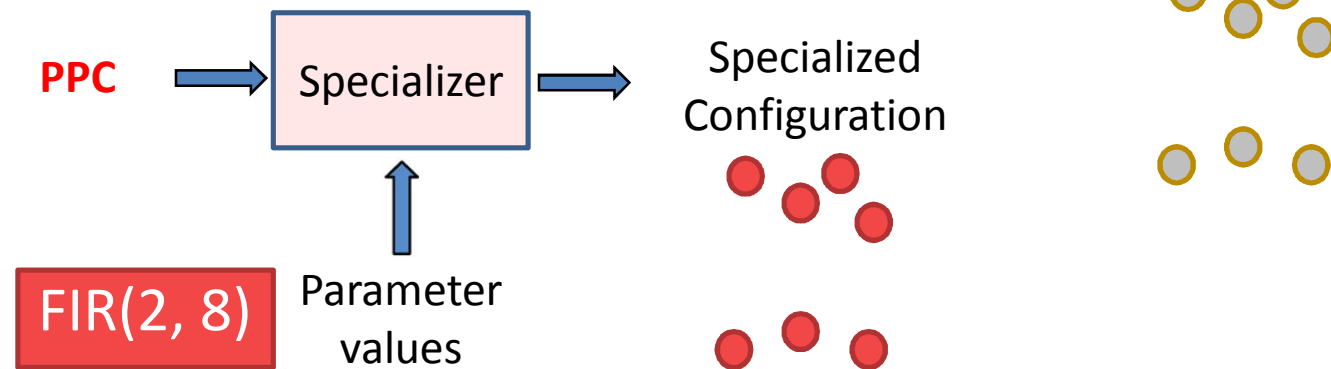
- What is Parameterized Run-time Reconfiguration?
- Profiling your applications for parameters
- **Improvements in reconfiguration procedure**
- Routing for parameterized reconfiguration
- Addressing conventional modular reconfiguration

Reconfiguration Mapping

- Generic stage (compile-time)



- Specialization stage (run-time)



* K. Bruneel and D. Stroobandt, "Dynamic Data Folding with Parameterizable FPGA Configurations," Todaes 2011.

Contribution

- Reduce the specialization overhead
 - Area overhead (resources of specializer / memory to store PPC)
 - Time overhead (specialization time)
 - On High Hierarchical Level (HHL)
 - By exploiting the regularity in applications
 - On Low Hierarchical Level (LHL)
 - Through efficient designing for the specialization process

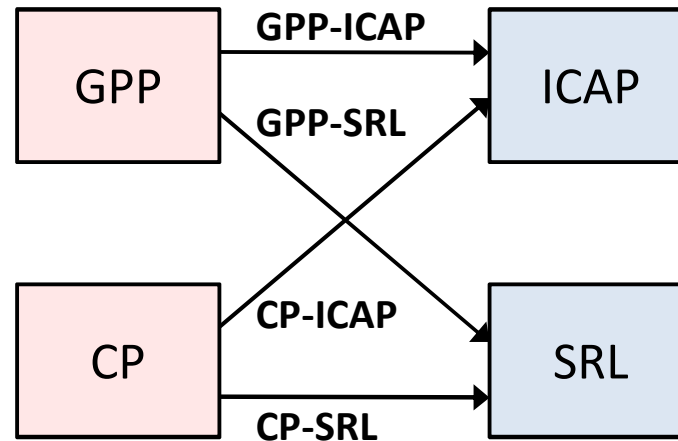
For 16-taps FIR on Virtex-II Pro

	PPC Memory (Kbyte)	Reduction factor
Initially	375.07	1
HHL	10.45	35
LHL	2.80	133

Evaluation of the Parameterized Bit Stream

- Evaluation in GPP
 - Embedded processors such as PowerPC, MicroBlaze, PicoBlaze, ...
 - Partial Parameterized Configuration compiled to C function on processor on parameter change
- Evaluation in Custom Processor
 - Main target: evaluate Boolean network (AIG) with limited number of operators
 - Designed CP based on a stack machine architecture with load and store instructions

Evaluation of the Parameterized Bit Stream



- Four different solutions are used
- The PowerPC is used as an example for a GPP

Results Specialization Overhead

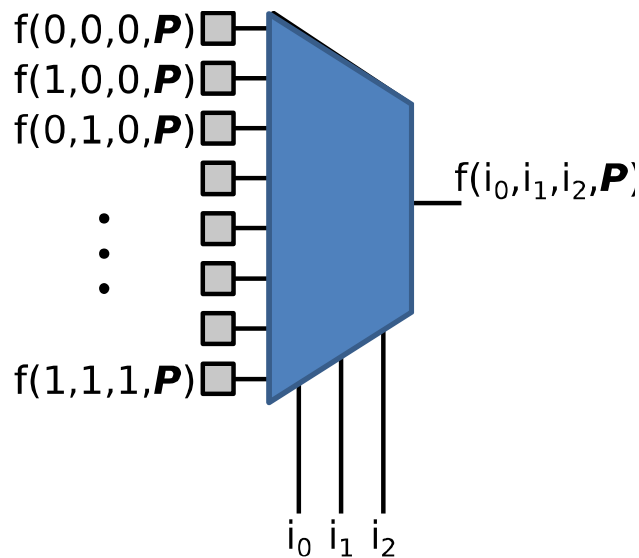
Solution	FIR			TCAM			RegEx		
	PPC Kbyte	CBS Kbyte	S. time msec	PPC Kbyte	CBS Kbyte	S. time msec	PPC Kbyte	CBS Kbyte	S. time msec
GPP-ICAP	10.45	32	52.59	2.10	13.1	20.320	2.95	8.3	2.456
GPP-SRL	10.45	-	1.50	2.10	-	0.228	2.95	-	0.144
CP-ICAP	3.4	77	4.18	0.65	39.2	1.057	1.20	34.6	0.842
CP-SRL	2.8	-	2.24	0.54	-	0.216	1.00	-	0.107

- Memory requirements for PPC: CP is better
- Specialization time (evaluation + communication + manipulation + configuration): SRL is better
- Combination: CP-SRL is best solution

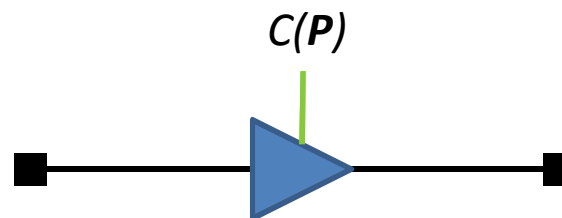
Outline

- What is Parameterized Run-time Reconfiguration?
- Profiling your applications for parameters
- Improvements in reconfiguration procedure
- **Routing** for parameterized reconfiguration
- Addressing conventional modular reconfiguration

The Generic Stage of the TCON Tool Flow



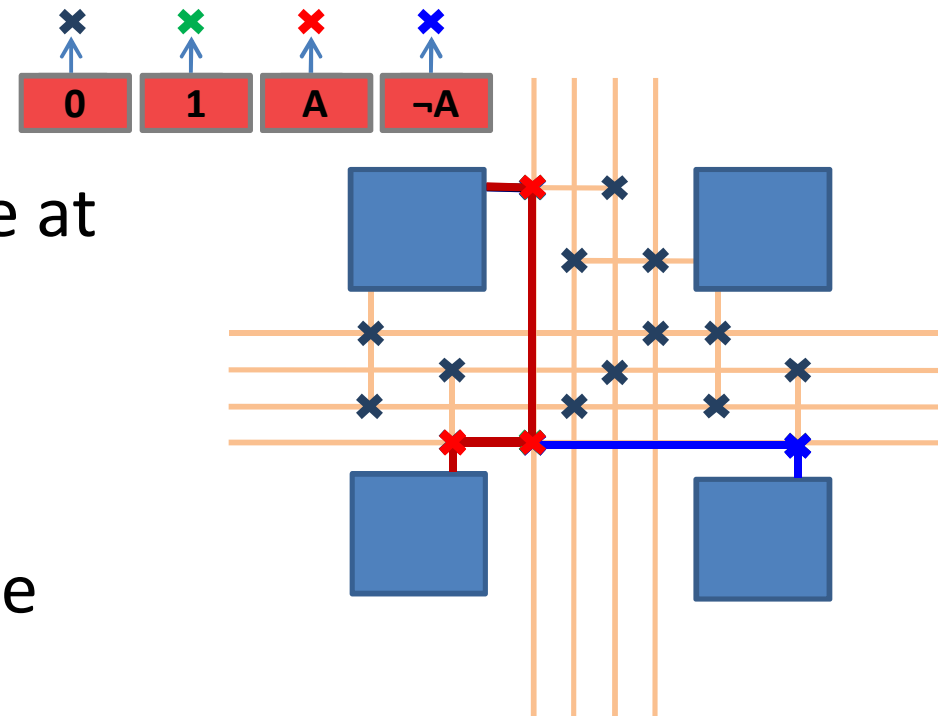
Tuneable LUT
TLUT



Tuneable Connection
TCON

Routing TCONs

- TCONs that are not active at the same time can share routing resources
- Exploit this property to minimize routing resource usage
- Place & Route algorithms

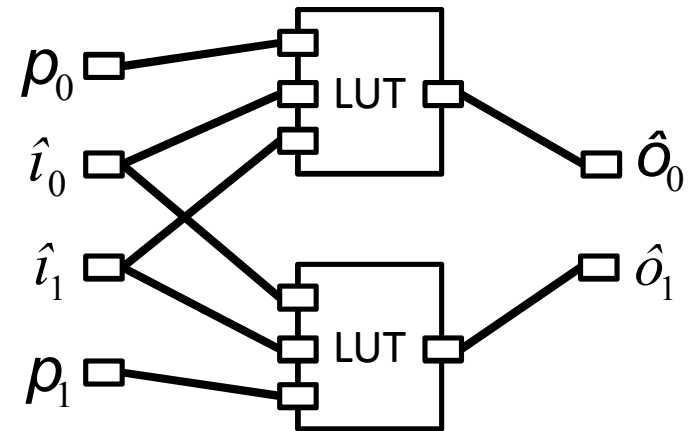


Dynamically specializing the interconnect network by reconfiguring the switches

2 x 2 Crossbar example: Summary

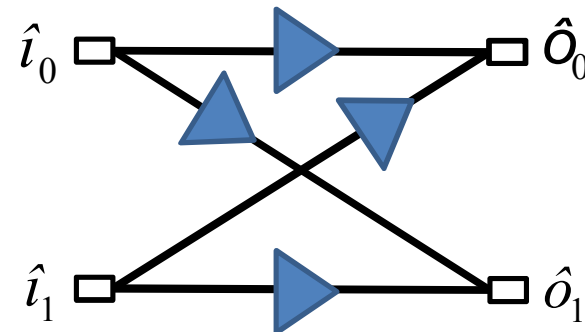
- Conventional implementation:

- 2 CLBs
- 6 IOBs
- 8 Connections



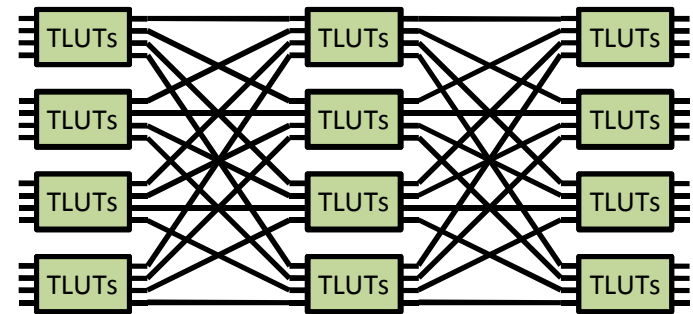
- TCON implementation:

- 0 CLBs
- 4 IOBs
- 4 Tuneable Connections

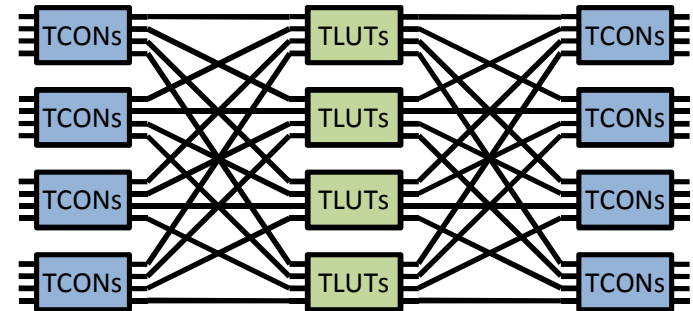


Clos Network Implementations

- Stages are built up of 4 x 4 Crossbars switches
 - Each with 8 control inputs
- 256 inputs, 256 outputs
- 3 types of implementation:
 - Conventional
 - TLUT
 - TCON (Combined)



TLUT implementation



TCON implementation

256 x 256 Clos Switch

	Conv.	TLUT	TCON
LUTs	6760	1792 (3.77x)	768 (8.80x)
Depth	12	7	3
Wires	97994	25353 (3.87x)	17853 (5.49x)
Min. Chan. Width	9	13	14
Routing Time [min]	257	20 (13x)	10 (25x)

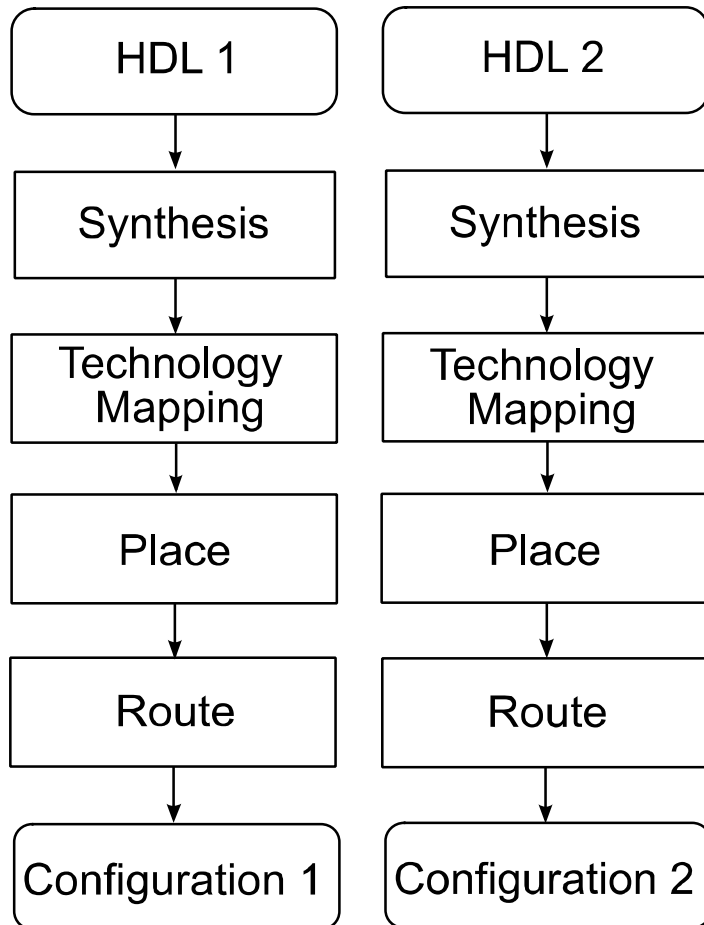
Outline

- What is Parameterized Run-time Reconfiguration?
- Profiling your applications for parameters
- Improvements in reconfiguration procedure
- Routing for parameterized reconfiguration
- Addressing conventional **modular reconfiguration**

Multi-mode circuit

- Several circuits, called **modes**, that are used mutually exclusive in time
- Example: software defined radio
- This work is an extension of the micro-reconfiguration technique :
 - not only circuits with infrequently changing input values
 - but also **multi-mode circuits**

Static vs parameterized bits



After implementation of multi-mode circuit with conventional Dynamic Partial Reconfiguration (DPR):

- Each memory cell corresponds to a collection of bit values, one for each mode
- This collection of bit values is called
 - a *static bit*, when the values are the same for all circuits
 - A *parameterized bit*, otherwise

Clustering of dynamic bits

- Only memory cells that contain a parameterized bit *need to* be rewritten during run-time
- Configuration memory of an FPGA is frame-based
- Conventional DPR flow:
 - Parameterized bits are *scattered* over the frames
 - Results in longer reconfiguration times
- Approach in this work:
 - Divide configuration frames into dynamic and static ones
 - *Cluster* parameterized bits into the dynamic frames

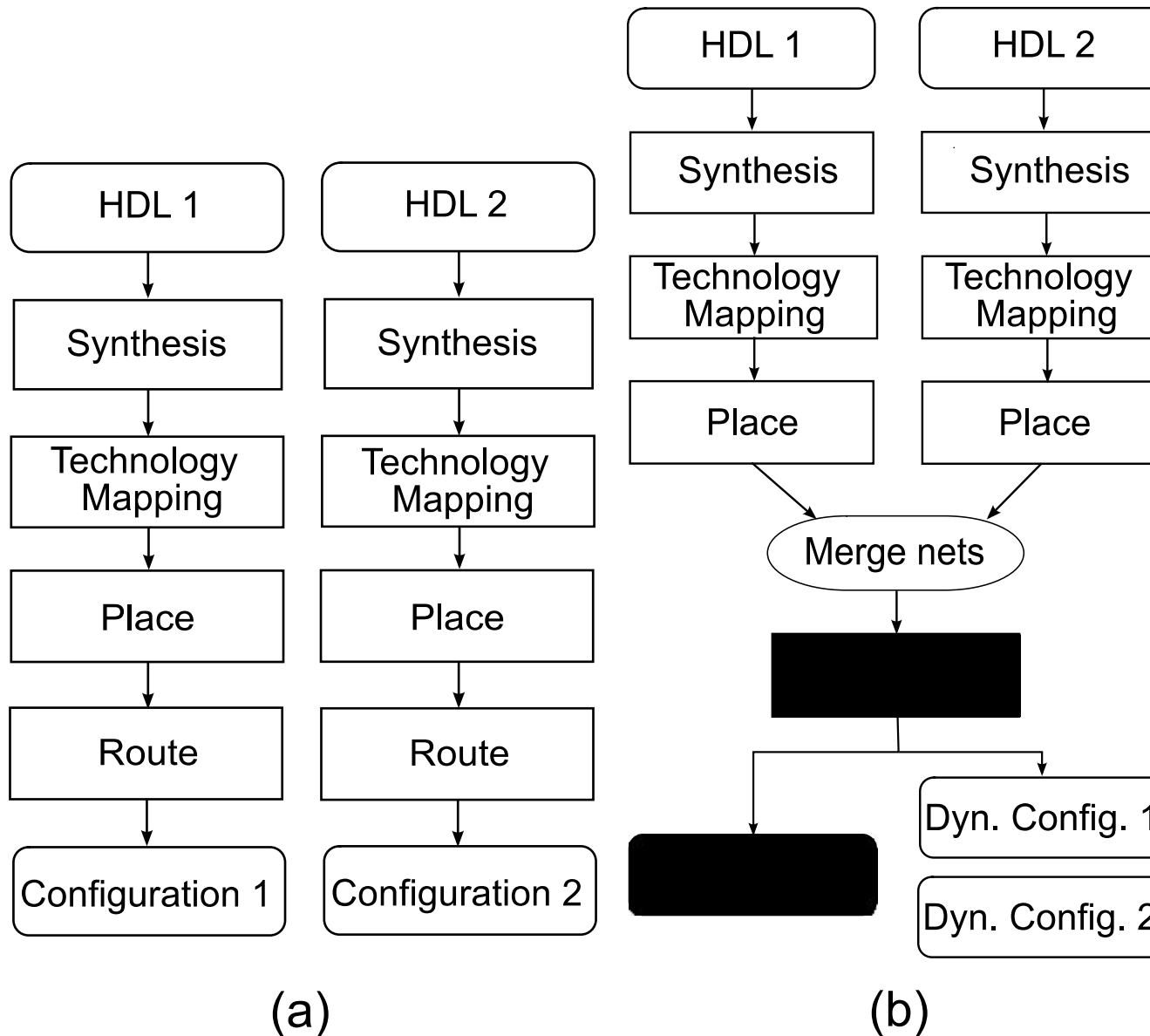
CLBs vs routing

- In our experiments:
 - 10% of the configuration memory consists of CLB bits
 - 90% of the configuration memory consists of routing bits

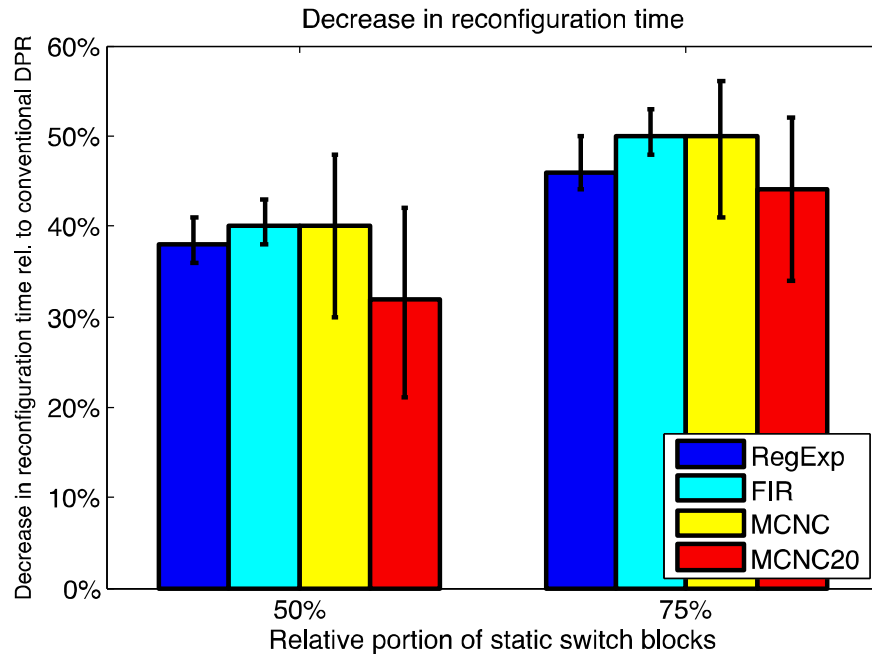
→ Most of the time spent in reconfiguring the routing infrastructure

- Focus on reducing reconfiguration time of routing
 - All CLB frames are dynamic
 - Routing frames are divided in static and dynamic ones
 - Novel router, called StaticRoute, that clusters parameterized routing bits in the dynamic routing frames

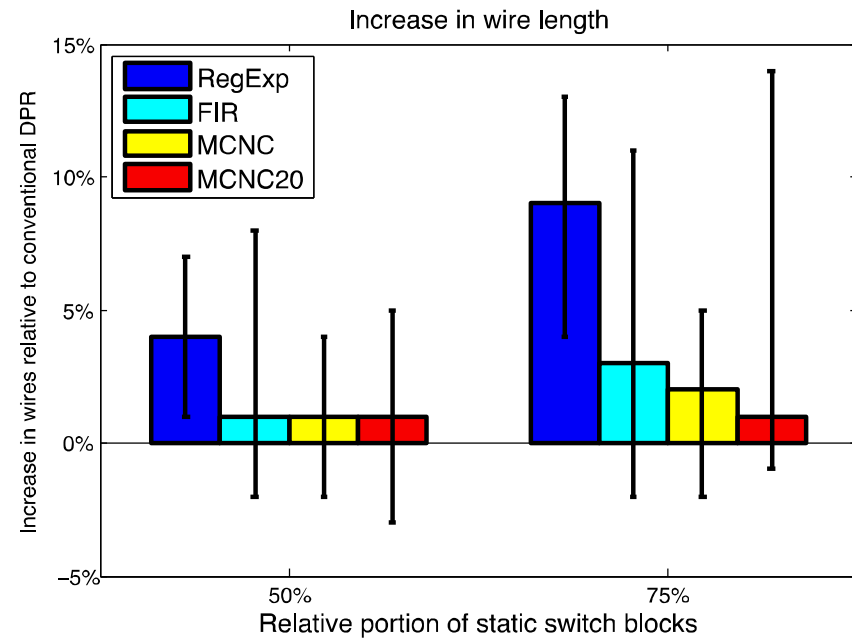
New DPR tool flow



Results



- Benefit: 40% to 50% decrease in reconfiguration time, dependent of size static part



- Cost: wire length increase of around 5% on average

Conclusions

- Parameter Pruning and FD estimate greatly reduce time required for DCS profiling, but still allow correct identification of DCS opportunities.
- We reduce the specialization overhead by
 - exploiting the **regularity** in applications on a HHL
 - efficient designing for the **specialization process** on a LHL
- The exploration of the specialization environment shows that the CP_SRL provides **efficient** implementations with **minimum overhead**
- TCON tool flow
 - Quickly generates specialized configurations with a specialized routing interconnect network
 - Saves area (LUTs and wires) on FPGA and reduces logic depth
- Work extended to multi-mode circuits with up to 50% reduction in reconfiguration overhead and a modest (5%) increase in wirelength

Last slide

- Much of this work was done in the framework of the EU-FP7 project FASTER (see next talk)
- Questions?
- More information: <http://hes.elis.ugent.be/>