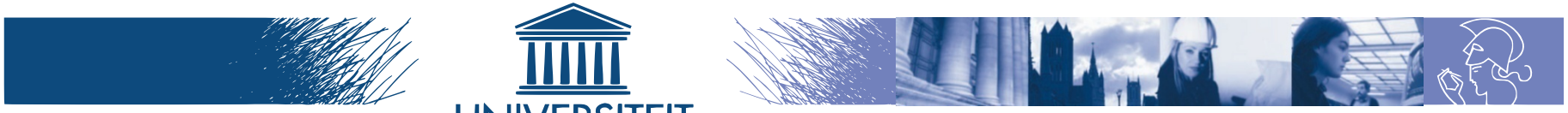


Will FPGA reconfiguration change the synthesis problem?

Prof. Dirk Stroobandt

Ghent University, Belgium

Hardware and Embedded Systems group

Outline

- What is Parameterized Run-time Reconfiguration?
- The importance of the parameter choice
- Effects on logic synthesis

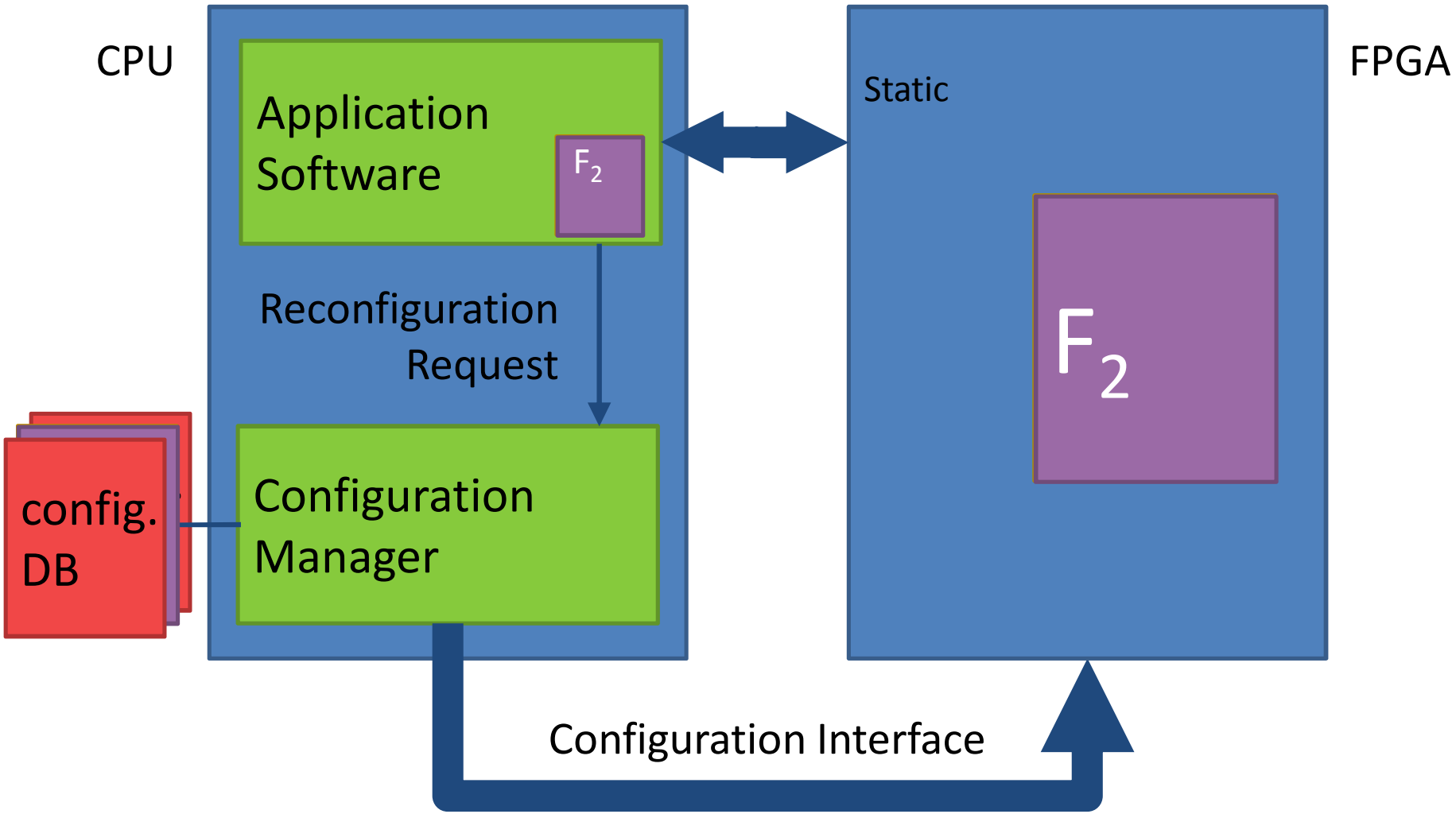
Outline

- What is Parameterized Run-time Reconfiguration?
- The importance of the parameter choice
- Effects on logic synthesis

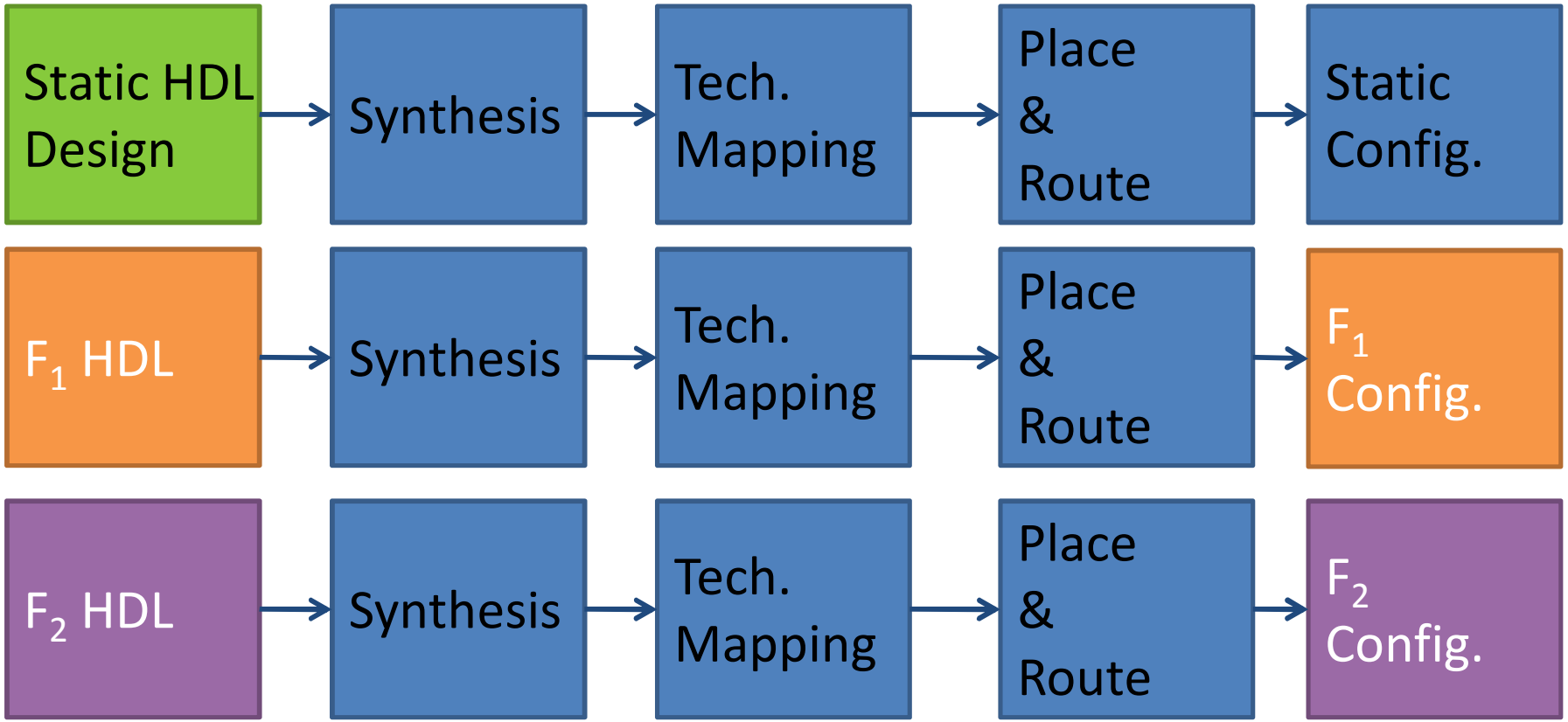
FPGA Run-Time Reconfiguration?

- 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



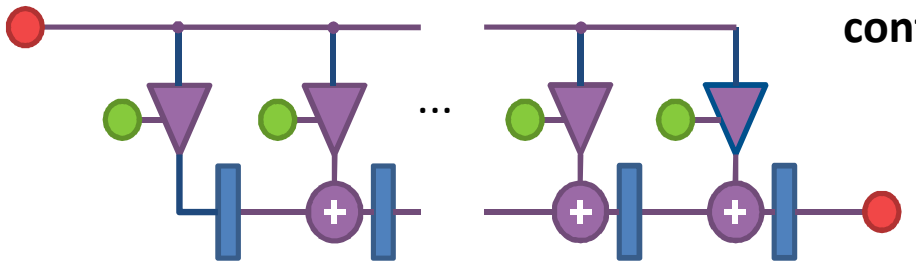
Conventional Tool Flow



Dynamic Circuit Specialization not feasible!

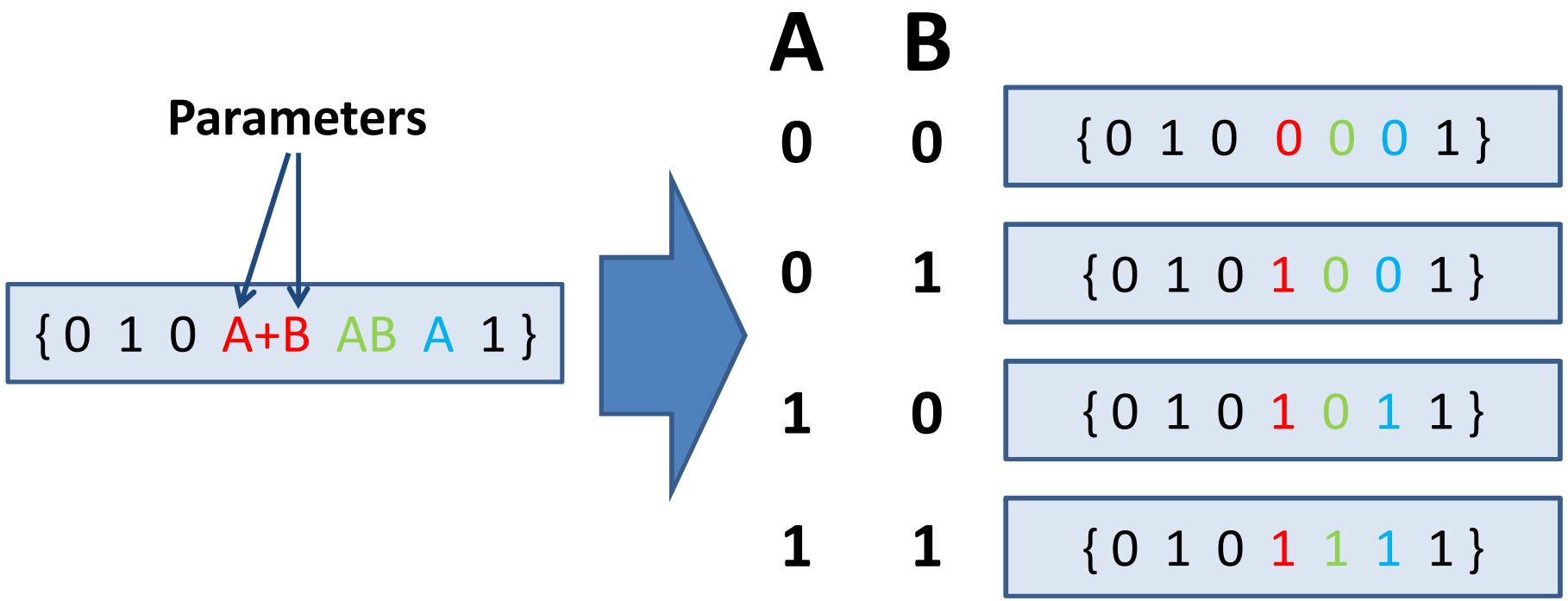
- Application where part of the input data changes infrequently
  - Conventional implementation (no reconfiguration):

    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)



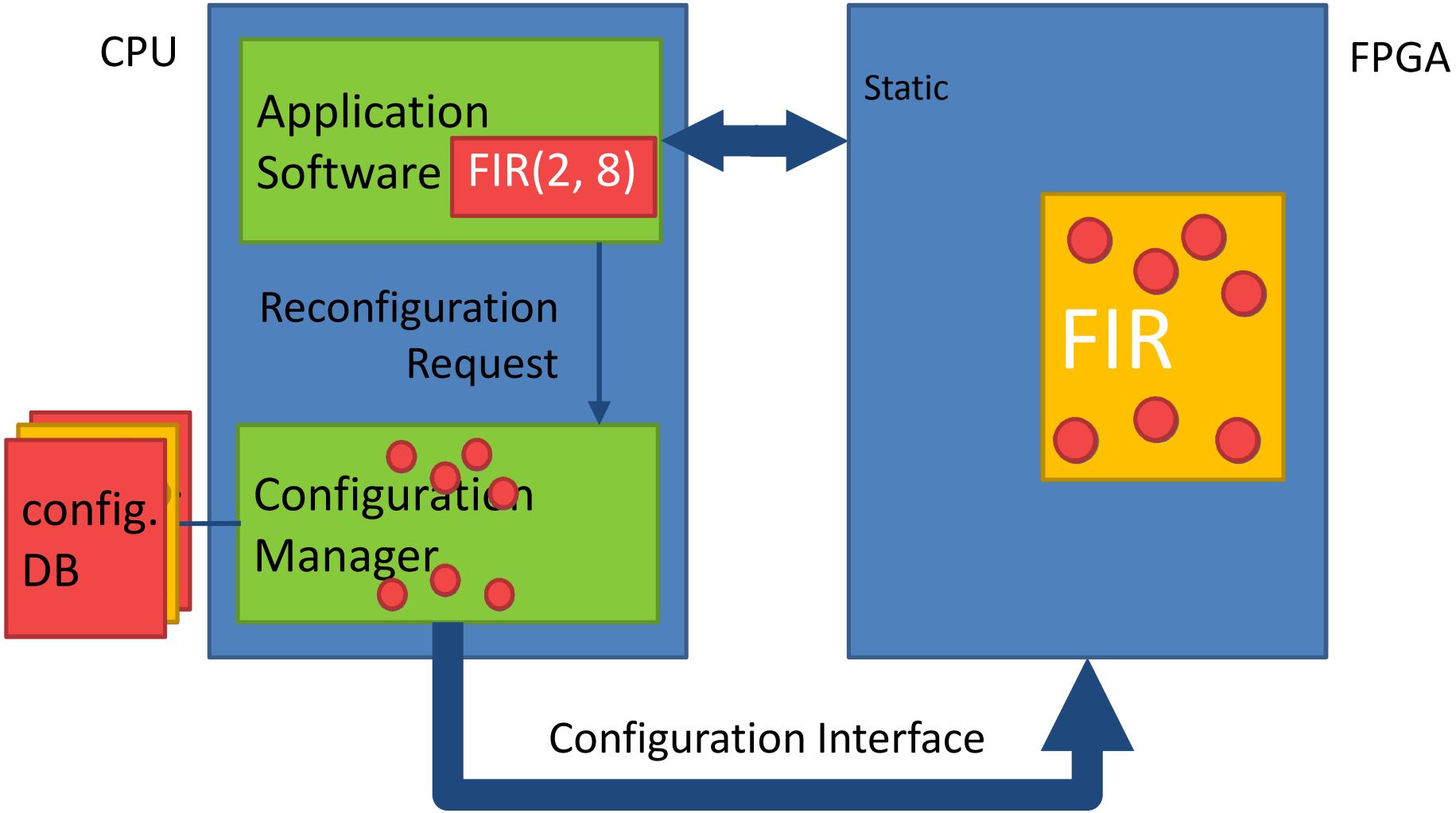
2^{128} possible configurations!

Our solution: Parameterized Configuration



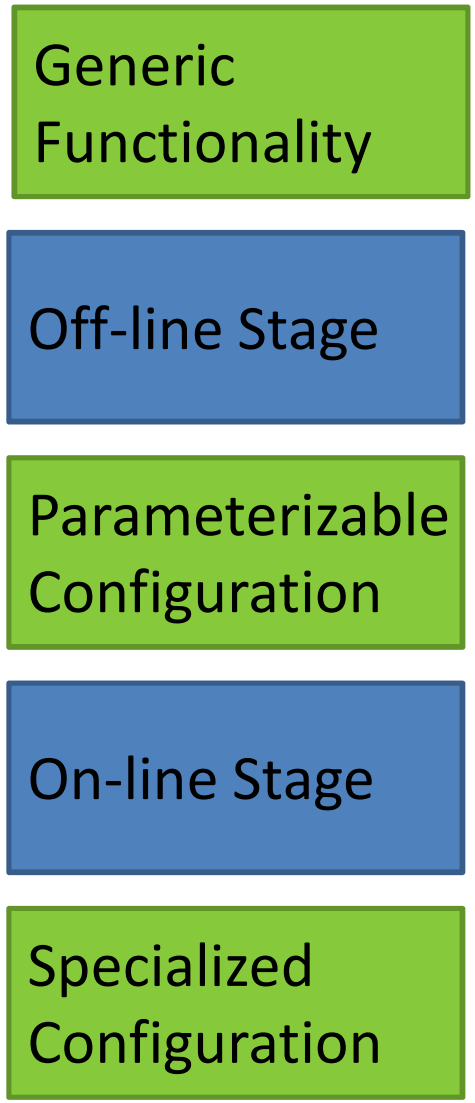
* K. Bruneel and D. Stroobandt, “Automatic Generation of Run-time Parameterizable Configurations,” FPL 2008.

Dynamic Circuit Specialization (micro-reconfiguration)

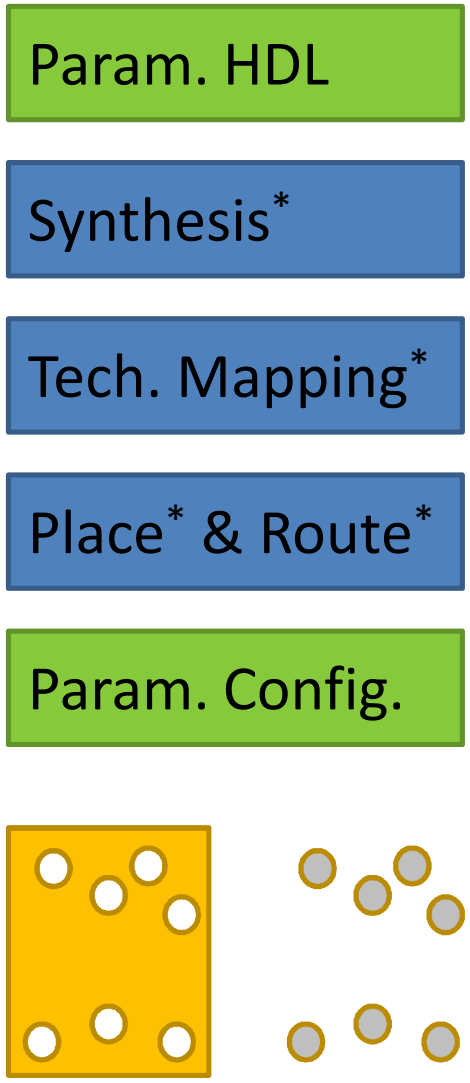


Two stage approach

- Off-line stage:
  - In: **Generic functionality**
    - Specification of the generic functionality
    - Distinction regular and parameter inputs
  - Out: **Parameterizable Configuration**
    - Software function
    - outputs specialized configurations for given parameter values
- On-line stage:
  - Evaluate parameterizable configuration
  - Out: **Specialized Configuration**
  - **Repeat** every time parameters change



Param. Configuration Tool Flow

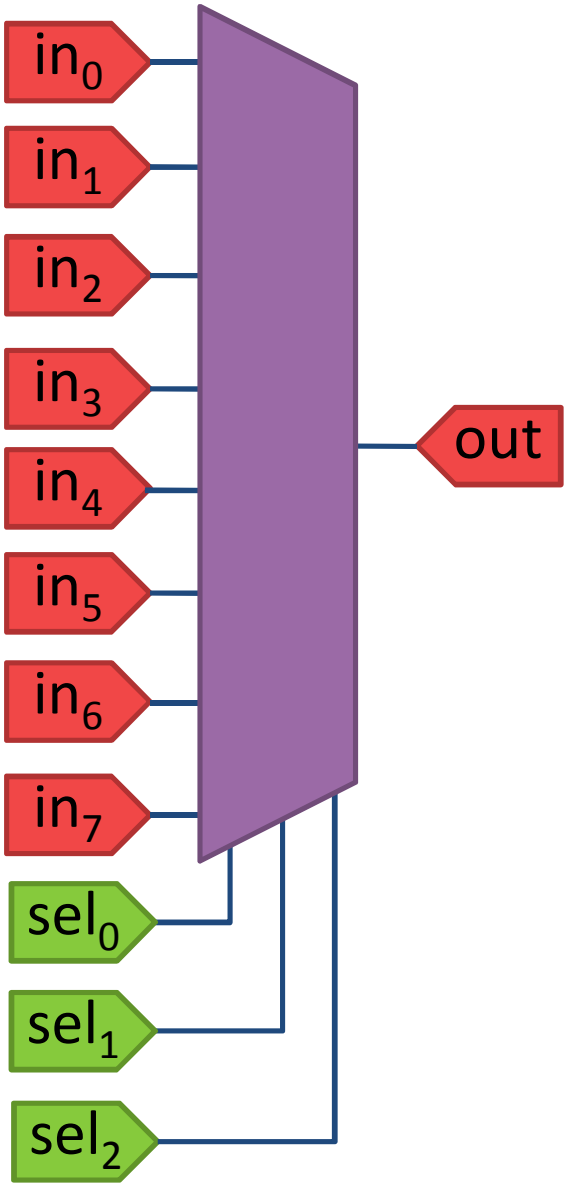


- **Tunable truth table** bits
  - Adapted Tech. Mapper: TMAP
  - Map to Tunable LUTs (TLUTs)
  - [FPL2008], [ReConFig2008], [DATE2009]
- Tunable routing bits
  - Adapted Tech. Mapper
  - Adapted Placer
  - Adapted Router

Outline

- What is Parameterized Run-time Reconfiguration?
- The importance of the parameter choice
- Effects on logic synthesis

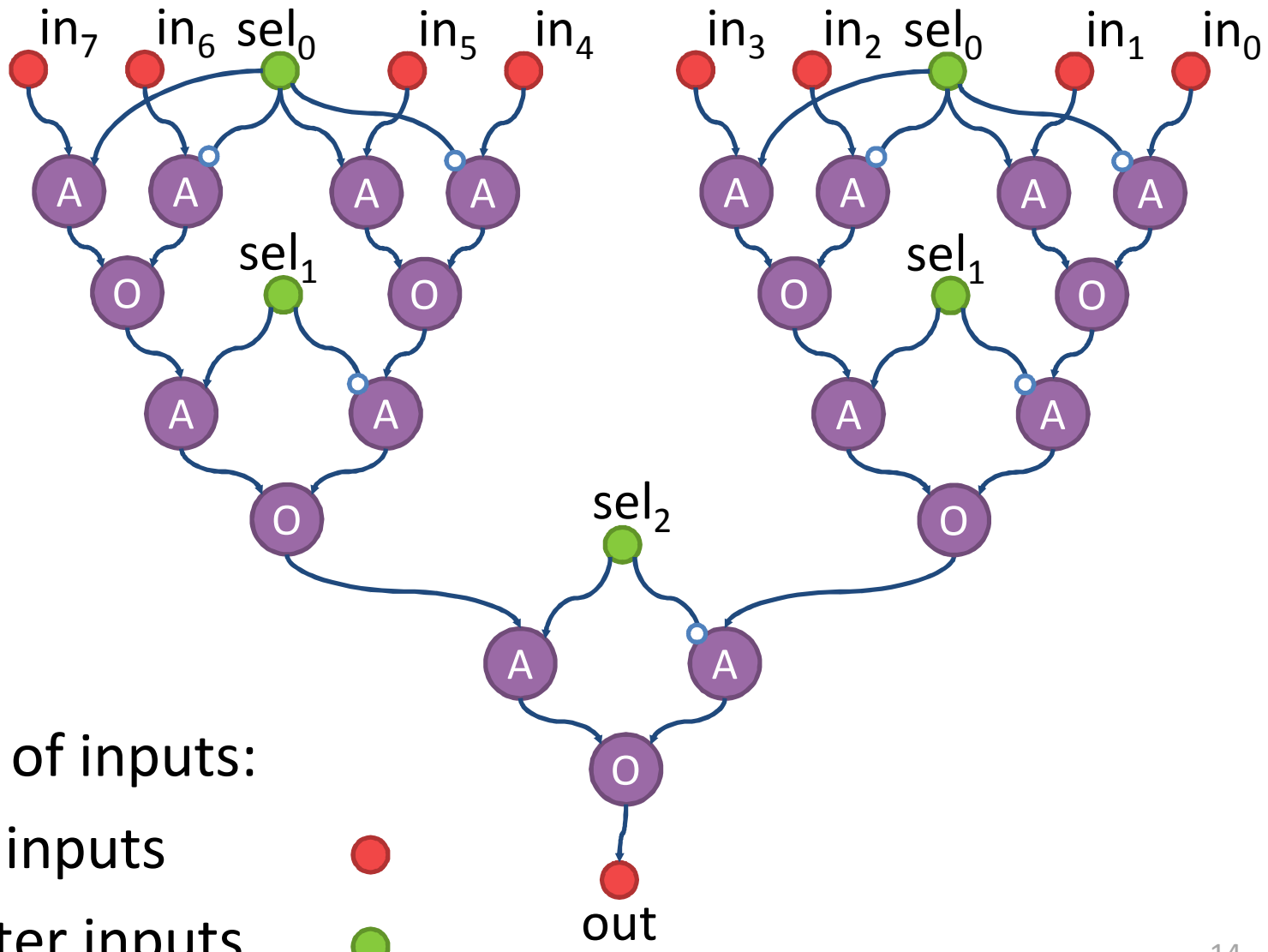
Parameterizable HDL design




```
entity multiplexer is
port(
  --BEGIN PARAM
  sel : in  std_logic_vector(2 downto 0);
  --END PARAM
  in  : in  std_logic_vector(7 downto 0);
  out : out std_logic
);
end multiplexer;

architecture behaviour of multiplexer is
begin
  out <= in(conv_integer(sel));
end behaviour;
```

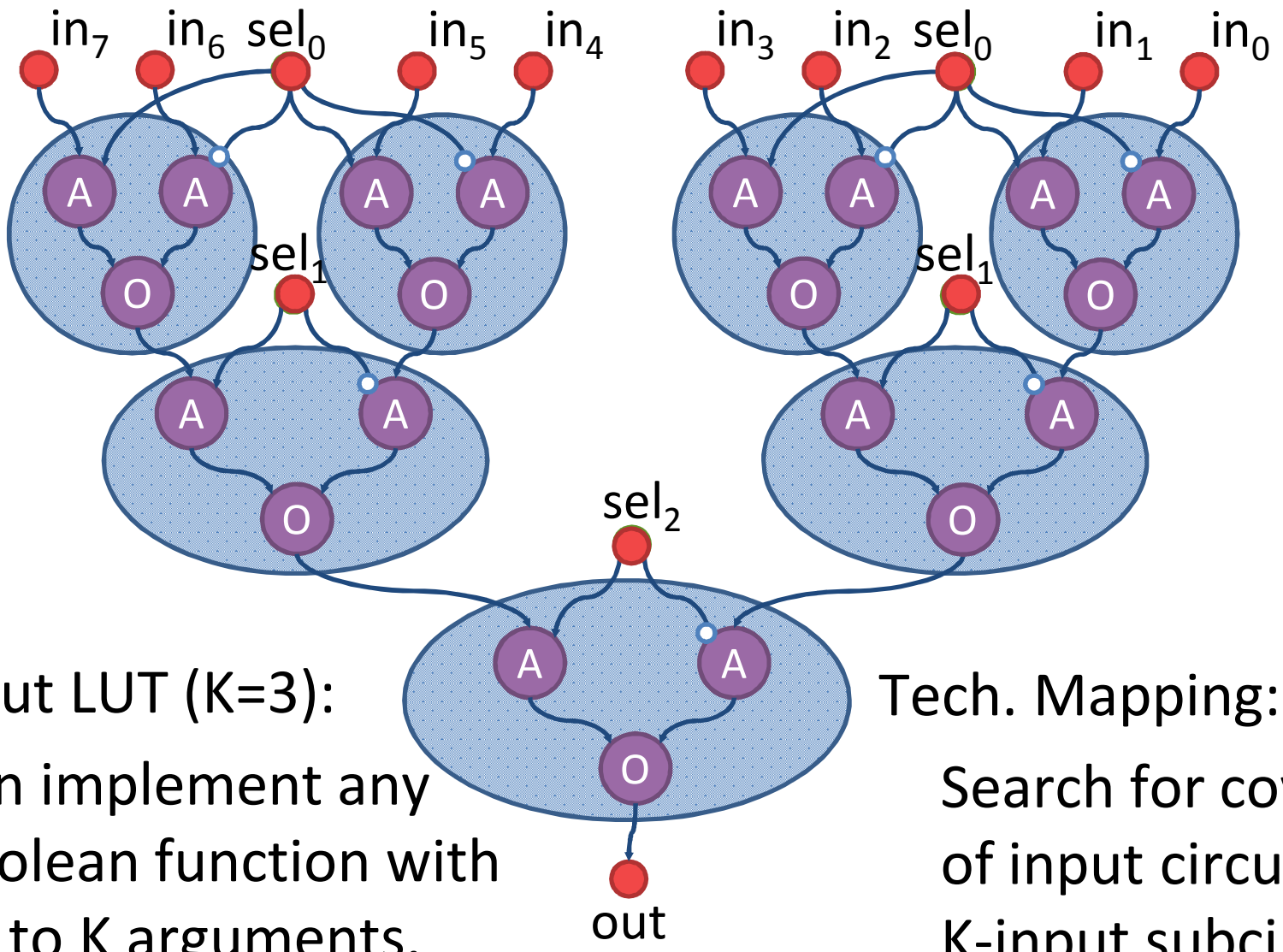
Synthesis*



Two types of inputs:

- Regular inputs
- Parameter inputs

Conventional technology mapping



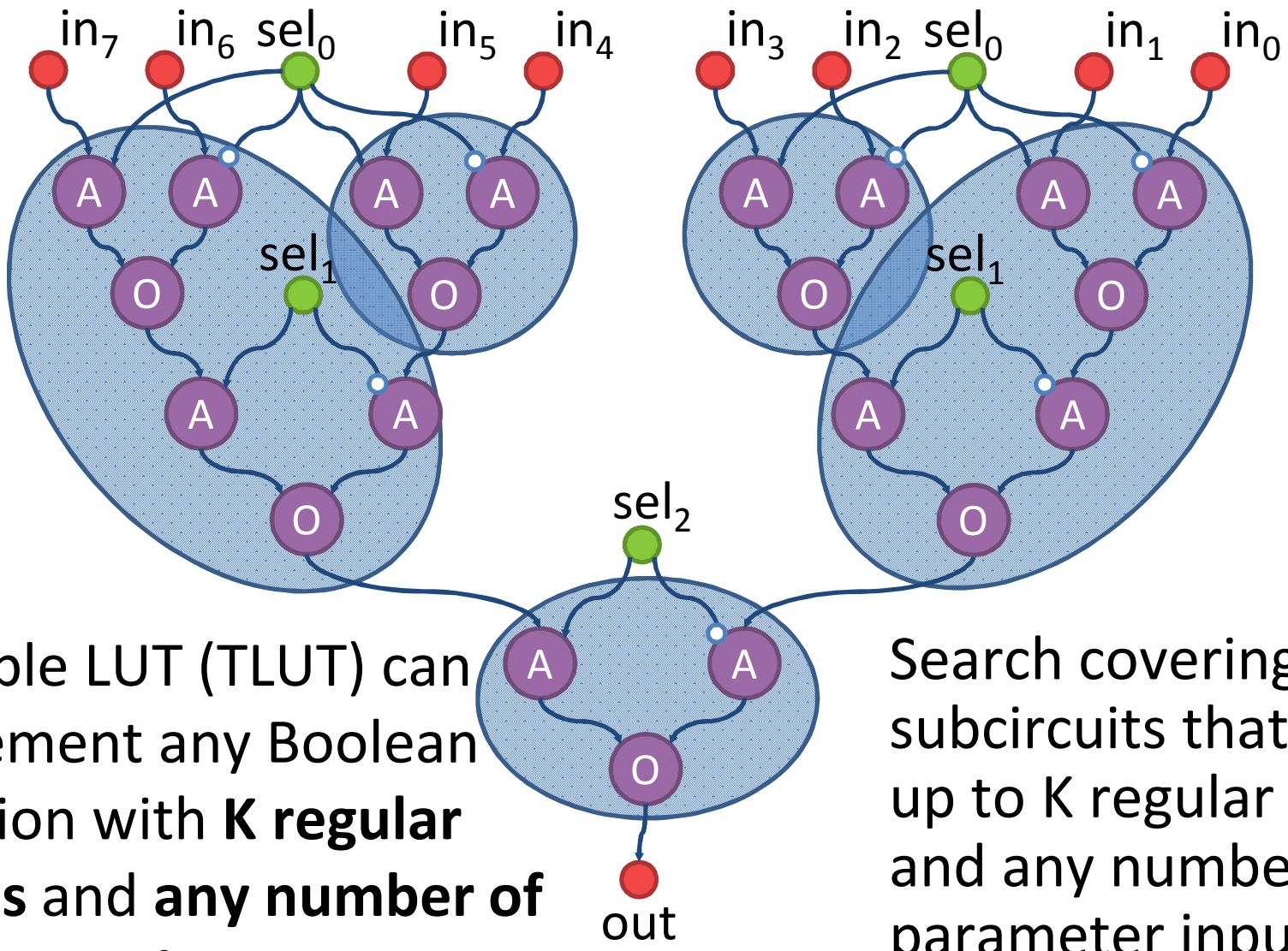
K-input LUT (K=3):

Can implement any Boolean function with up to K arguments.

Tech. Mapping:

Search for covering of input circuit with K-input subcircuits.

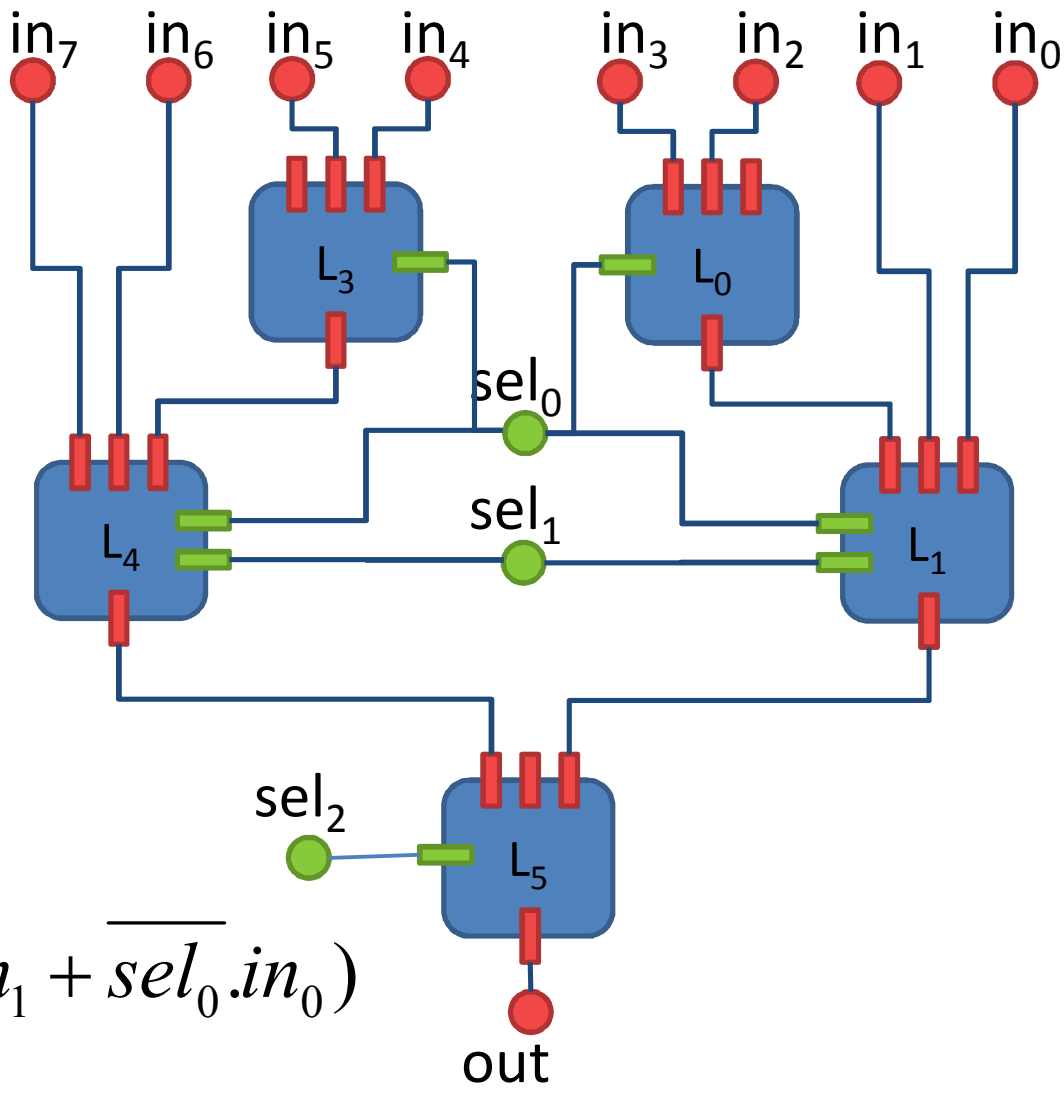
TMAP: Tunable LUT mapping



Tunable LUT (TLUT) can implement any Boolean function with **K regular inputs** and **any number of parameter inputs.**

Search covering with subcircuits that have up to K regular inputs and any number of parameter inputs.

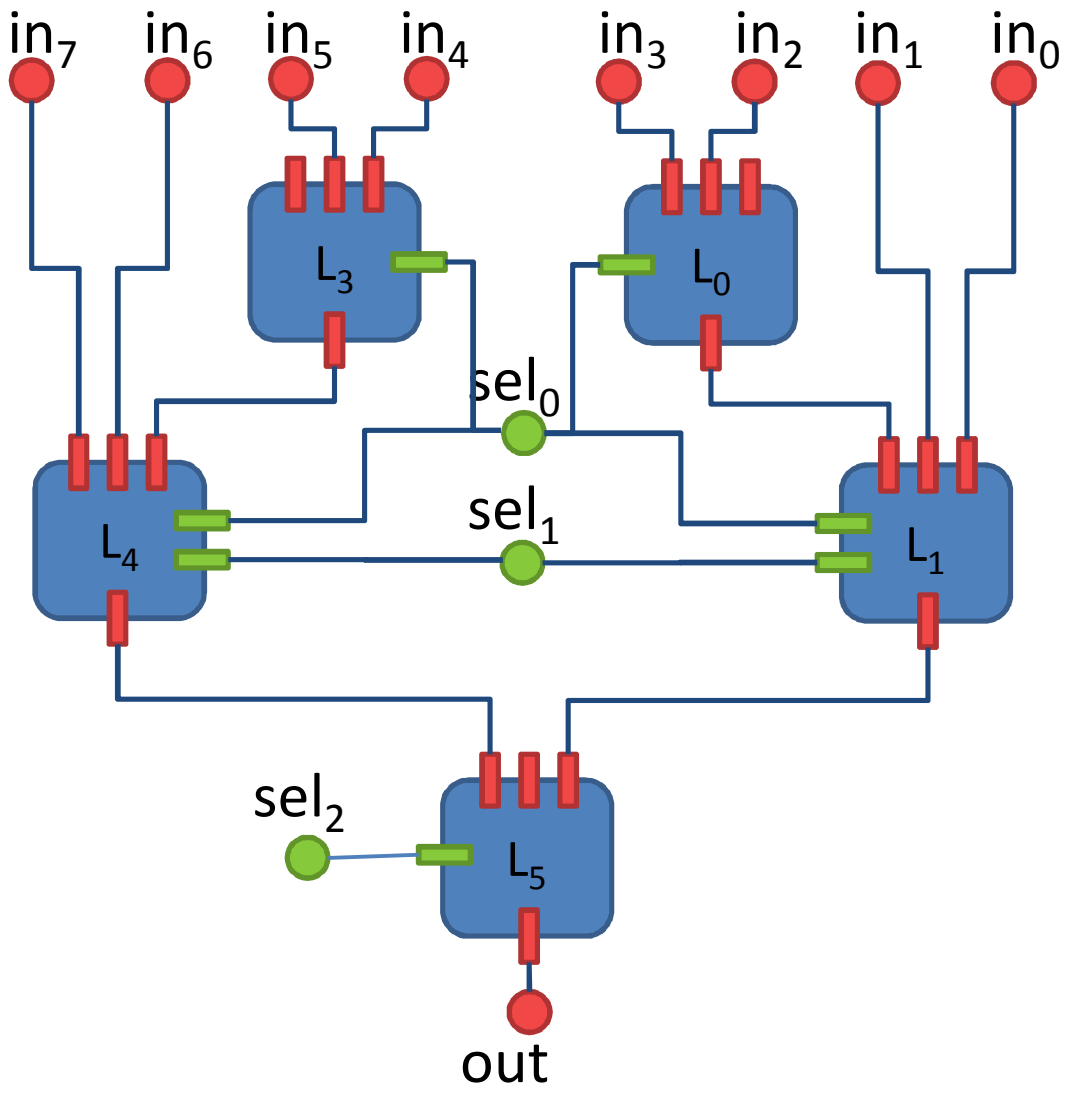
LUT structure and functionality



$$L_0 = sel_0.in_3 + \overline{sel_0}.in_2$$

$$L_1 = sel_1.L_0 + \overline{sel_1}.(sel_0.in_1 + \overline{sel_0}.in_0)$$

Place and Route



Experiment: 16-tap FIR, 8-bit coefficients

	Generic	Parameterizable configuration	Specialized
area (LUTs)	2999	**1301 (-56%)**	1146
clock freq. (MHz)	84	**115 (+37%)**	119
gen. time (ms)	0	**0.166**	35634
memory (kB)	0	**29**	2^{128} conf.

[illegible] **(5 orders)**

– [illegible]

– [illegible]

When should we use parameterized reonfiguration?

Use the Functional Density as a measure for implementation efficiency.

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

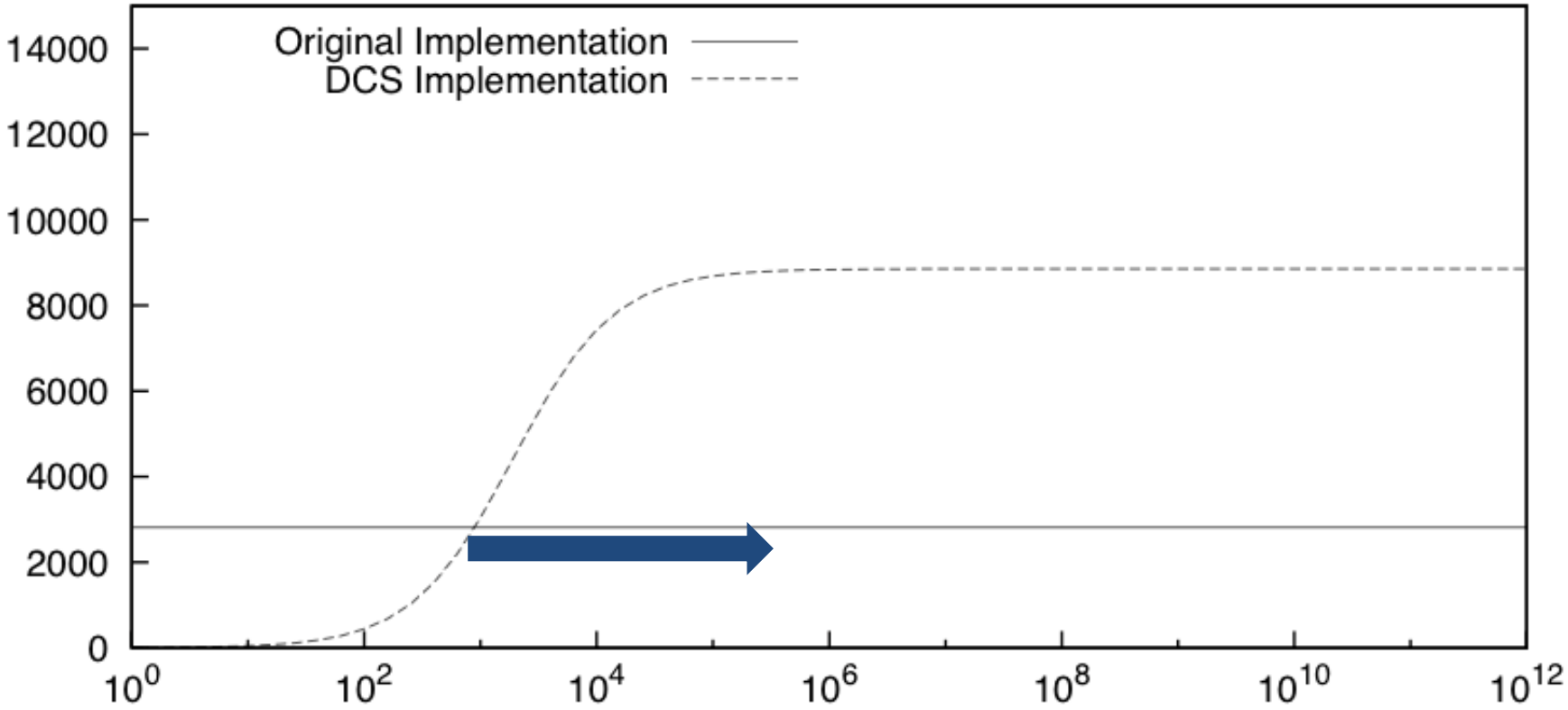
A: The area needed

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



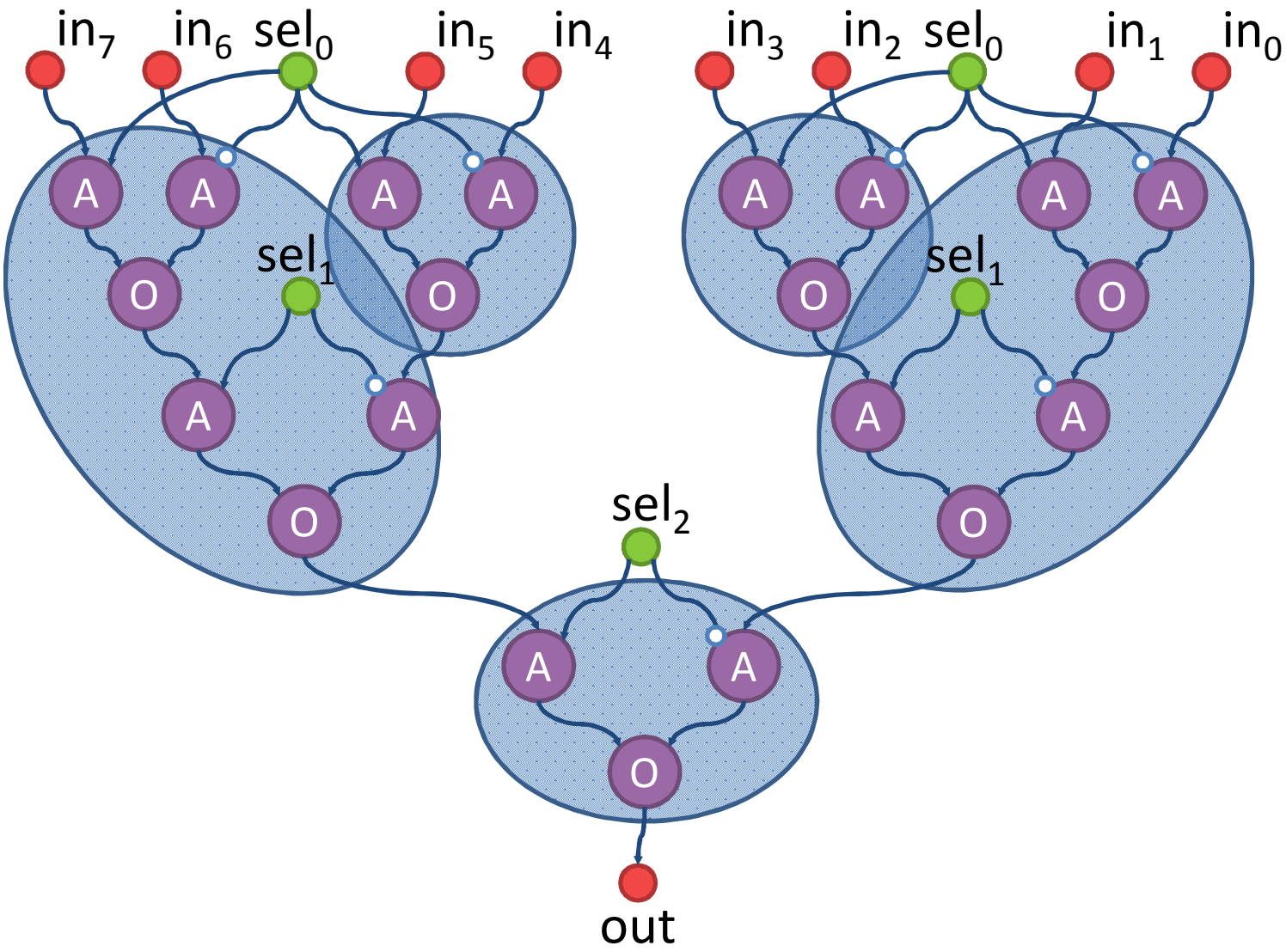
Avg. Time between parameter changes (clock cycles)

Profiler to trade off gain versus overhead of reconfiguration

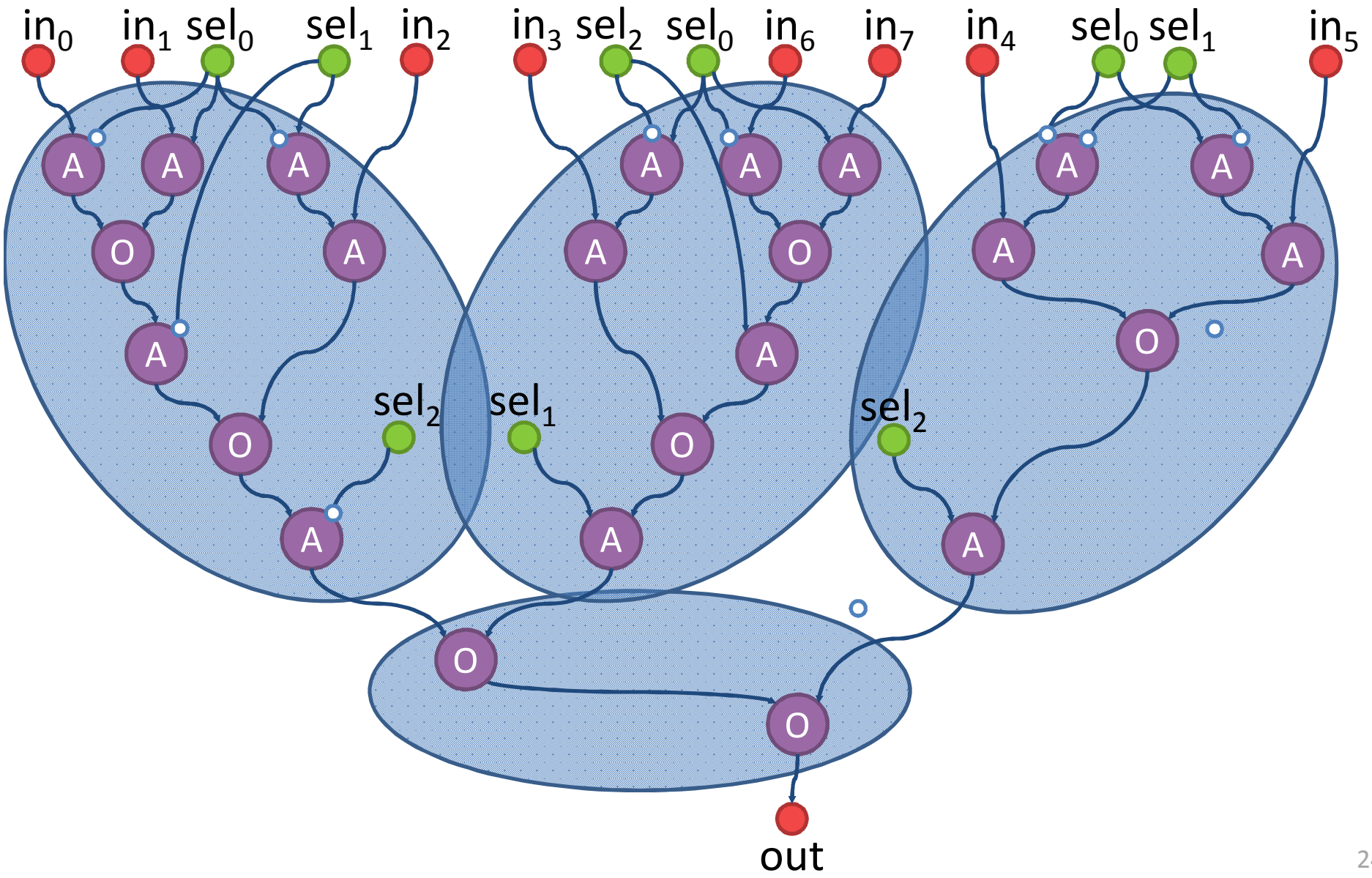
Outline

- What is Parameterized Run-time Reconfiguration?
- The importance of the parameter choice
- Effects on logic synthesis

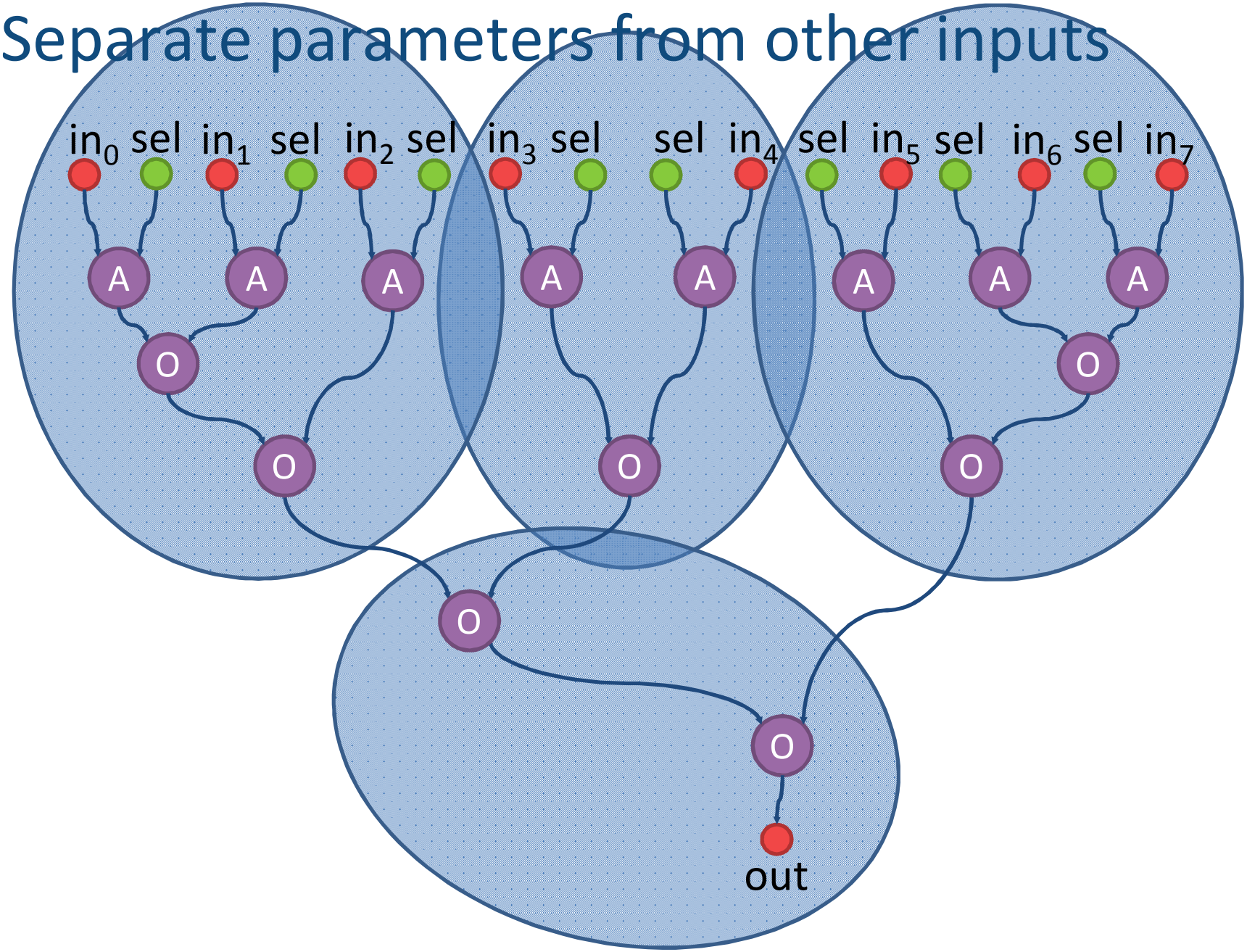
Original logic synthesis solution (3-input LUT)



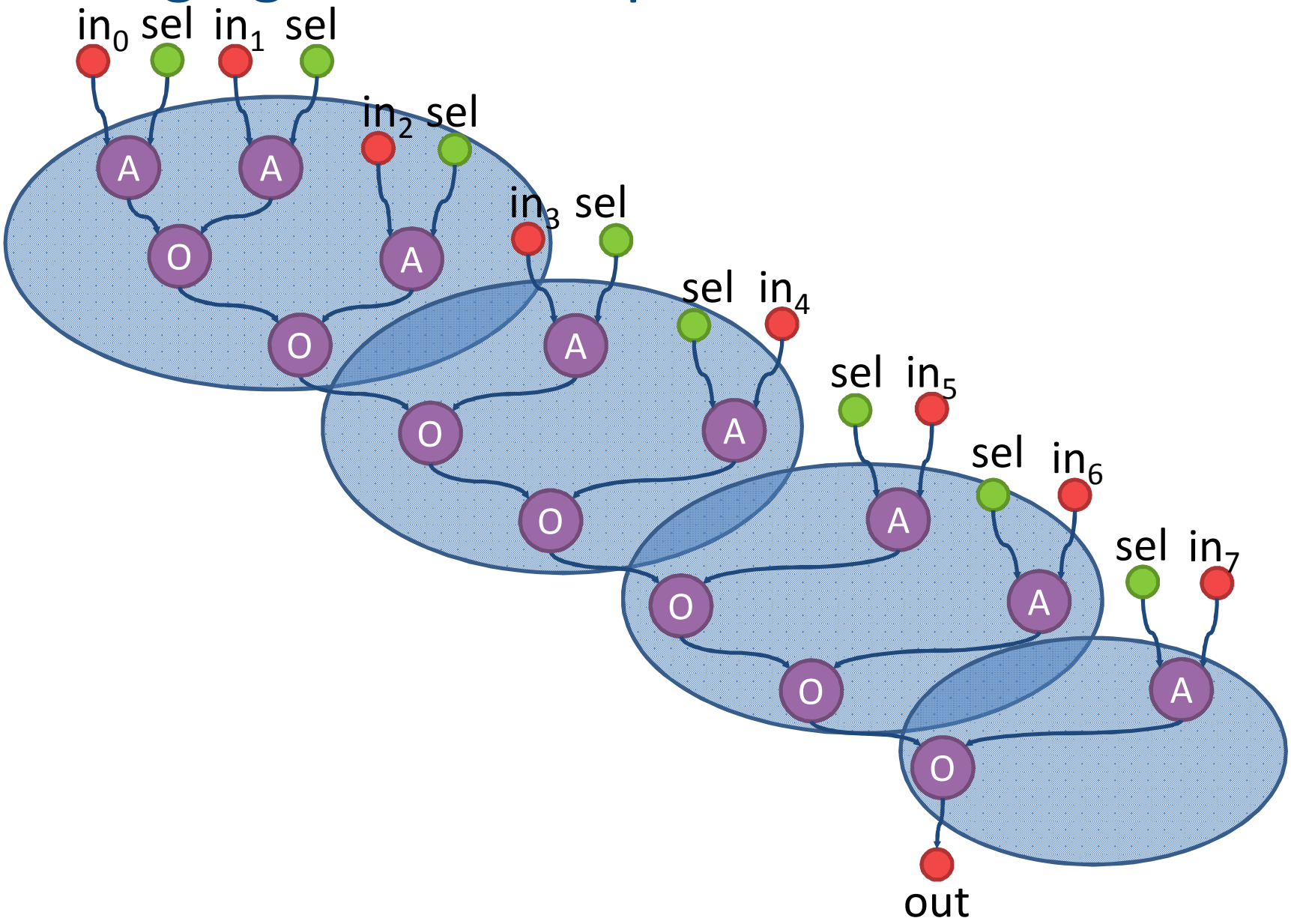
Making subtrees according to K regular inputs



Separate parameters from other inputs



Changing the tree depth



Conclusions

- Parameterized reconfiguration opens up new optimization possibilities using run-time reconfiguration
- Parameters are to be treated differently in Technology Mapping
- Therefore parameters and regular inputs should be treated differently in logic synthesis
- Cost of parameter calculations (Boolean functions) should also be taken into account
- New challenge in synthesis

Submit to IWLS

Paper abstract sumission: March 11, 2016

www.iwls.org

Last slide

- Much of this work was done in the framework of the EU-FP7 project FASTER and is now continued in the EU-H2020 project (FETHPC) EXTRA

- Tools at https://github.com/UGent-HES/tlut_flow

- Questions?

- More information: http://hes.elis.ugent.be/