PARALLEL SYSTEMS ARCHITECTURE LAB

Customizing FPGA Designs using RapidWright

Ahmed Ben Haj Yahia

January 15th, 2020

Introduction & Abstract

- Emergence of cloud computing services using multi tenant FPGAs
- A diverse collection of FPGA voltage attacks that can be exploited in multi-tenant scenarios have been reported [1].
- Voltage sensors have already been used previously to assess voltage changes in the power distribution network under typical FPGA workloads [2].

So, the final goal of this project is to embed sesors into already placed and routed designs in order to monitor power or thermal consumption

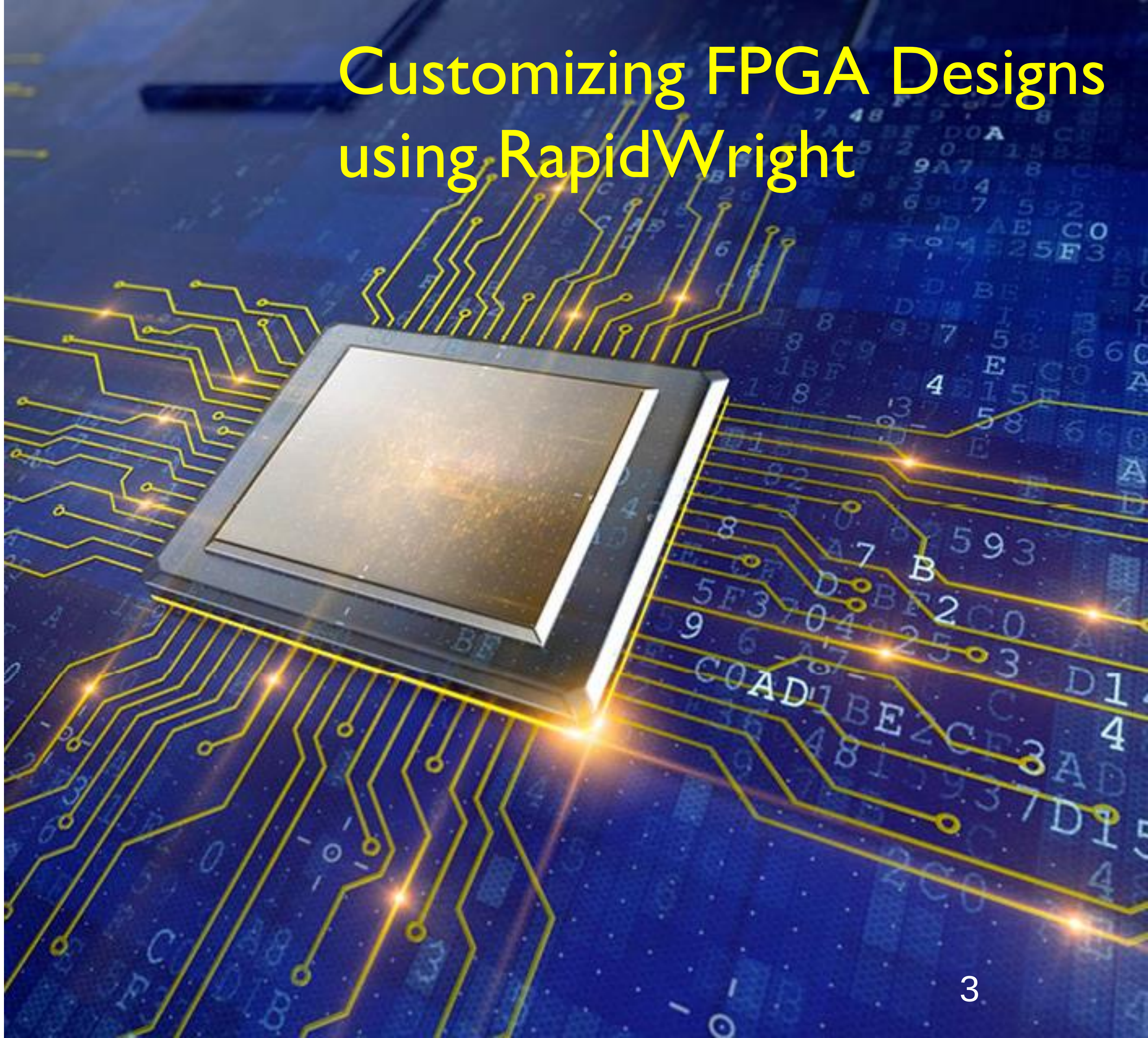
[1] S. S.Mirzargar and M. Stojilovic´, "Physical Side-Channel Attacks and Covert Communication on FPGAs: A Survey", EPFL, Tech. Rep., 2019.

[1] S. S.Mirzargar and M. Stojilovic´, "Physical Side-Channel Attacks and Covert Communication on FPGAs: A Survey", EPFL, Tech. Rep., 2019.

Outline

- Introduction
- **Overview**
  - **FPGA Architecture**
  - **RapidWright**
  - **Ring-oscillator sensor**

- Methods & Implementation
- Experiments and automating the process
- Results for VTR benchmarks



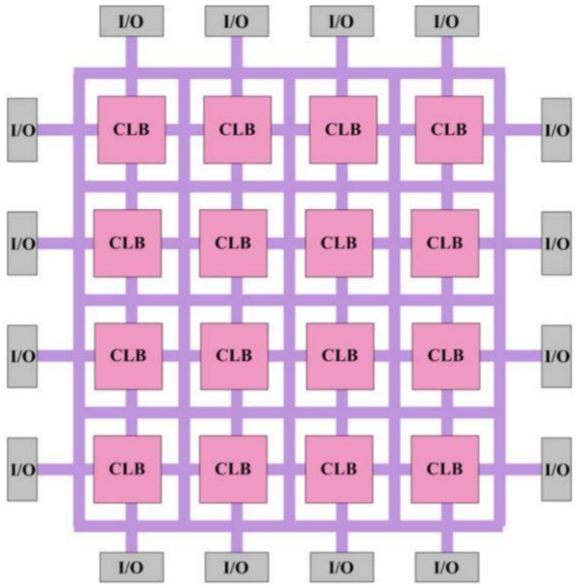
FPGA Architecture

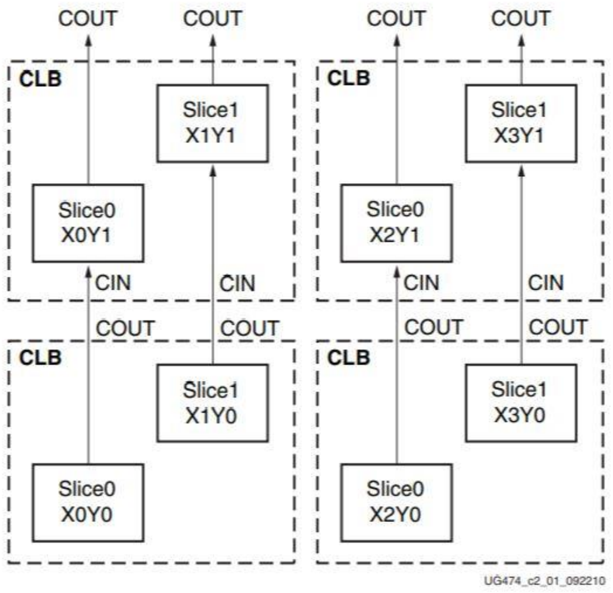
A **Field programmable gate array** (FPGA consists) of three major components:

- Configurable Logic Blocks (**CLB**), to implement logic functions
- Programmable Routing (interconnects), which implements functions
- **I/O blocks**, which are used to make off-chip connections

Each CLB contains a pair of slices, and each slice is composed of four LUTs (or **lookup tables**).

LUTs are constructed over simple memories that store Boolean functions. they have a fixed number of inputs and are coupled to a multiplexer and a Flip-flop in order to build any combinatorial or sequential circuits.

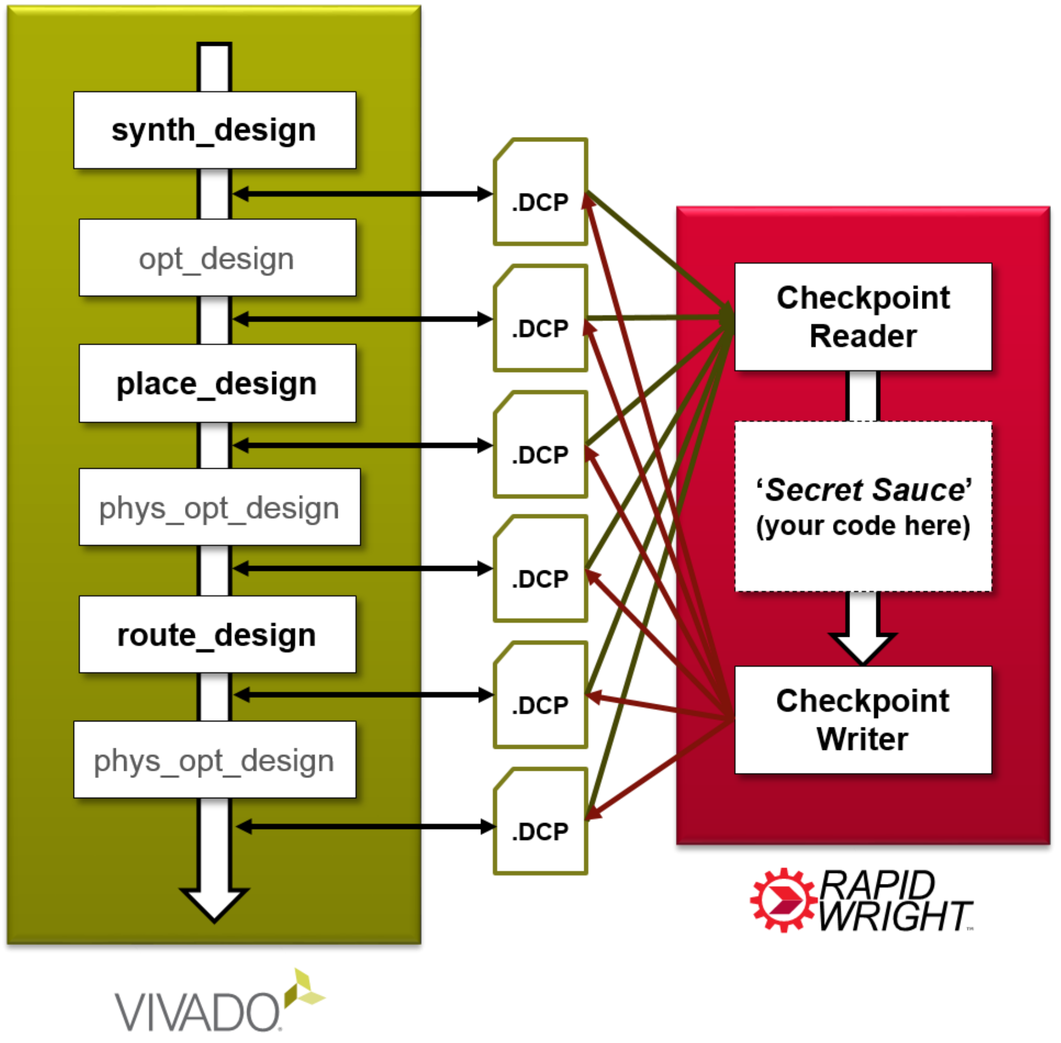




RapidWright

- RapidWright is an open source platform from Xilinx Research labs.
- It provides users with multiple API's in order to read, write and modify design check-point files (DCPs) outside of Vivado.

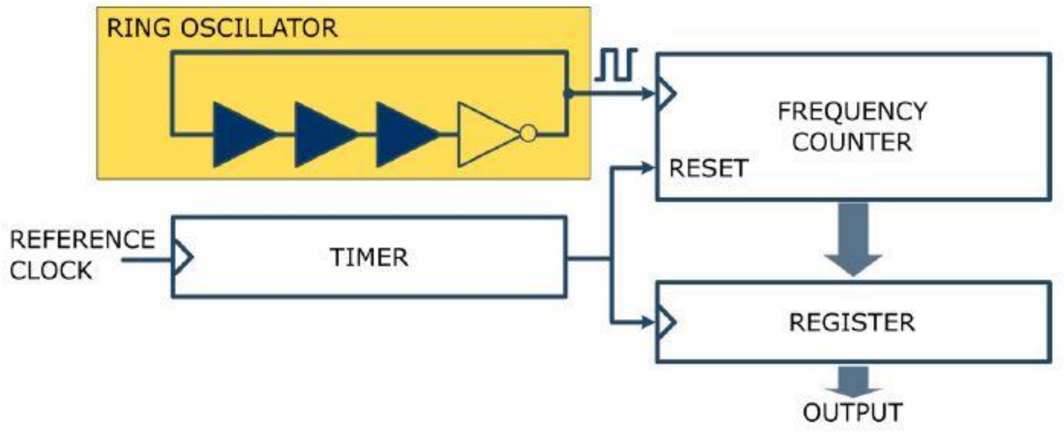
So, we will **read** a Design checkpoint file, use those **APIs** to embed our sensors, then **generate** a new resulting DCP.



Ring-oscillator sensors

- Conventional sensors for characterizing voltage noise of FPGAs are based on ring oscillators that feeds a frequency counter.
  - Voltage changes leads to a delay
  - =>measure that delay

- A **ring oscillator** (RO) is composed of an odd number of Inverters (NOT gates) in a loop.



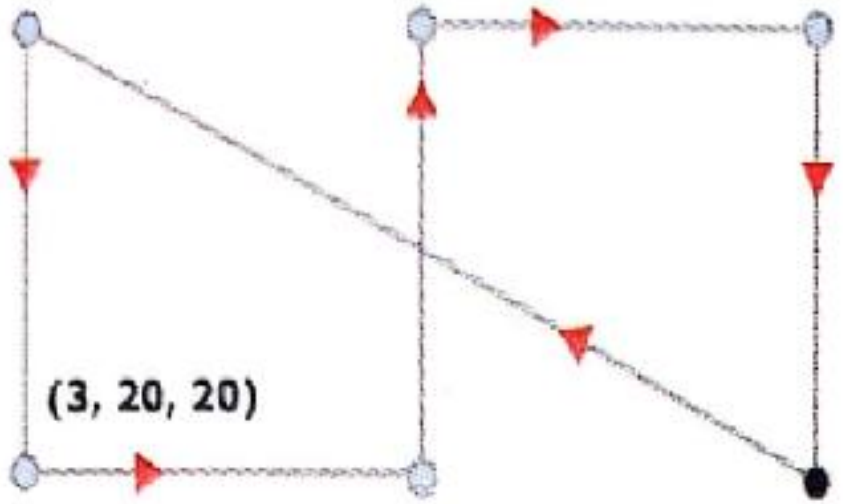
In our case, we configure only one of the LUTs of our RO sensors as Inverter, and let the other being buffers.

Ring-oscillator sensors

We define topology of our ring oscillators is using 3 parameters:

- N : number of unique X coordinates
- S : distance between adjacent columns
- H : distance between LUTs of same column

As in [3], we have limited our search space to a configurations (N=3,H=20, S=20). So, the ring oscillator sensors that we have used have 6 LUTs placed with regular spacing.



Note that a 7th LUT is added outside the loop, to connect our RO sensors to the controller

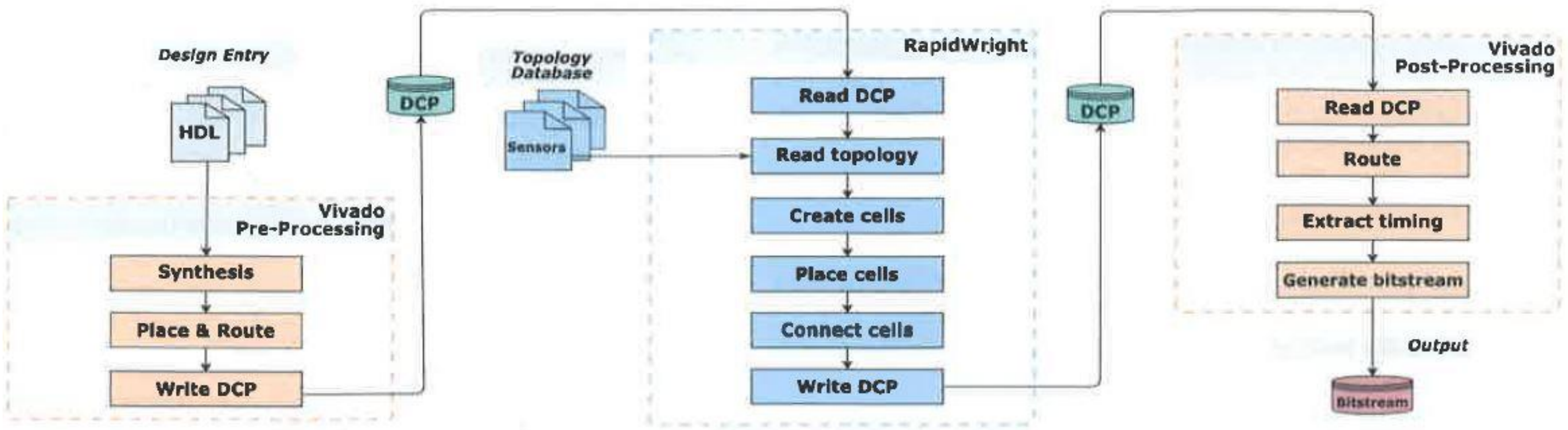
[3] CloudMoles, "Surveillance of Power-Wasting Activities by Infiltrating Undercover Sensors.", (under review).

Overview

So to summarize, we want to insert n=13 sensors into a selected design.

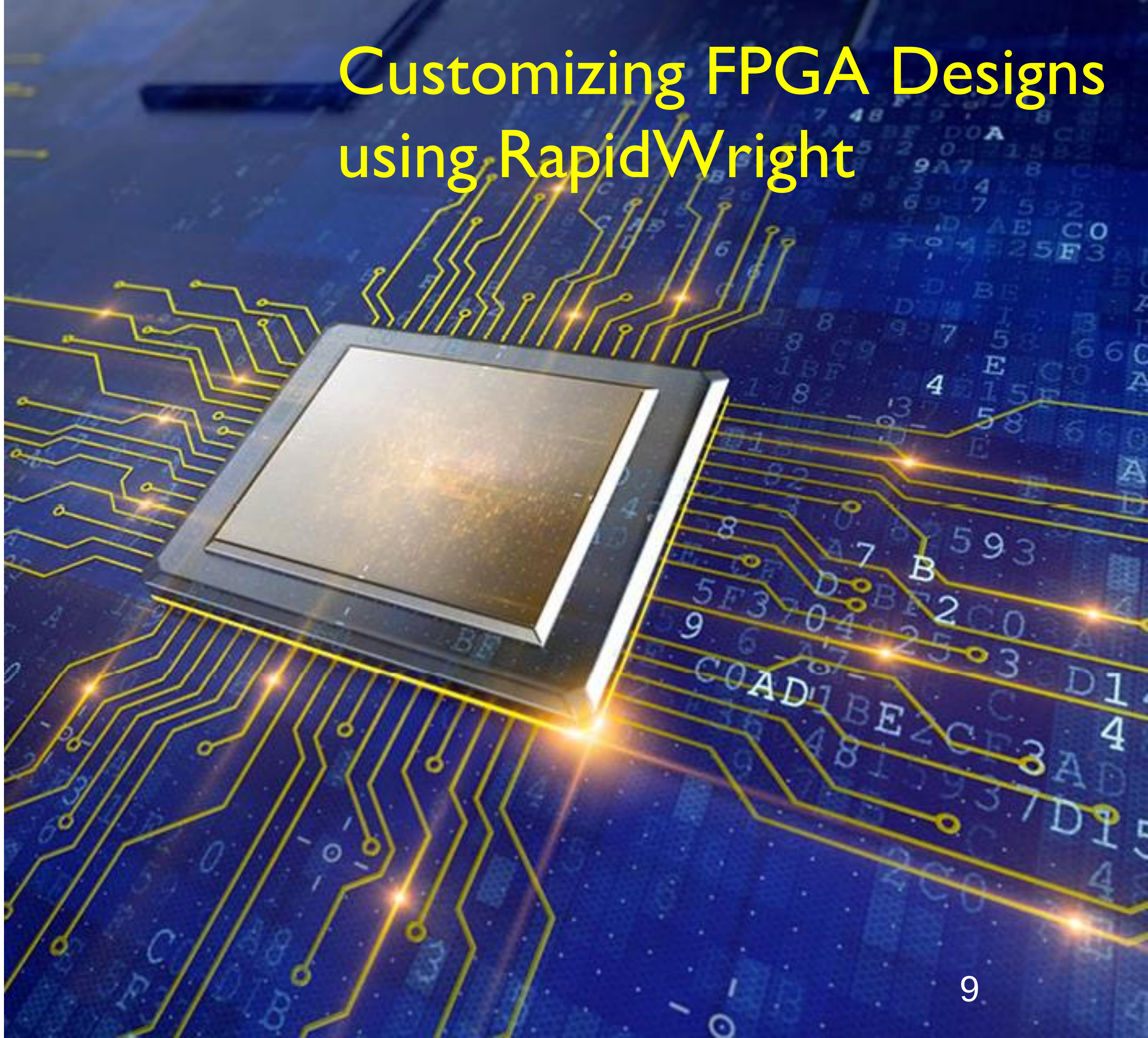
The coordinates and types of the LUTs forming these sensors are specified in a description file that will be used in RapidWright to select and insert the desired cells.

RapidWright will then output a DCP file and a TCL file containing corresponding TCL commands. These commands are finally applied by Vivado on the generated DCP to Place and Route, and generate the Bitstream.



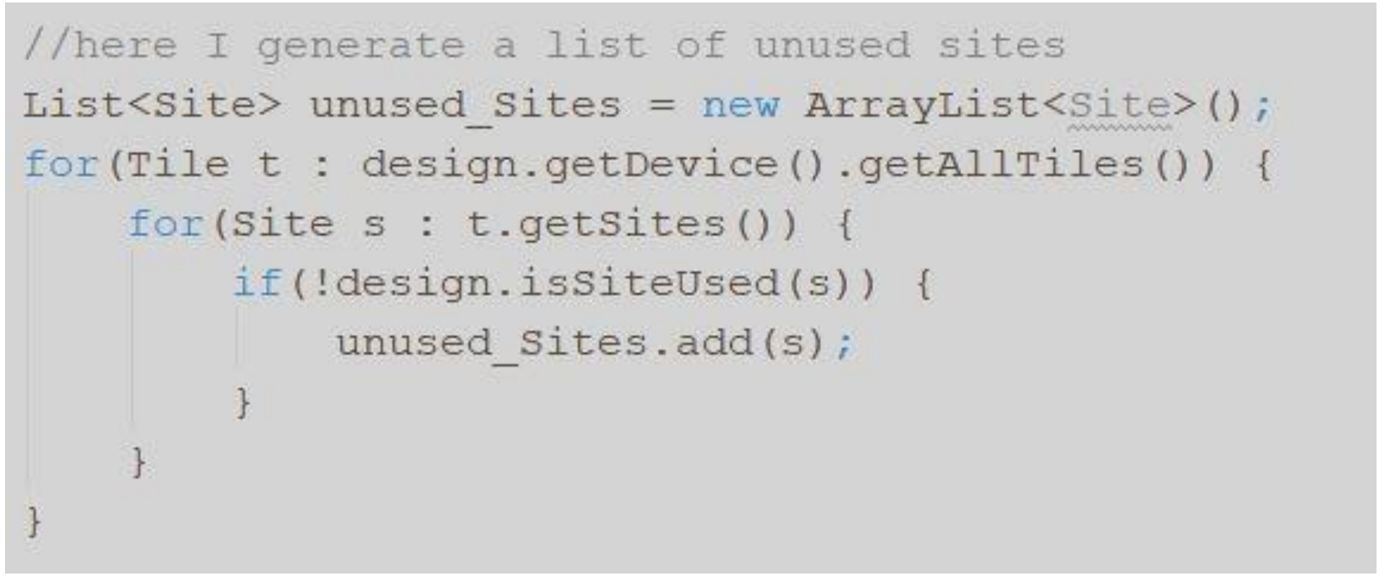
Outline

- Introduction
- Overview
  - FPGA Architecture
  - RapidWright
  - Ring-oscillator sensor
- **Methods & Implementation**
- Experiments and automating the process
- Results for VTR benchmarks



Finding free LUTs

For a specific design, we each time iterate over all tiles of the corresponding device. For each tile, we iterate over its sites; if a site is used -i.e. verified using API design class function *isSiteUsed(s)* - we consider all its BELs as occupied. Otherwise, we store it in *unused_sites*. From these unusedsites, we will generate a list of all free BELs and store it in ***free_cells***.



```java
//here I generate a list of unused sites
List<Site> unused_Sites = new ArrayList<Site>();
for(Tile t : design.getDevice().getAllTiles()) {
    for(Site s : t.getSites()) {
        if(!design.isSiteUsed(s)) {
            unused_Sites.add(s);
        }
    }
}
```

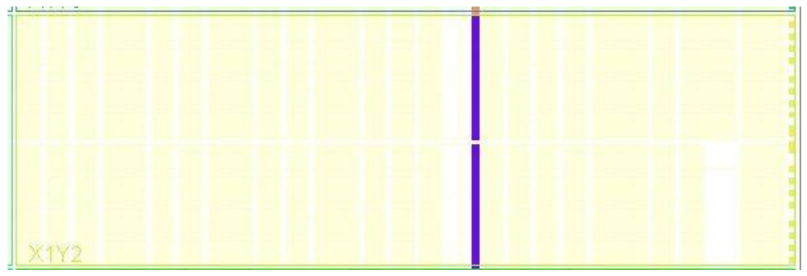
In our process, we consider that the design is changing each time a sensor is placed. So this method of finding free LUTs will be repeated several times.

Choosing a valid LUT

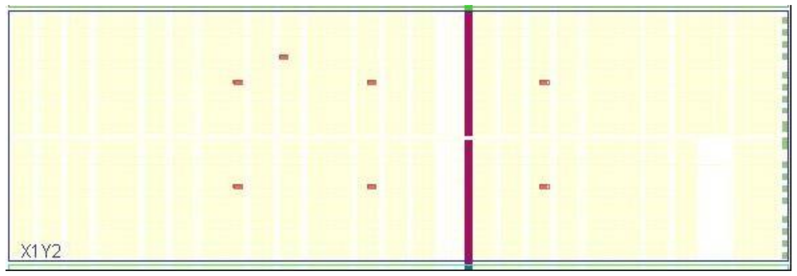
We implemented a function *generate_offsets* that, given a search radius, generates a list of offsets.

Using the desired location in the description file, these offsets are later used to create a list *candidates*. This list will contain the coordinates of all the LUTs located within the search radius from the desired location.

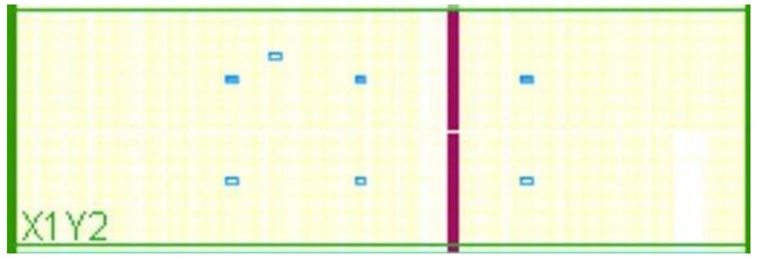
Once we have the two lists ***free_cells*** and ***candidates***, we keep only the ***free_candidates***.
If there is no free_candidates, we increment the search radius and search again, until we find a minimum of one free candidate LUT.



Empty Clock Region (CR)



CR with placed cells

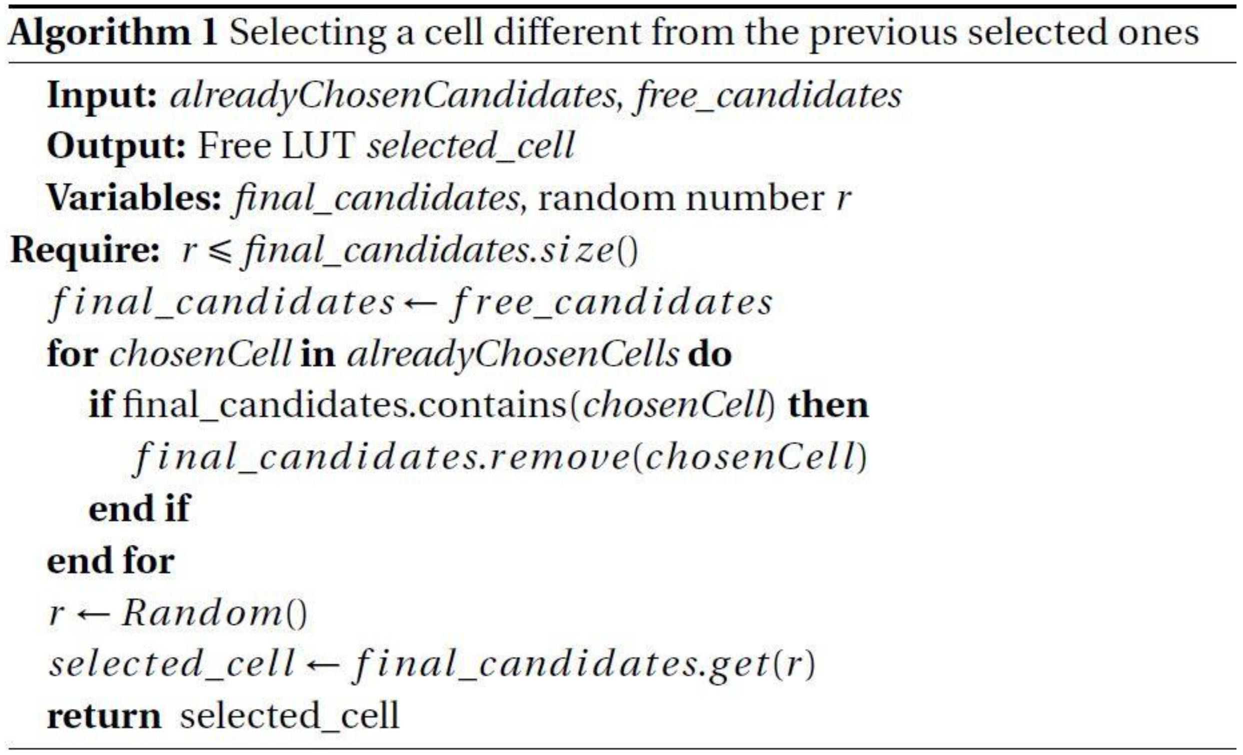


CR with place and routed cells

Generating multiple solutions

Later, when applying these methods to a specific design, and in order to maximize our chances of obtaining a valid configuration, we want to present **multiple** and **different** solutions.

This pseudo algorithm explains the procedure.



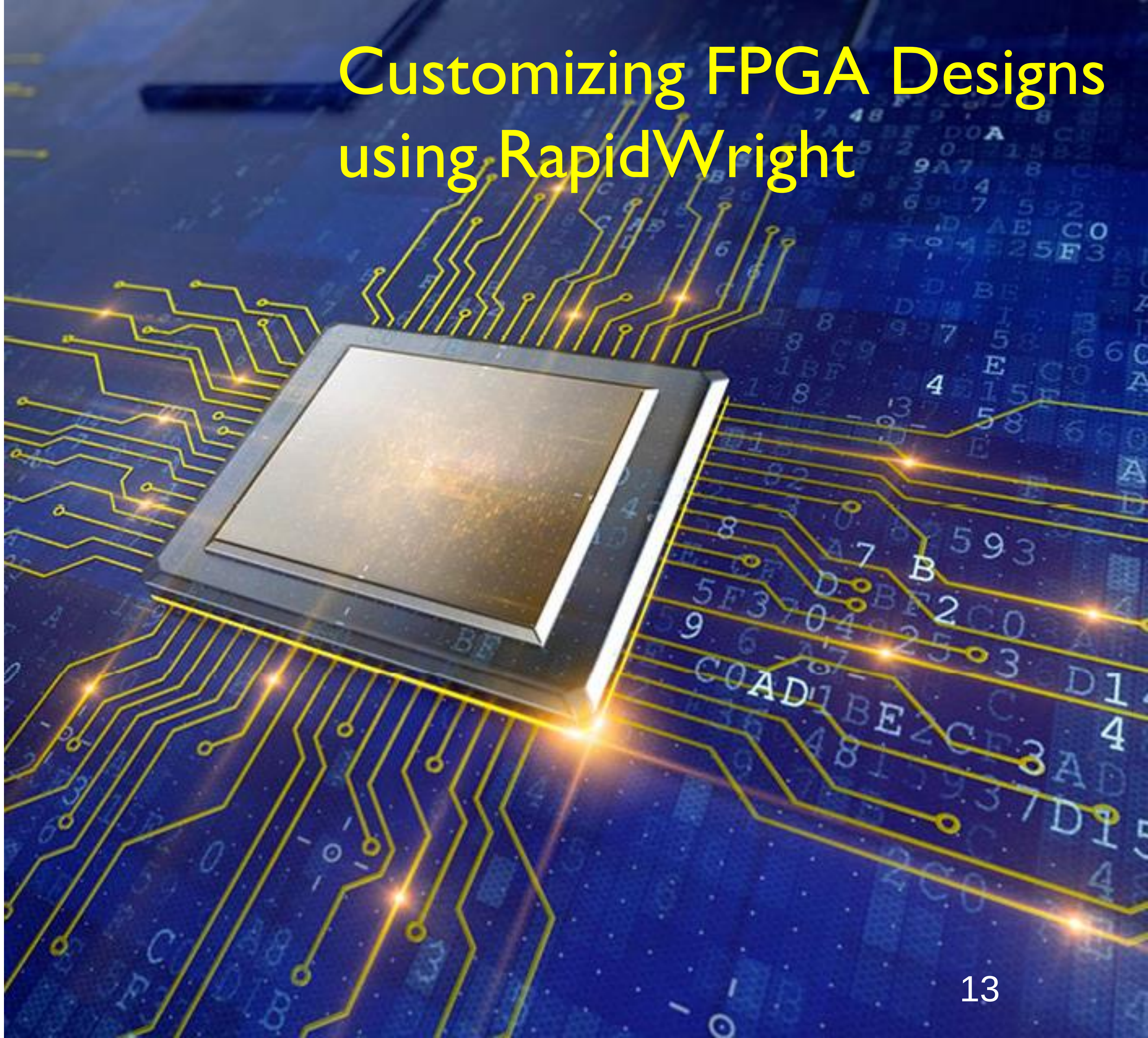
Algorithm 1 Selecting a cell different from the previous selected ones

```
    Input: $alreadyChosenCandidates, free\_candidates$
    Output: Free LUT $selected\_cell$
    Variables: $final\_candidates$, random number $r$
Require: $r \leqslant final\_candidates.size()$
    $final\_candidates \leftarrow free\_candidates$
    for $chosenCell$ in $alreadyChosenCells$ do
        if final_candidates.contains($chosenCell$) then
            $final\_candidates.remove(chosenCell)$
        end if
    end for
    $r \leftarrow Random()$
    $selected\_cell \leftarrow final\_candidates.get(r)$
    return selected_cell
```

So basically, we keep track of the already selected LUTs for previous solutions, and each time before selecting a valid LUT , we check if it has already been used previously.

Outline

- Introduction
- Overview
  - FPGA Architecture
  - RapidWright
  - Ring-oscillator sensor
- Methods & Implementation
- **Experiments and automating the process**
- Results for VTR benchmarks



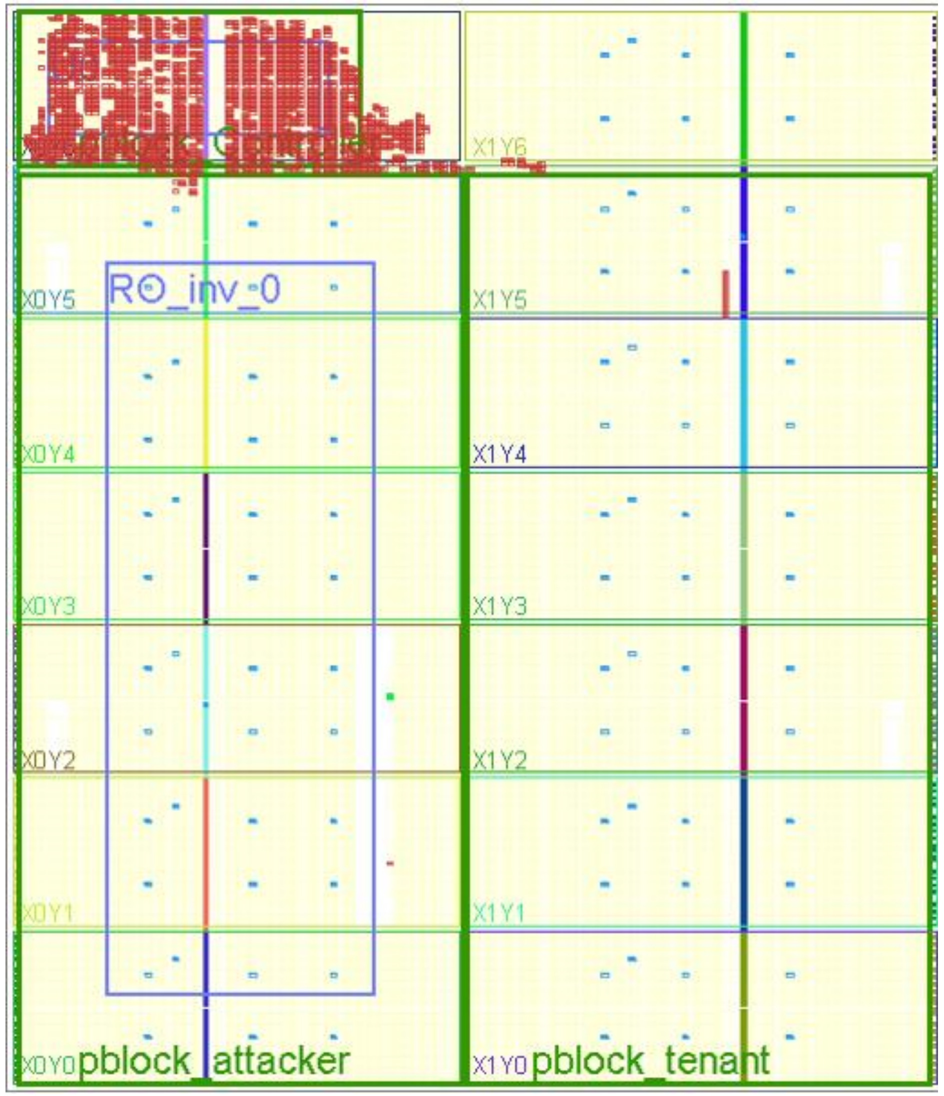
Experiments

1. On Empty Design

Since the design is empty, the desired cells should be free and the search radius will always stays at 0.

=>The coordinates of the selected cells are exactly the same as specified in the description file.

The figure shows our successfully placed and routed sensors into an empty design.



Experiments

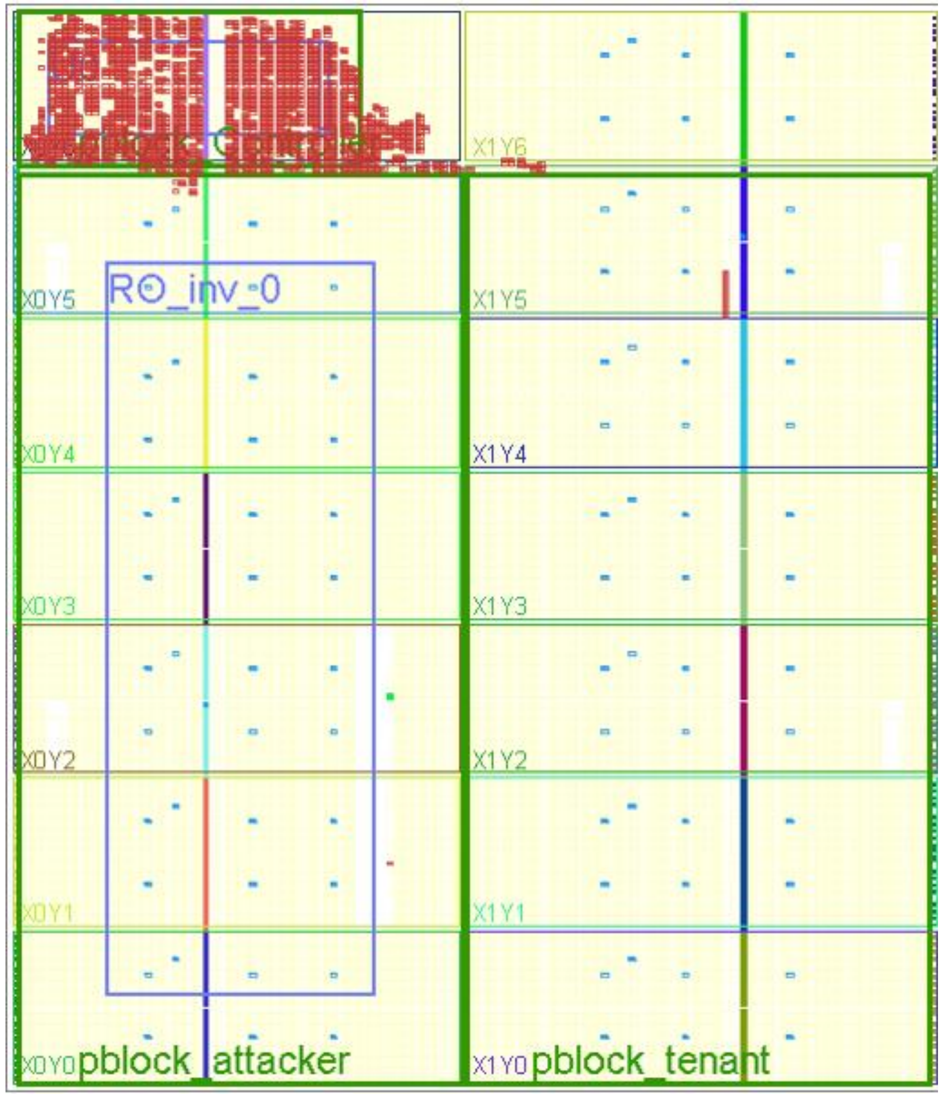
2. On Noisy Design

The noisy tenant is composed of the following components:

- **2000** instances of **seed_generator**
- **2000** instances of 32-bit **LFSRs** (Linear Feedback Shift Register)
- **2000** instances of 16-bit **adders**

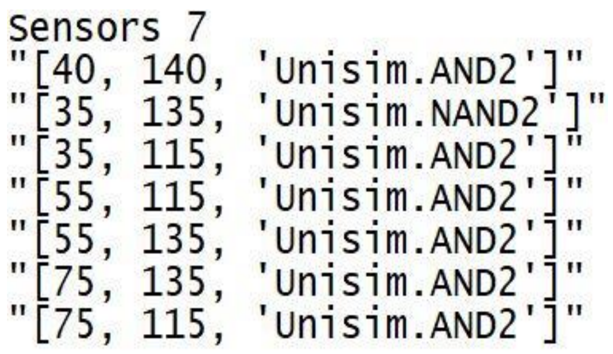
The operations between these components are intended to increase **power consumption**.

Since some desired cells are already being used by the design, the search for free cells lead to a higher search radius.



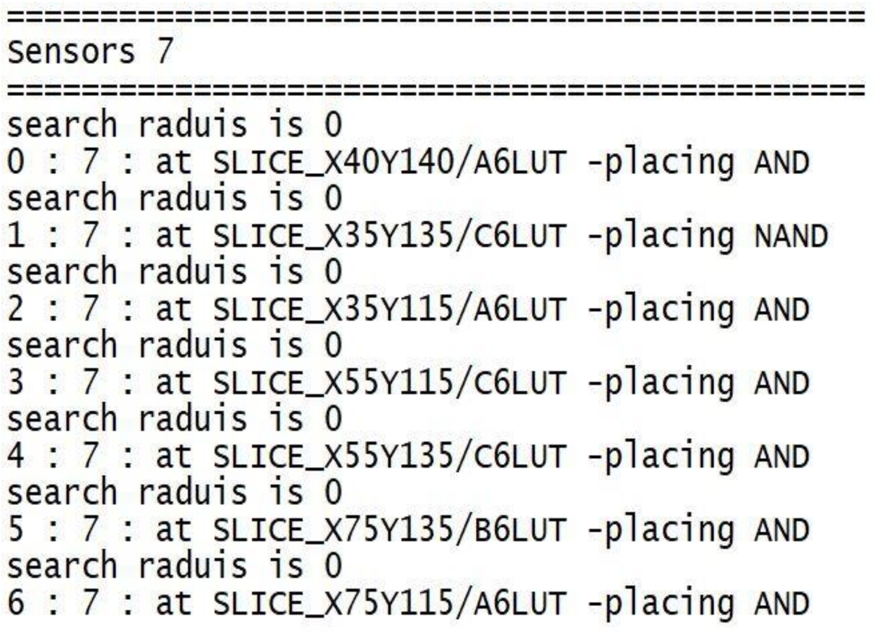
Experiments

3. Comparing locations of cells of sensor 7



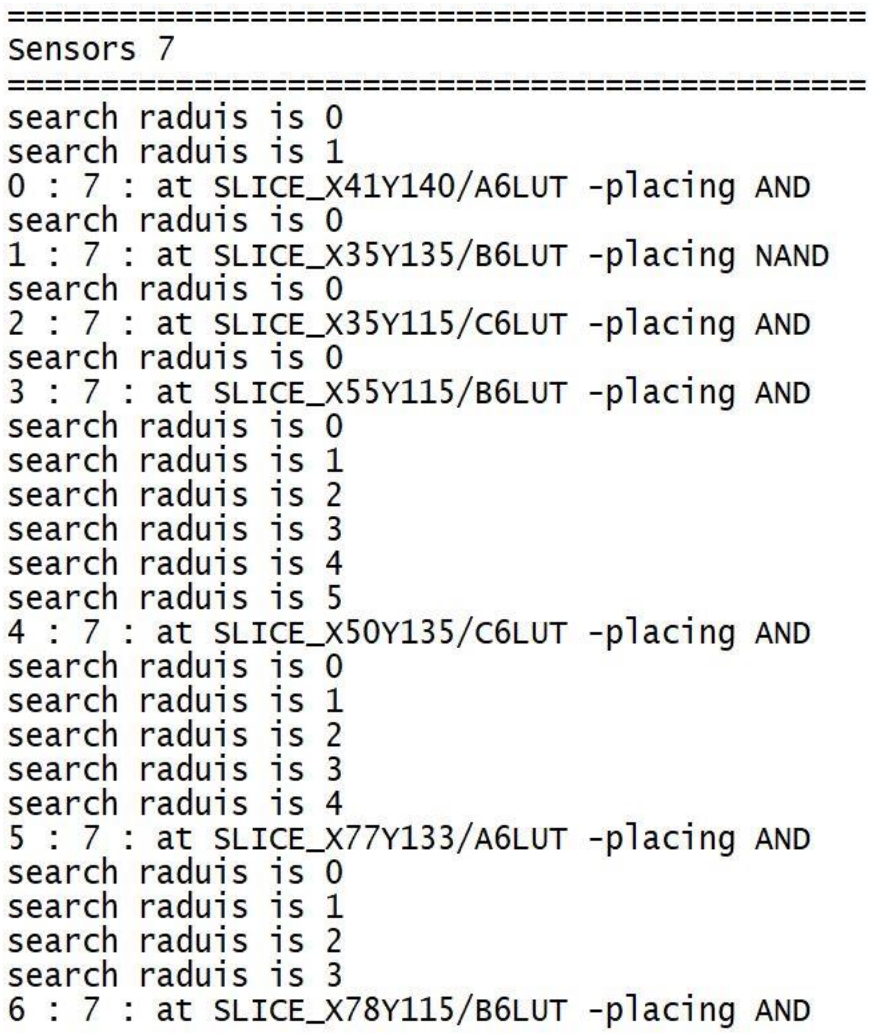
```
Sensors 7
"[40, 140, 'Unisim.AND2']"
"[35, 135, 'Unisim.NAND2']"
"[35, 115, 'Unisim.AND2']"
"[55, 115, 'Unisim.AND2']"
"[55, 135, 'Unisim.AND2']"
"[75, 135, 'Unisim.AND2']"
"[75, 115, 'Unisim.AND2']"
```

Desired locations from description file



```
==================================================
Sensors 7
==================================================
search raduis is 0
0 : 7 : at SLICE_X40Y140/A6LUT -placing AND
search raduis is 0
1 : 7 : at SLICE_X35Y135/C6LUT -placing NAND
search raduis is 0
2 : 7 : at SLICE_X35Y115/A6LUT -placing AND
search raduis is 0
3 : 7 : at SLICE_X55Y115/C6LUT -placing AND
search raduis is 0
4 : 7 : at SLICE_X55Y135/C6LUT -placing AND
search raduis is 0
5 : 7 : at SLICE_X75Y135/B6LUT -placing AND
search raduis is 0
6 : 7 : at SLICE_X75Y115/A6LUT -placing AND
```

Locations of selected LUTs on empty design



```
==================================================
Sensors 7
==================================================
search raduis is 0
search raduis is 1
0 : 7 : at SLICE_X41Y140/A6LUT -placing AND
search raduis is 0
1 : 7 : at SLICE_X35Y135/B6LUT -placing NAND
search raduis is 0
2 : 7 : at SLICE_X35Y115/C6LUT -placing AND
search raduis is 0
3 : 7 : at SLICE_X55Y115/B6LUT -placing AND
search raduis is 0
search raduis is 1
search raduis is 2
search raduis is 3
search raduis is 4
search raduis is 5
4 : 7 : at SLICE_X50Y135/C6LUT -placing AND
search raduis is 0
search raduis is 1
search raduis is 2
search raduis is 3
search raduis is 4
5 : 7 : at SLICE_X77Y133/A6LUT -placing AND
search raduis is 0
search raduis is 1
search raduis is 2
search raduis is 3
6 : 7 : at SLICE_X78Y115/B6LUT -placing AND
```

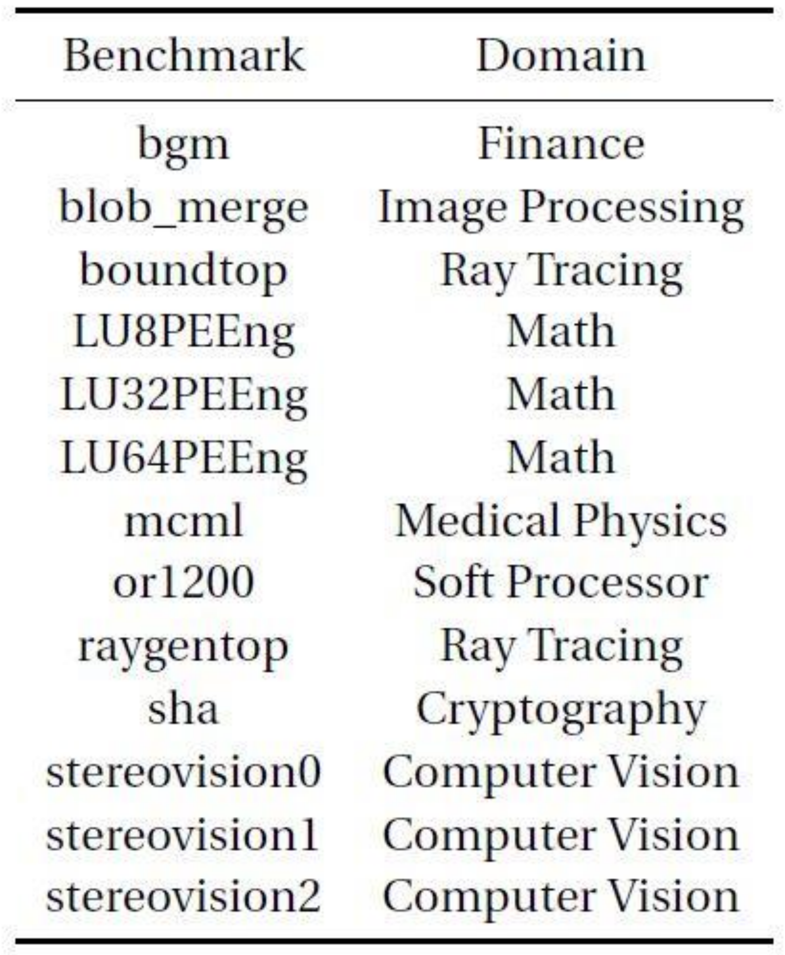
Locations of selected LUTs on noisy design

Automating the process

- VTR benchmarks

The VTR benchmarks shown in the figure are a set of real-sized industrial circuits included with Verilog-to-Routing (VTR).

So, the goal of the automated process is to generate 10 different solutions for each benchmark



Benchmark	Domain
bgm	Finance
blob_merge	Image Processing
boundtop	Ray Tracing
LU8PEEng	Math
LU32PEEng	Math
LU64PEEng	Math
mcml	Medical Physics
or1200	Soft Processor
raygentop	Ray Tracing
sha	Cryptography
stereovision0	Computer Vision
stereovision1	Computer Vision
stereovision2	Computer Vision

Automating the process

- On RapidWright

  At the end, It outputs a report Output.txt containing the following information:

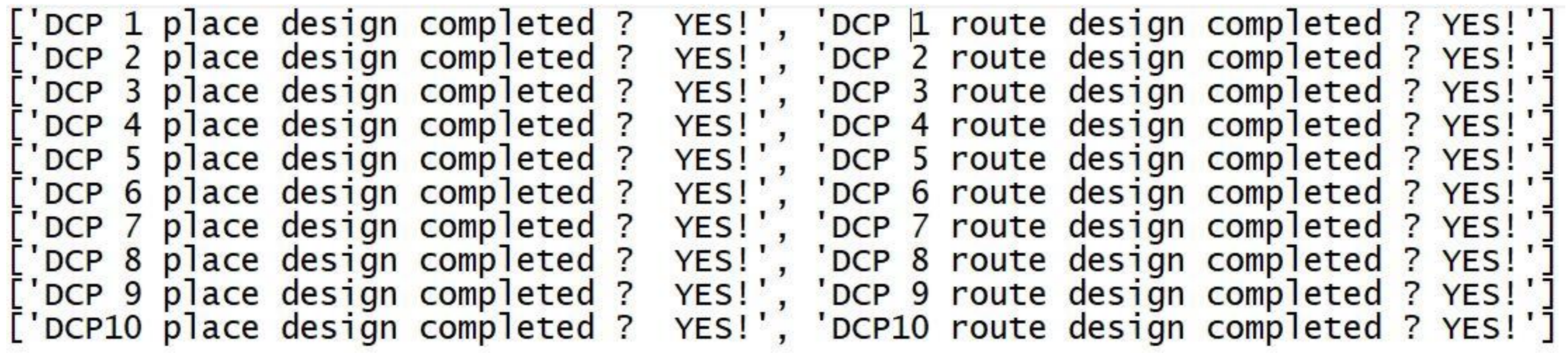
  - Which benchmark is being tested.
  - How many and precisely which Clock Regions are occupied by the benchmark.
  - The DCP being formed.
  - The sensor being placed.
  - The search radius
  - The selected cells, their location and the gate type

Automating the process

- On Vivado

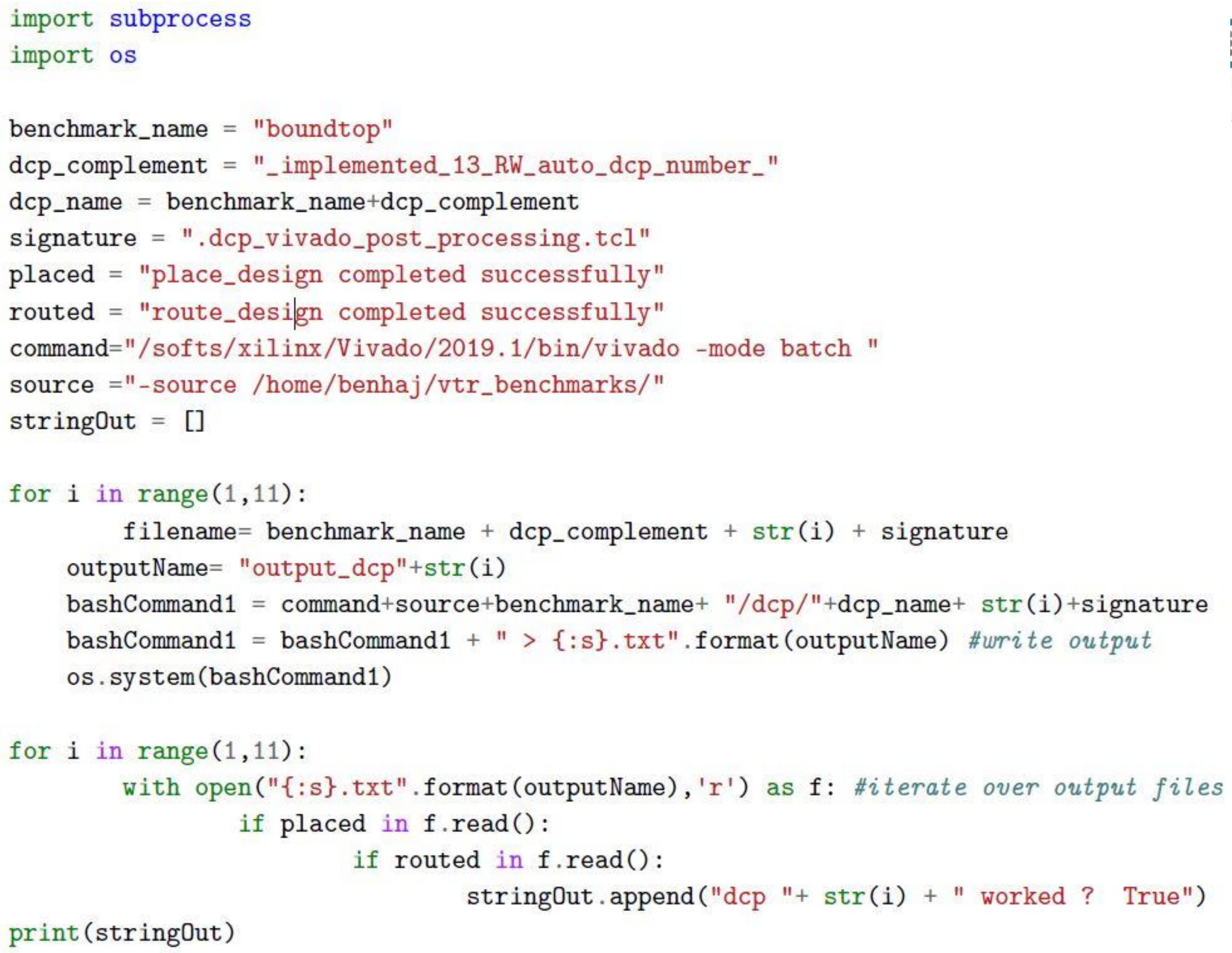
After generating the 10 DCP files, we will run a **script** (see next slide) that iterates over all the proposed solutions, and each time, apply the corresponding TCL commands on Vivado's batch mode.
It then outputs in **result.txt**, for each DCP, if the design was successfully placed and routed.

This is a view on the results.txt file for **bgm** benchmark



```
['DCP 1 place design completed ?  YES!', 'DCP 1 route design completed ? YES!']
['DCP 2 place design completed ?  YES!', 'DCP 2 route design completed ? YES!']
['DCP 3 place design completed ?  YES!', 'DCP 3 route design completed ? YES!']
['DCP 4 place design completed ?  YES!', 'DCP 4 route design completed ? YES!']
['DCP 5 place design completed ?  YES!', 'DCP 5 route design completed ? YES!']
['DCP 6 place design completed ?  YES!', 'DCP 6 route design completed ? YES!']
['DCP 7 place design completed ?  YES!', 'DCP 7 route design completed ? YES!']
['DCP 8 place design completed ?  YES!', 'DCP 8 route design completed ? YES!']
['DCP 9 place design completed ?  YES!', 'DCP 9 route design completed ? YES!']
['DCP10 place design completed ?  YES!', 'DCP10 route design completed ? YES!']
```

PARALLEL SYSTEMS
ARCHITECTURE LAB



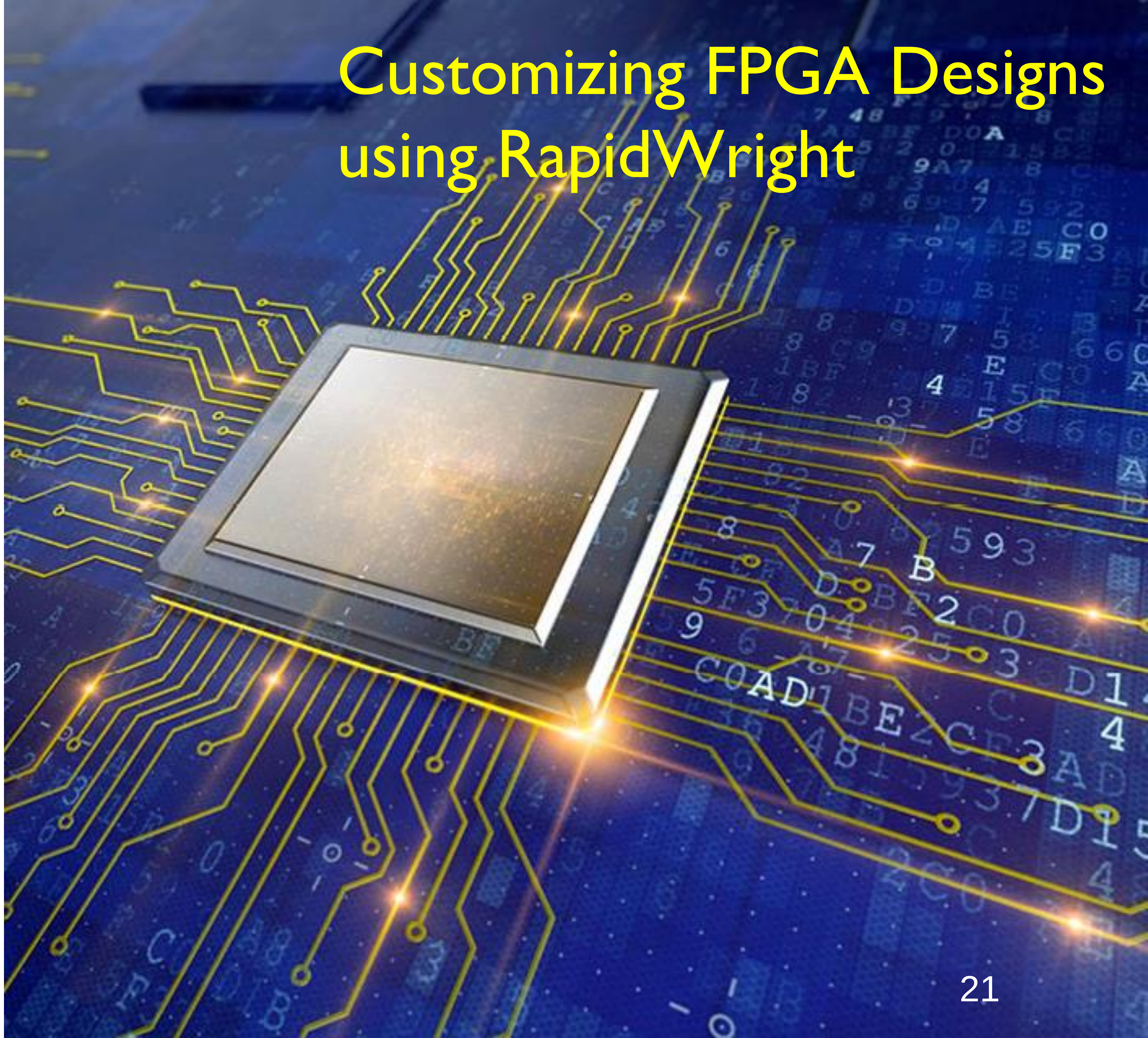
```python
import subprocess
import os

benchmark_name = "boundtop"
dcp_complement = "_implemented_13_RW_auto_dcp_number_"
dcp_name = benchmark_name+dcp_complement
signature = ".dcp_vivado_post_processing.tcl"
placed = "place_design completed successfully"
routed = "route_design completed successfully"
command="/softs/xilinx/Vivado/2019.1/bin/vivado -mode batch "
source ="-source /home/benhaj/vtr_benchmarks/"
stringOut = []

for i in range(1,11):
        filename= benchmark_name + dcp_complement + str(i) + signature
    outputName= "output_dcp"+str(i)
    bashCommand1 = command+source+benchmark_name+ "/dcp/"+dcp_name+ str(i)+signature
    bashCommand1 = bashCommand1 + " > {:s}.txt".format(outputName) #write output
    os.system(bashCommand1)

for i in range(1,11):
        with open("{:s}.txt".format(outputName),'r') as f: #iterate over output files
                if placed in f.read():
                        if routed in f.read():
                                stringOut.append("dcp "+ str(i) + " worked ?  True")
print(stringOut)
```

Outline



Customizing FPGA Designs using RapidWright

Results

```
==================================
RapidWright Sensors Placement
        vers. 1.0- EPFL-LAP 2019
==================================

Benchmark : bgm
3 used Clock Regions :  X1Y4  X1Y5  X1Y6
==================================
==========   DCP 1   ==============
==================================
==================================
Sensors 0
==================================
search raduis is 0
0 : 0 : at SLICE_X150Y340/B6LUT -placing AND
search raduis is 0
1 : 0 : at SLICE_X145Y335/A6LUT -placing NAND
search raduis is 0
2 : 0 : at SLICE_X145Y315/A6LUT -placing AND
search raduis is 0
3 : 0 : at SLICE_X165Y315/B6LUT -placing AND
search raduis is 0
4 : 0 : at SLICE_X165Y335/B6LUT -placing AND
search raduis is 0
5 : 0 : at SLICE_X185Y335/B6LUT -placing AND
search raduis is 0
6 : 0 : at SLICE_X185Y315/C6LUT -placing AND
==================================
Sensors 1
==================================
search raduis is 0
0 : 1 : at SLICE_X40Y285/C6LUT -placing AND
search raduis is 0
1 : 1 : at SLICE_X35Y280/C6LUT -placing NAND
search raduis is 0
2 : 1 : at SLICE_X35Y260/C6LUT -placing AND
search raduis is 0
3 : 1 : at SLICE_X55Y260/C6LUT -placing AND
search raduis is 0
4 : 1 : at SLICE_X55Y280/C6LUT -placing AND
search raduis is 0
5 : 1 : at SLICE_X75Y280/C6LUT -placing AND
search raduis is 0
6 : 1 : at SLICE_X75Y260/C6LUT -placing AND
==================================
Sensors 2
==================================
search raduis is 0
search raduis is 1
0 : 2 : at SLICE_X151Y290/B6LUT -placing AND
search raduis is 0
1 : 2 : at SLICE_X145Y285/B6LUT -placing NAND
search raduis is 0
2 : 2 : at SLICE_X145Y265/B6LUT -placing AND
search raduis is 0
search raduis is 1
search raduis is 2
3 : 2 : at SLICE_X165Y263/B6LUT -placing AND
search raduis is 0
search raduis is 1
search raduis is 2
search raduis is 3
4 : 2 : at SLICE_X165Y288/B6LUT -placing AND
search raduis is 0
5 : 2 : at SLICE_X185Y285/B6LUT -placing AND
search raduis is 0
6 : 2 : at SLICE_X185Y265/C6LUT -placing AND
==================================
Sensors 3
==================================
search raduis is 0
0 : 3 : at SLICE_X40Y235/C6LUT -placing AND
search raduis is 0
1 : 3 : at SLICE_X35Y230/A6LUT -placing NAND
search raduis is 0
2 : 3 : at SLICE_X35Y210/B6LUT -placing AND
search raduis is 0
3 : 3 : at SLICE_X55Y210/C6LUT -placing AND
search raduis is 0
4 : 3 : at SLICE_X55Y230/C6LUT -placing AND
search raduis is 0
5 : 3 : at SLICE_X75Y230/A6LUT -placing AND
search raduis is 0
6 : 3 : at SLICE_X75Y210/A6LUT -placing AND
==================================
Sensors 4
==================================
search raduis is 0
search raduis is 1
search raduis is 2
search raduis is 3
0 : 4 : at SLICE_X147Y240/C6LUT -placing AND
search raduis is 0
1 : 4 : at SLICE_X145Y235/C6LUT -placing NAND
search raduis is 0
2 : 4 : at SLICE_X145Y215/B6LUT -placing AND
search raduis is 0
3 : 4 : at SLICE_X165Y215/A6LUT -placing AND
search raduis is 0
search raduis is 1
4 : 4 : at SLICE_X165Y234/C6LUT -placing AND
search raduis is 0
5 : 4 : at SLICE_X185Y235/A6LUT -placing AND
search raduis is 0
6 : 4 : at SLICE_X185Y215/A6LUT -placing AND
==================================
Sensors 5
==================================
search raduis is 0
0 : 5 : at SLICE_X40Y190/B6LUT -placing AND
search raduis is 0
1 : 5 : at SLICE_X35Y185/A6LUT -placing NAND
search raduis is 0
2 : 5 : at SLICE_X35Y165/C6LUT -placing AND
search raduis is 0
3 : 5 : at SLICE_X55Y165/C6LUT -placing AND
search raduis is 0
4 : 5 : at SLICE_X55Y185/C6LUT -placing AND
search raduis is 0
5 : 5 : at SLICE_X75Y185/A6LUT -placing AND
search raduis is 0
6 : 5 : at SLICE_X75Y165/C6LUT -placing AND
==================================
Sensors 6
==================================
search raduis is 0
0 : 6 : at SLICE_X150Y190/C6LUT -placing AND
search raduis is 0
1 : 6 : at SLICE_X145Y185/A6LUT -placing NAND
search raduis is 0
2 : 6 : at SLICE_X145Y165/B6LUT -placing AND
search raduis is 0
3 : 6 : at SLICE_X165Y165/B6LUT -placing AND
search raduis is 0
4 : 6 : at SLICE_X165Y185/C6LUT -placing AND
search raduis is 0
5 : 6 : at SLICE_X185Y185/A6LUT -placing AND
search raduis is 0
6 : 6 : at SLICE_X185Y165/B6LUT -placing AND
==================================
Sensors 7
==================================
search raduis is 0
0 : 7 : at SLICE_X40Y140/C6LUT -placing AND
search raduis is 0
1 : 7 : at SLICE_X35Y135/A6LUT -placing NAND
search raduis is 0
2 : 7 : at SLICE_X35Y115/B6LUT -placing AND
search raduis is 0
3 : 7 : at SLICE_X55Y115/B6LUT -placing AND
search raduis is 0
4 : 7 : at SLICE_X55Y135/B6LUT -placing AND
search raduis is 0
5 : 7 : at SLICE_X75Y135/g AND
search raduis is 0C6LUT -placin
6 : 7 : at SLICE_X75Y115/C6LUT -placing AND
==================================
Sensors 8
==================================
search raduis is 0
0 : 8 : at SLICE_X150Y140/B6LUT -placing AND
search raduis is 0
1 : 8 : at SLICE_X145Y135/B6LUT -placing NAND
search raduis is 0
2 : 8 : at SLICE_X145Y115/B6LUT -placing AND
search raduis is 0
3 : 8 : at SLICE_X165Y115/C6LUT -placing AND
search raduis is 0
4 : 8 : at SLICE_X165Y135/C6LUT -placing AND
search raduis is 0
5 : 8 : at SLICE_X185Y135/B6LUT -placing AND
search raduis is 0
6 : 8 : at SLICE_X185Y115/A6LUT -placing AND
==================================
Sensors 9
==================================
search raduis is 0
0 : 9 : at SLICE_X40Y90/B6LUT -placing AND
search raduis is 0
1 : 9 : at SLICE_X35Y85/C6LUT -placing NAND
search raduis is 0
2 : 9 : at SLICE_X35Y65/C6LUT -placing AND
search raduis is 0
3 : 9 : at SLICE_X55Y65/A6LUT -placing AND
search raduis is 0
4 : 9 : at SLICE_X55Y85/C6LUT -placing AND
search raduis is 0
5 : 9 : at SLICE_X75Y85/A6LUT -placing AND
search raduis is 0
6 : 9 : at SLICE_X75Y65/C6LUT -placing AND
==================================
Sensors 10
==================================
search raduis is 0
0 : 10 : at SLICE_X150Y90/C6LUT -placing AND
search raduis is 0
1 : 10 : at SLICE_X145Y85/B6LUT -placing NAND
search raduis is 0
2 : 10 : at SLICE_X145Y65/B6LUT -placing AND
search raduis is 0
3 : 10 : at SLICE_X165Y65/C6LUT -placing AND
search raduis is 0
4 : 10 : at SLICE_X165Y85/C6LUT -placing AND
search raduis is 0
5 : 10 : at SLICE_X185Y85/B6LUT -placing AND
search raduis is 0
6 : 10 : at SLICE_X185Y65/B6LUT -placing AND
==================================
Sensors 11
==================================
search raduis is 0
0 : 11 : at SLICE_X40Y40/B6LUT -placing AND
search raduis is 0
1 : 11 : at SLICE_X35Y35/C6LUT -placing NAND
search raduis is 0
2 : 11 : at SLICE_X35Y15/B6LUT -placing AND
search raduis is 0
3 : 11 : at SLICE_X55Y15/A6LUT -placing AND
search raduis is 0
4 : 11 : at SLICE_X55Y35/A6LUT -placing AND
search raduis is 0
5 : 11 : at SLICE_X75Y35/C6LUT -placing AND
search raduis is 0
6 : 11 : at SLICE_X75Y15/A6LUT -placing AND
==================================
Sensors 12
==================================
search raduis is 0
0 : 12 : at SLICE_X150Y40/A6LUT -placing AND
search raduis is 0
1 : 12 : at SLICE_X145Y35/A6LUT -placing NAND
search raduis is 0
2 : 12 : at SLICE_X145Y15/B6LUT -placing AND
search raduis is 0
3 : 12 : at SLICE_X165Y15/A6LUT -placing AND
search raduis is 0
4 : 12 : at SLICE_X165Y35/C6LUT -placing AND
search raduis is 0
5 : 12 : at SLICE_X185Y35/C6LUT -placing AND
search raduis is 0
6 : 12 : at SLICE_X185Y15/A6LUT -placing AND
==================================
```

Sensor 2 in X1Y6
Sensor 4 in X1Y5
Sensor 6 in X1Y4

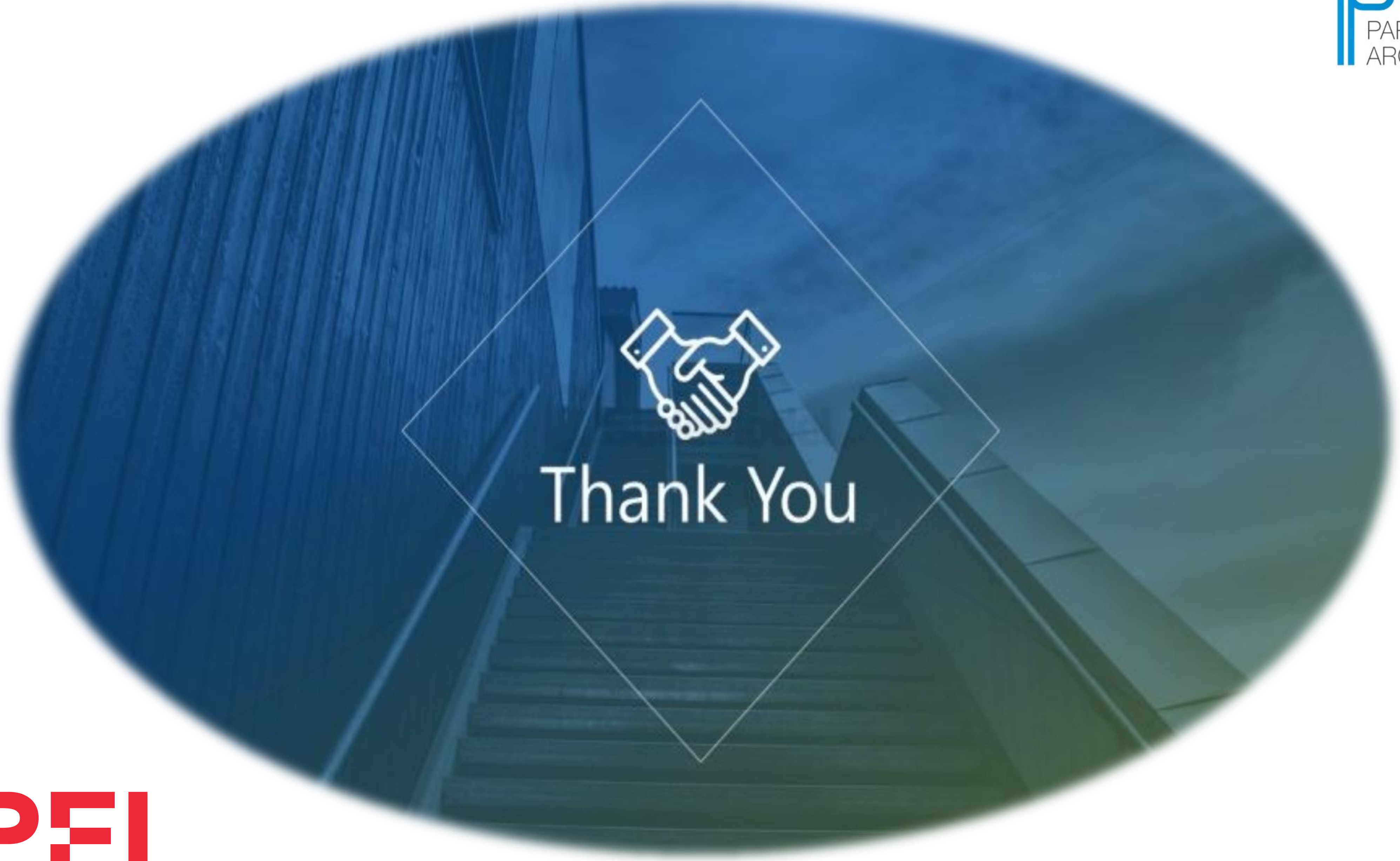
Results

The maximum search radius reached was in **mcml** benchmark with a value of **9.**
(It's the most noisy design)

The maximum in all the other benchmarks is **5**.

For designs **or1200** and **boundtop** the locations of the selected LUTs were the same as desired. (They are quite empty)

```
=================================
Sensors 9
=================================search radius is 0
0 : 9 : at SLICE_X40Y90/B6LUT -placing AND
search raduis is 0
1 : 9 : at SLICE_X35Y85/A6LUT -placing
NAND
search raduis is 0
2 : 9 : at SLICE_X35Y65/A6LUT -placing AND
search raduis is 0
search raduis is 1
search raduis is 2
search raduis is 3
search raduis is 4
search raduis is 5
search raduis is 6
search raduis is 7
3 : 9 : at SLICE_X57Y60/C6LUT -placing AND
search raduis is 0
search raduis is 1
search raduis is 2
search raduis is 3
search raduis is 4
search raduis is 5
search raduis is 6
search raduis is 7
search raduis is 8
4 : 9 : at SLICE_X47Y85/B6LUT -placing AND
search raduis is 0
search raduis is 1
search raduis is 2
search raduis is 3
search raduis is 4
search raduis is 5
search raduis is 6
search raduis is 7
search raduis is 8
search raduis is 9
5 : 9 : at SLICE_X79Y80/A6LUT -placing AND
search raduis is 0
search raduis is 1
search raduis is 2
search raduis is 3
search raduis is 4
6 : 9 : at SLICE_X71Y65/A6LUT -placing AND
```



PARALLEL SYSTEMS ARCHITECTURE LAB

EPFL

ahmed.benhajyahia@epfl.ch