cnrs GDR Groupement de recherche
SOC² System On Chip - Systèmes embarqués et Objets Connectés

Exploration des architectures reconfigurables avec l'outil Verilog-To-Routing (VTR)

Workshop GDR SOC² - Tutoriel VTR

Sylvain TAKOUGANG

13/06/2023

Plan du tutoriel

Introduction :

- Flot typique de développement d'un FPGA
- VTR et outils libres

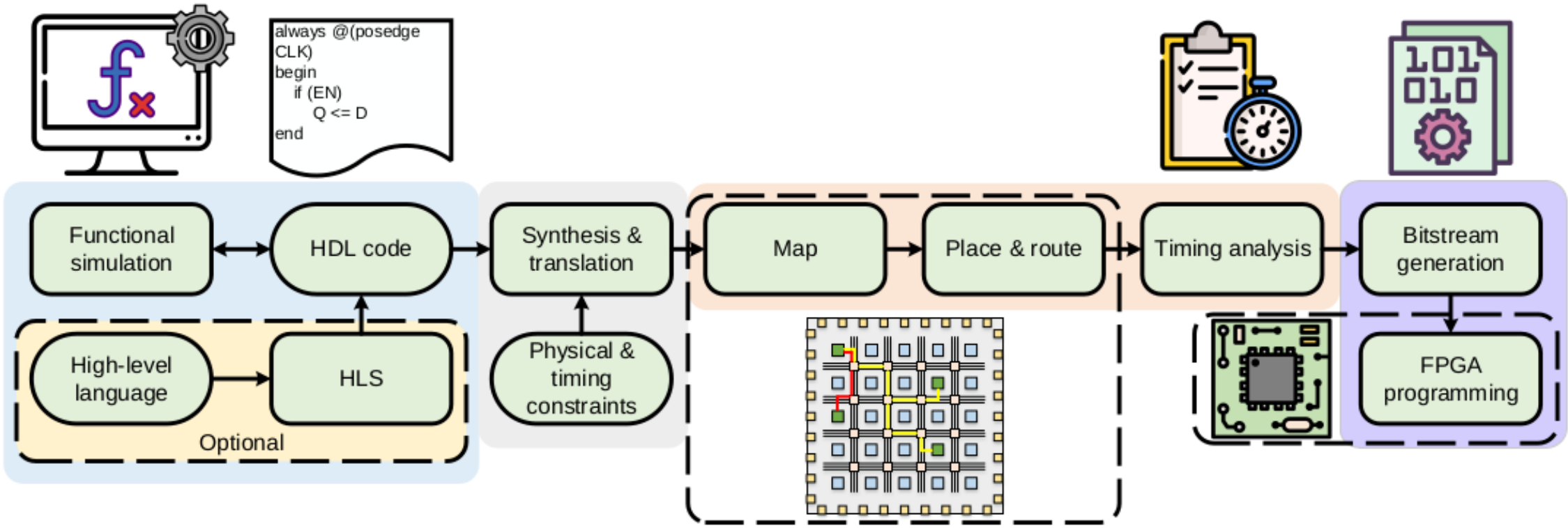
Prise en main VTR :

- Exécution manuel et automatique du flow VTR
- Modification d'architecture d'un FPGA et évaluations
- Graphe de routage, FASM et bitstream

Mini challenge :

- Implémenter le circuit SHA sur le FPGA fournit avec pour cahier de charge l'utilisation d'environ 70% les ressources logiques du FPGA

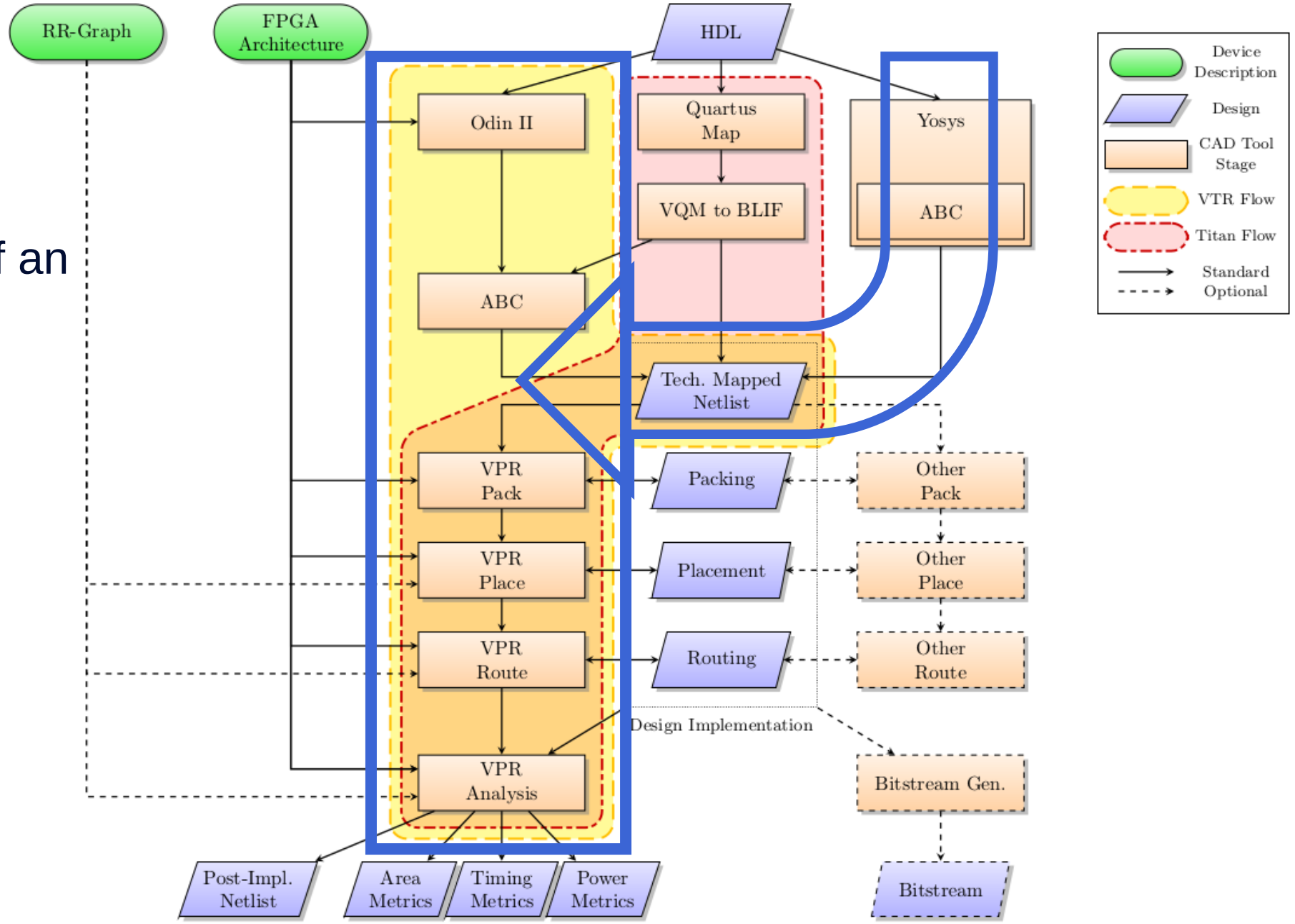
Flot de développement d'un FPGA



Flot VTR

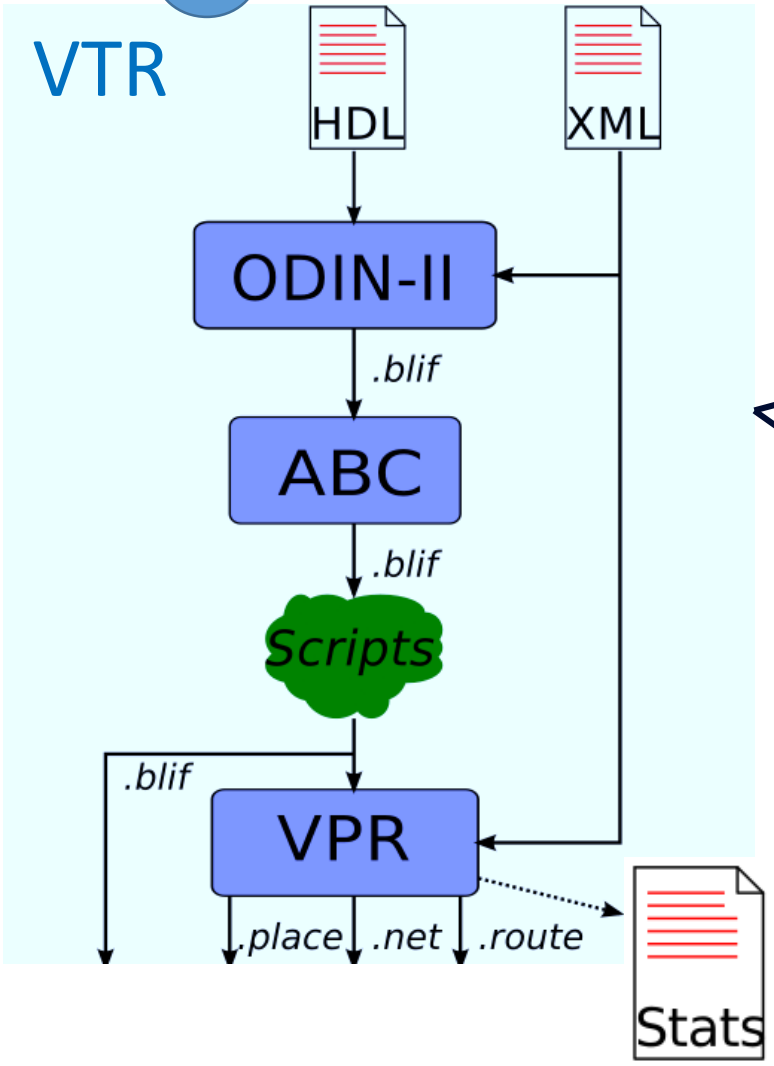
VTR inputs:

- XML description of an FPGA architecture
- Verilog/Blif/e(blif) circuit file

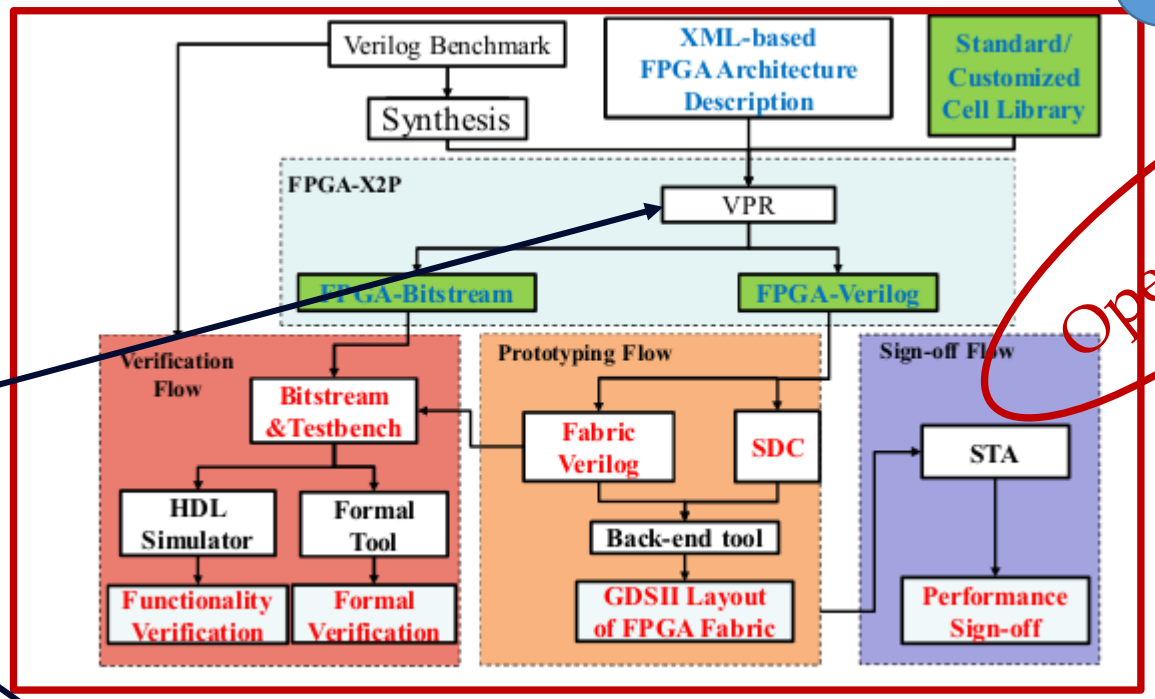


VTR et Outils libres …

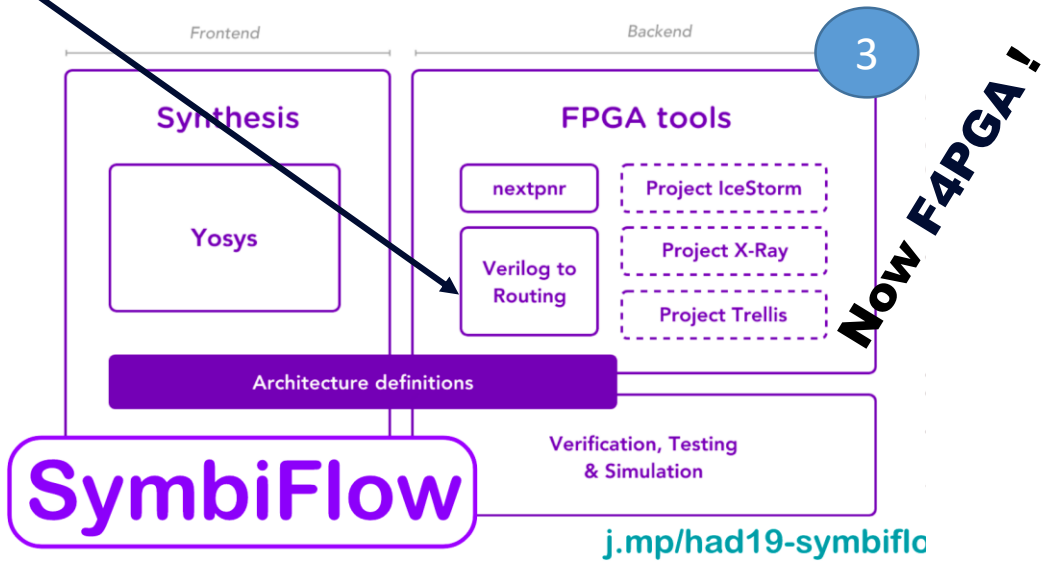
1



2



3

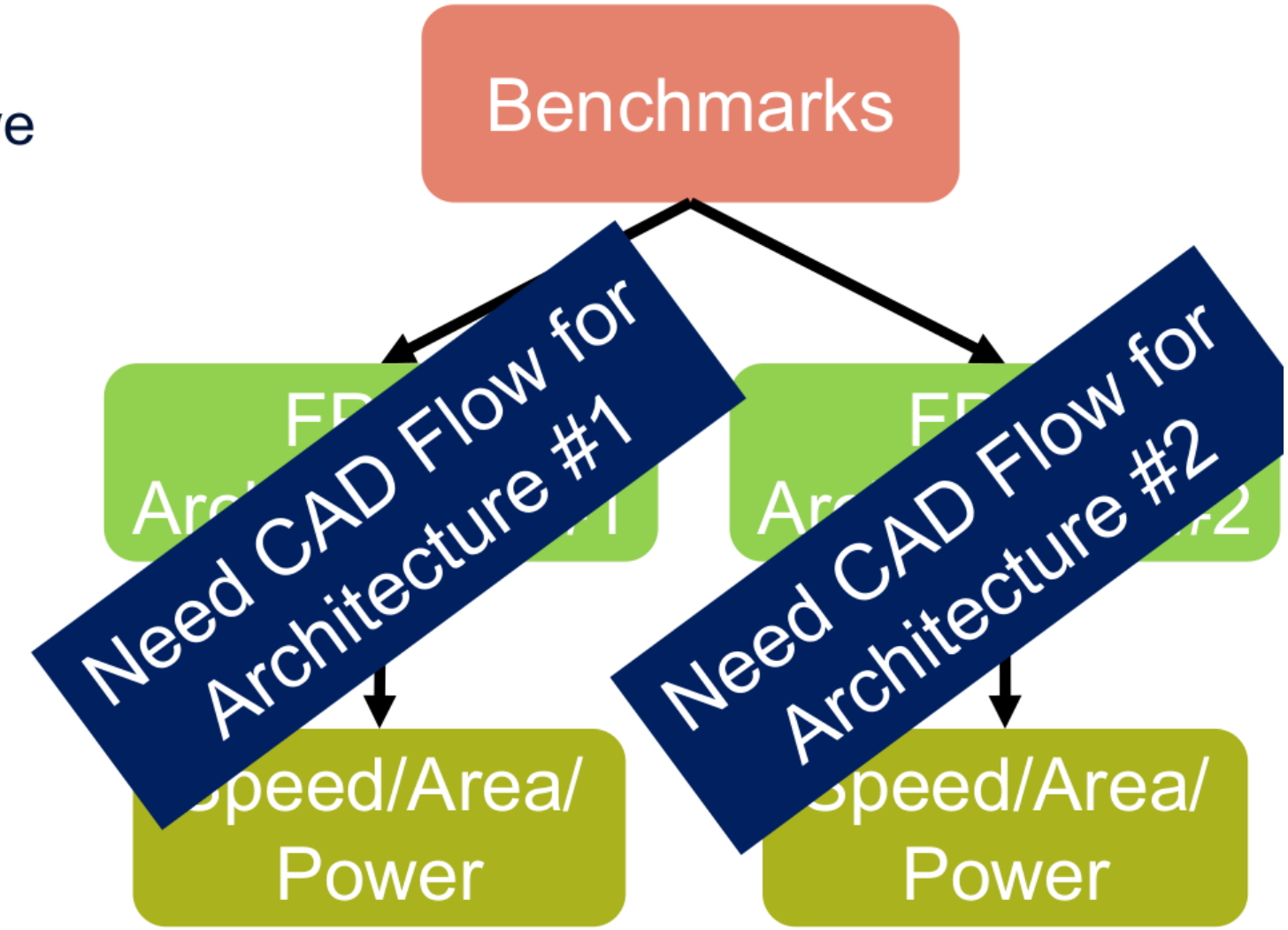


Besoins et Objectifs

Challenges:

- Need representative benchmarks
- Describe FPGA Architecture
- **Need high quality CAD flow**

FPGA architecture research

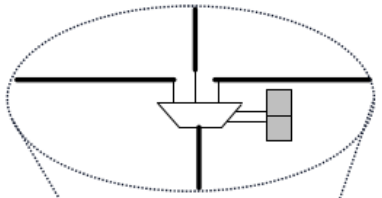


https://www.eecg.utoronto.ca/~vaughn/papers/vlsi_soc_2020_vtr.pdf

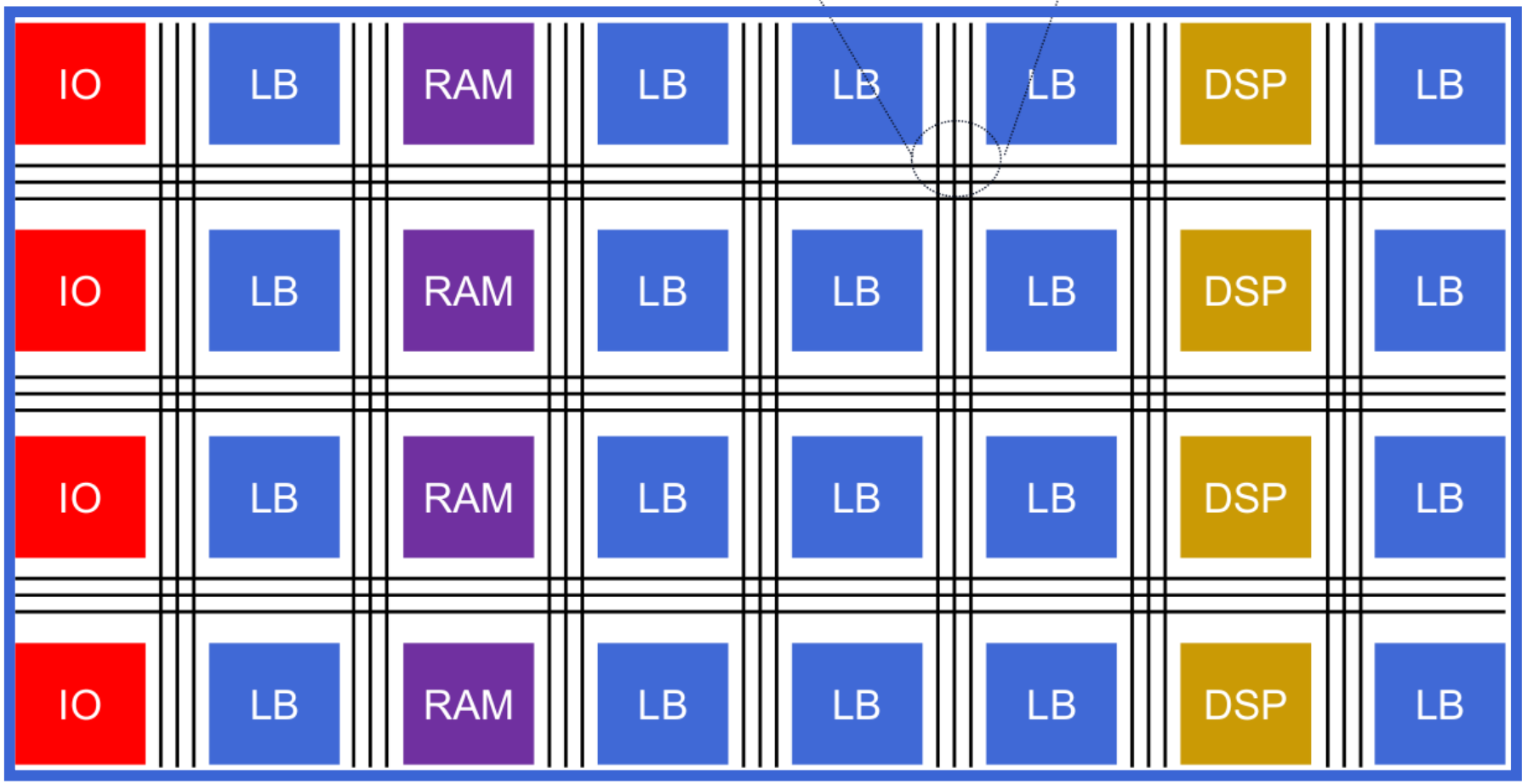
Rappel FPGA et customisation

Device

- Grid of Function Blocks
- Interconnect



Routing Switch

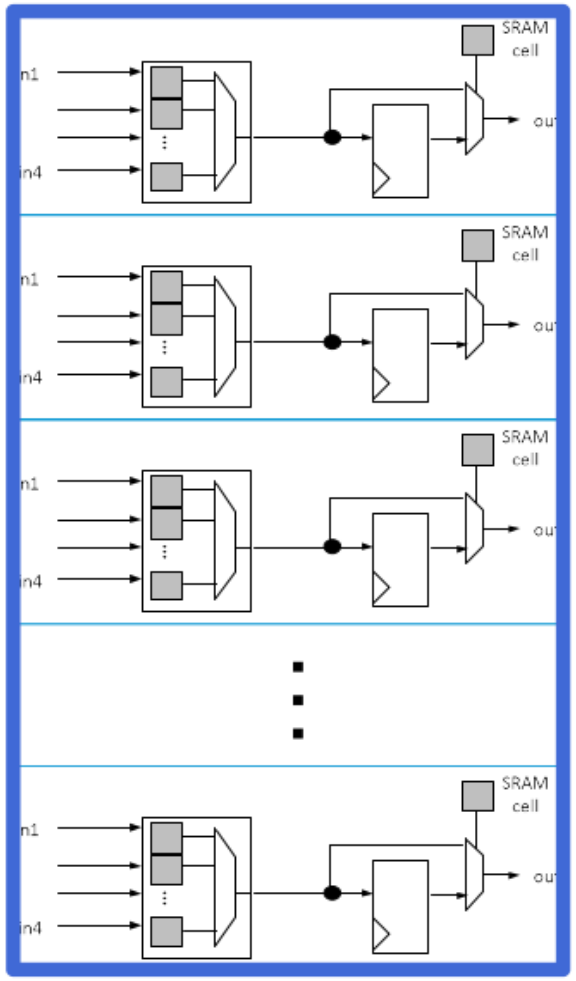


Rappel FPGA et customisation

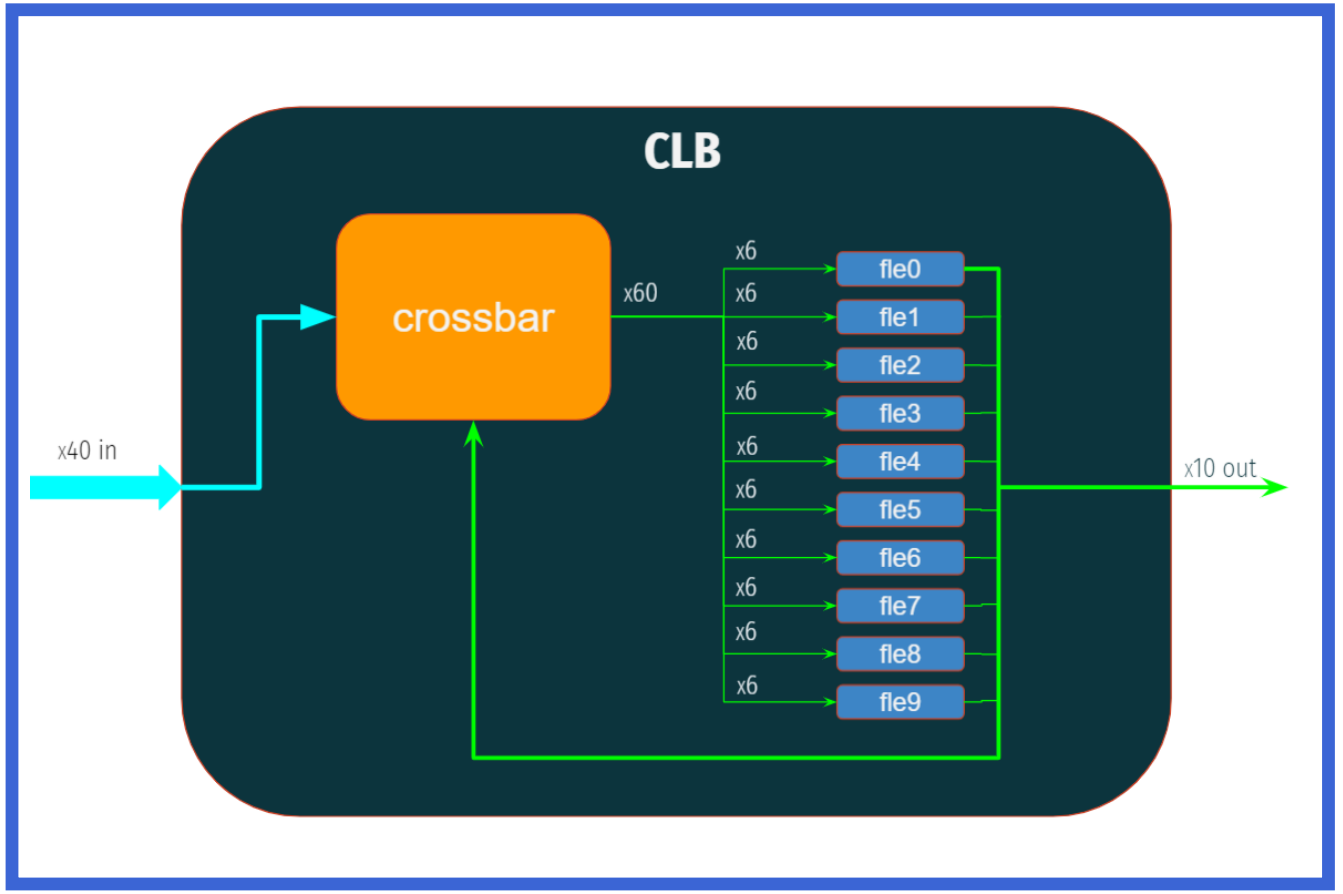
Logic Block (LB)

- Group of Logic Elements (LEs)

4-LUT



6-LUT



Description d'architecture d'un FPGA en XML

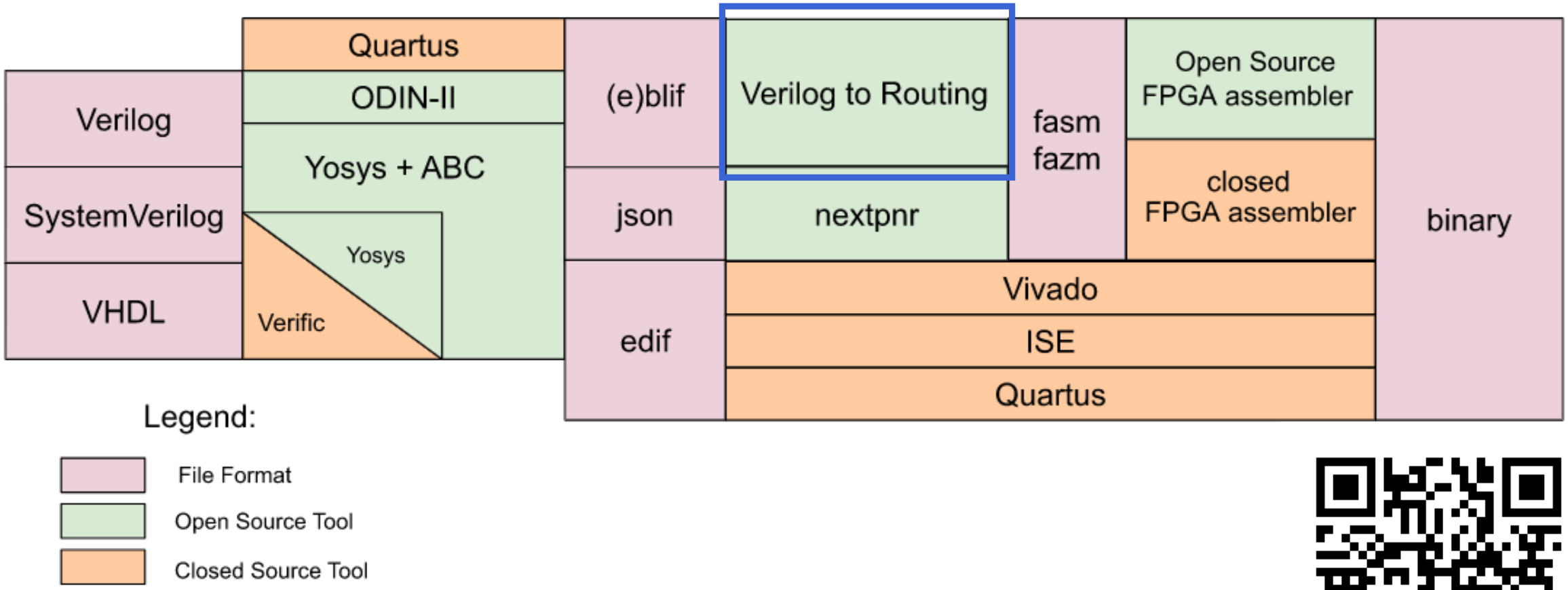
```
<architecture>

        <models>
        <layout>
        <device>
              <switch_block type="wilton" fs="3"/>
               <connection_block input_switch_name="ipin_cblock"/>

        <switchlist>
        <segmentlist>
        <complexblocklist>
              <pb_type …….>
              <interconnect>

</architecture>
```

XML (Extensible Markup Language)

https://docs.verilogtorouting.org/en/latest/arch/reference/

Vers les FPGAs commerciaux -> Flot F4PGA

F4PGA is based on VTR and targets a wide range of commercial FPGAs



https://f4pga.readthedocs.io/en/latest/

Flot F4PGA

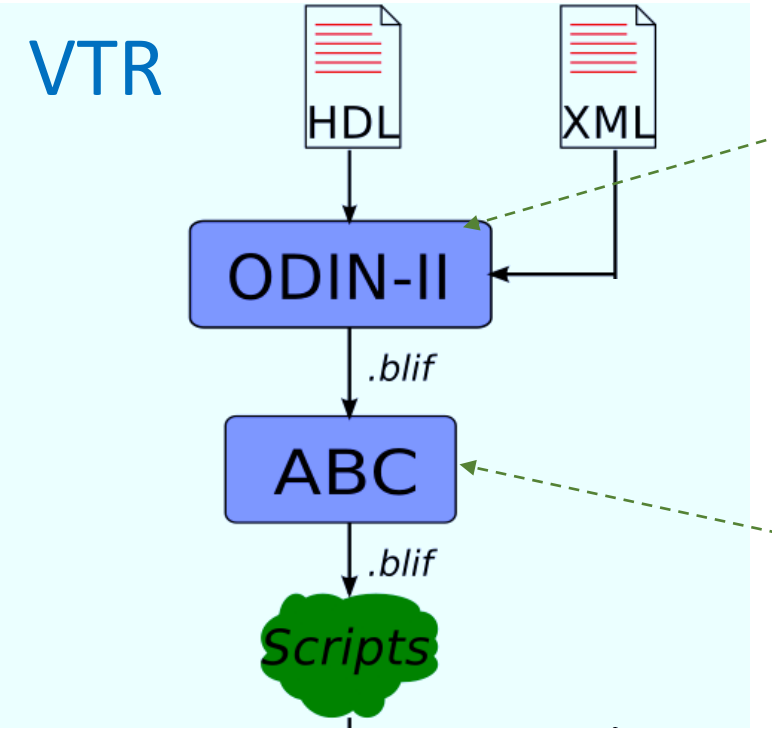
F4PGA is based on VTR and targets a wide range of commercial FPGAs

FPGA manufacturers	AMD-Xilinx	Lattice	QuickLogic
Boards	Arty A7 (XC7A35T, XC7A100T) Artix-7 (XC7A35T) Zybo (XC7Z010) NeTV2 (XC7A35T) Nexys Video Artix-7 (XC7A200T-1SBG484C)	ECP5-5G Versa (LFE5UM5G-45F) ULX3S (LFE5U-12F/-25F-45F/-85F) TinyFPGA Ex (LFE5U-85F/ LFE5UM5G-85F) iCE40-HX8K Breakout (iCE40-HX8K-B-EVN) iCEblink40LP1K Eval. Kit (iCE40LP1K-BLINK-EVN) iCEstick Eval. Kit (iCE40HX1K-STICK-EVN) DPControl iCEVision (iCE40UP5K)	QuickFeather Dev. Kit (EOS-S3)

Prise en main de VTR : Lab1

Logic synthesis

Go to folder « Lab1_synth0 »



1

```
$VTR_ROOT/ODIN_II/odin_II \
-a FPGA_architecture.xml \
-V verilog_circuit.v \
-o name.odin.blif
```

Ex: odin_II -a ./fpga_arch/EArch.xml -V ./bench/and2.v -o and2_0.blif

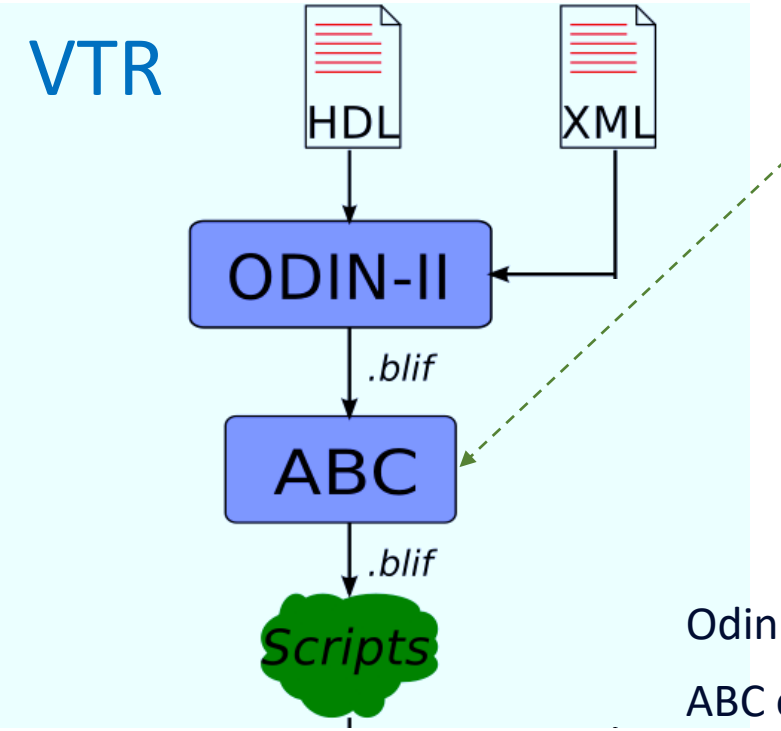
2

```
$VTR_ROOT/abc/abc \
-c 'read name.odin.blif; if -K 6; write_hie
name.odin.blif name.abc_no_clock.blif'
```

Ex: abc -c 'read and2_0.blif; if K 6; write_hie and2_0.blif and2_0noclk.blif'

Prise en main de VTR : Lab1

Logic synthesis



2

```
$VTR_ROOT/abc/abc \
-c 'read name.odin.blif; if -K 6; write_hie
name.odin.blif name.abc_no_clock.blif'
```

Ex: abc -c 'read and2_0.blif; if K 6; write_hie and2_0.blif and2_0noclk.blif'

3 Re-inserting clocks

Odin II output ->

ABC output ->

Input VPR ->

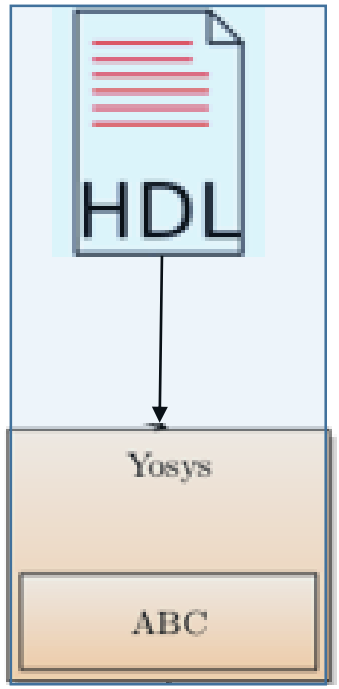
```
$VTR_ROOT/vtr_flow/scripts/restore_multicl
ock_latch.pl \
name.odin.blif \
name.abc_no_clock.blif \
name.pre-vpr.blif
```

Ex: restore_multiclock_latch.pl and2_0.blif and2_0noclk.blif and2_1.blif

Prise en main de VTR : Lab 1

Logic synthesis

Go to folder « Lab1_synth1 »



Verilog to Blif in one line using Yosys

```
${yosys_PATH}/yosys/yosys -o name.blif -S name.v
```

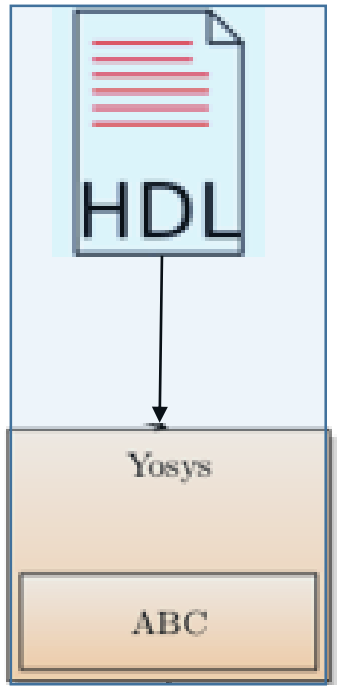
```
read_verilog name.v
hierarchy
proc; opt; memory; opt; techmap; opt
write_blif name.blif
```

Yosys:

- Can read a tcl file or « .ys » configuration file
- Can show graphically verilog HDL design

Prise en main de VTR : Lab 1

Logic synthesis



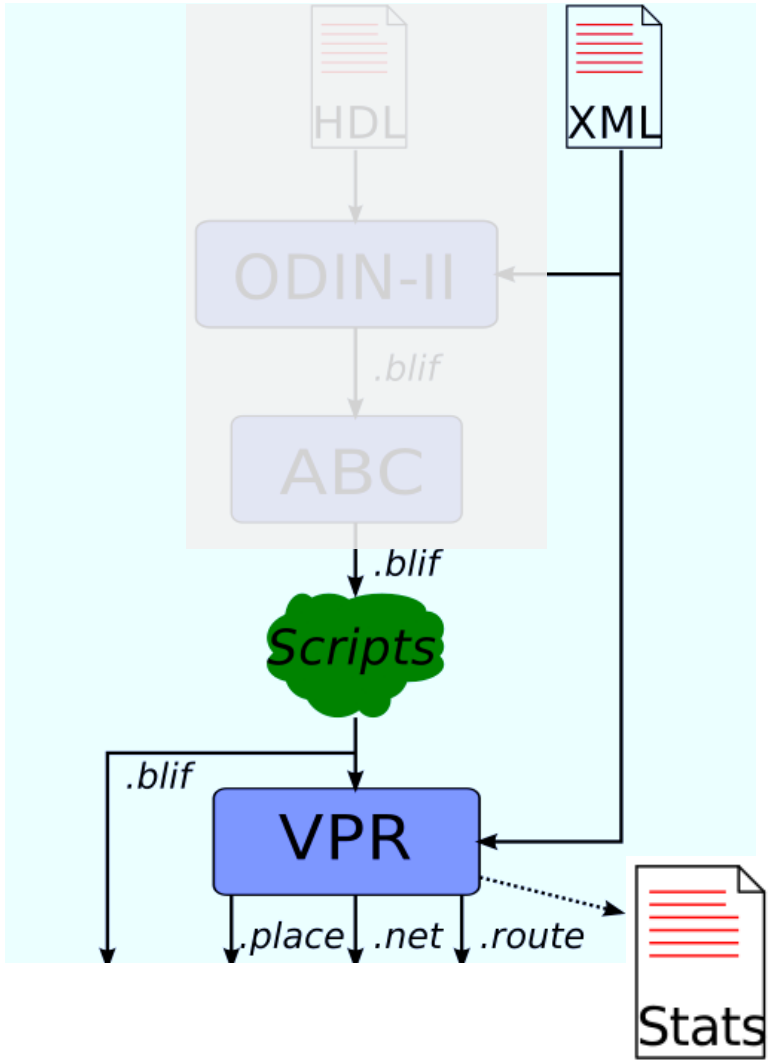
Yosys:

- Can read a tcl file or « .ys » configuration file
- Can show graphically Verilog HDL design

```
$yosys -p 'show' name.v
$yosys -p 'prep; show' name.v
$yosys -p 'synth; splitnets -ports; show' name.v
```

Prise en main de VTR : Lab 2

Pack, Place & Route



Go to folder « Lab2_vpr »

```
#Find the minimum routable channel width of my_circuit on my_arch
  vpr my_arch.xml my_circuit.blif
```

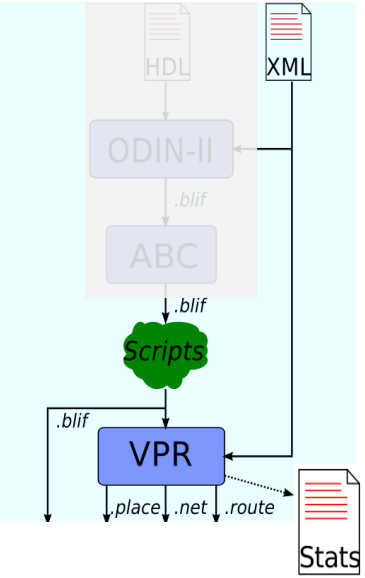
```
#Implement at a fixed channel width of 100
  vpr my_arch.xml my_circuit.blif --route_chan_width 100
```

```
  #Generate post-implementation netlist
  vpr my_arch.xml my_circuit.blif --gen_post_synthesis_netlist on

  #Write routing-resource graph to a file
  vpr my_arch.xml my_circuit.blif --write_rr_graph my_rr_graph.xml
```

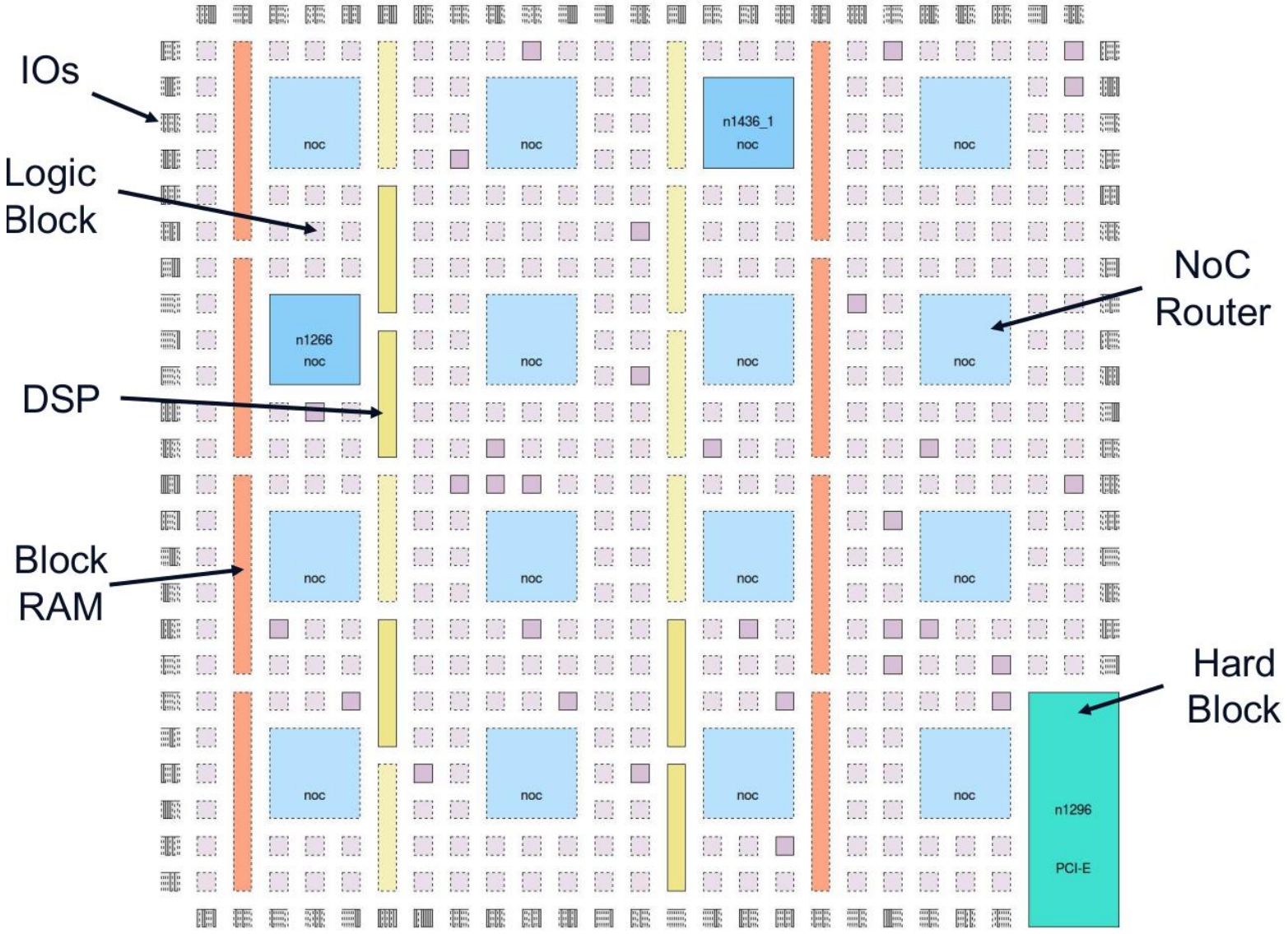
Prise en main de VTR : Lab 2

Pack, Place & Route

VPR Graphics

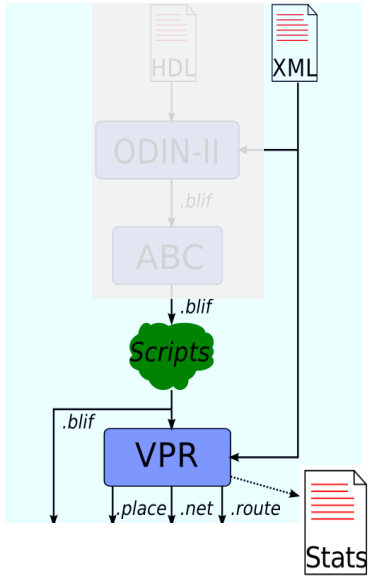



```
#Show interactive graphics
  vpr my_arch.xml my_circuit.blif --disp on
```

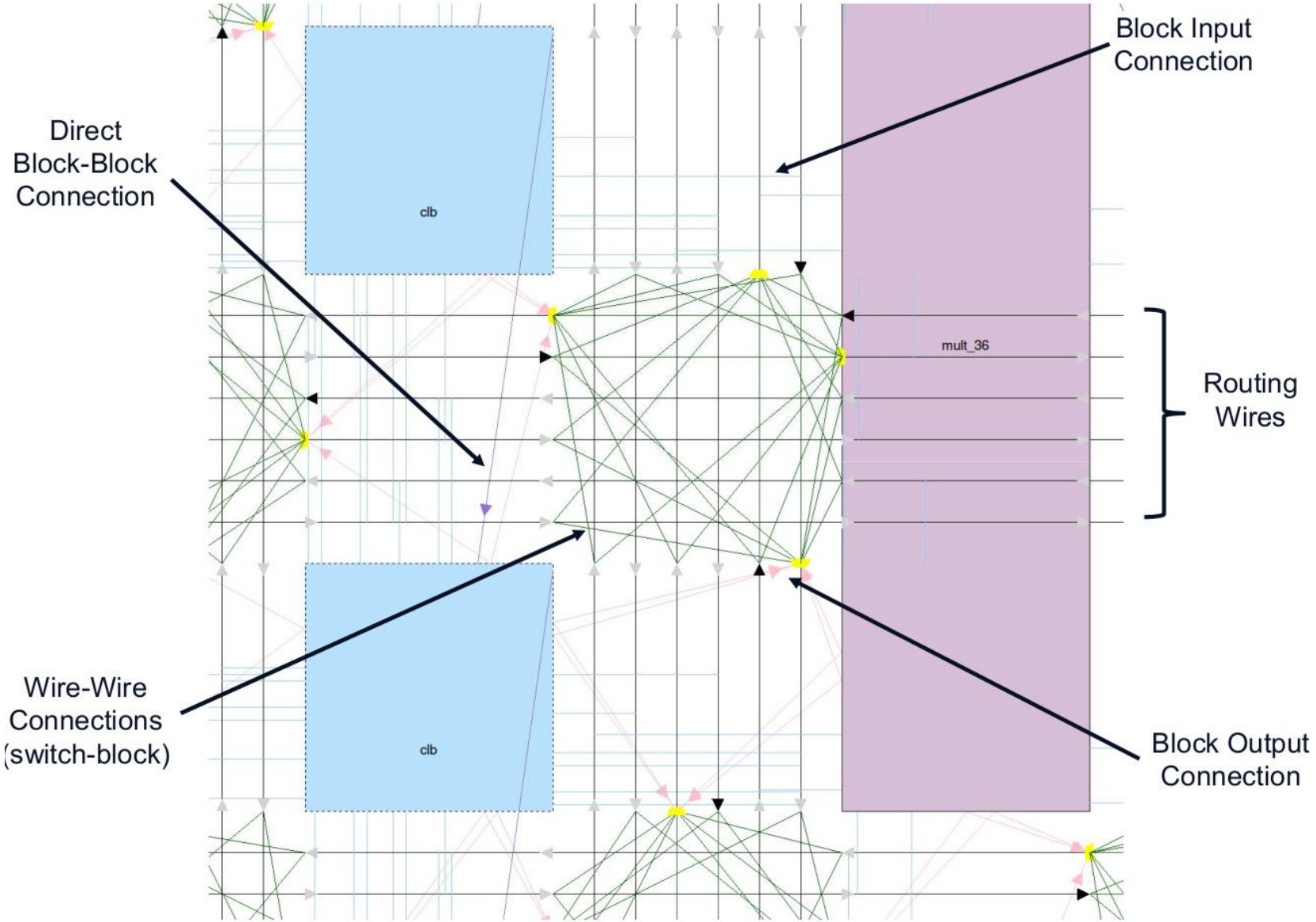


Prise en main de VTR : Lab 2

Pack, Place & Route



VPR Graphics



Prise en main de VTR : Lab 3

Verilog (HDL circuit) to routing (Pack, Place & Route) flow in one line

```
$VTR_ROOT/vtr_flow/scripts/run_vtr_flow.pl \
name.v\
Name_FPGA_archi.xml \
-options (optional)
-- vpr_options (optional)
```

Create a folder <entire_flow>

Example:

- Vpr_options : --route_chan_width N; --place
- Options: -temp_dir <Path> (optional)
  -starting_stage (odin, abc, scripts, vpr)
  -ending_stage (odin, abc, scripts, vpr)
  -power (power estimation), etc

Prise en main de VTR : FASM

Generate FPGA Assembly (FASM)

FASM file represent the design at a level detailed enough to allow generation of a bitstream to program a device

```
$VTR_ROOT/build/utils/fasm/genfasm \
FPGA_Archi.xml\
Circuit.blif \
-- options (optional)
```

Same like VPR command

Example:

- Option : --route_chan_width N --pack –place --route
  --read_rr_graph my_rr_graph

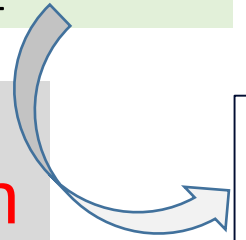
Prise en main de VTR : Lab 3

Run multiple tasks: Combination of Verilog circuits and FPGA architectures

```
$VTR_ROOT/vtr_flow/scripts/run_vtr_task.pl  <task_name1>  <task_name2>
```

or

```
$VTR_ROOT/vtr_flow/scripts/run_vtr_task.pl –l <task_list_file>
```



```
<task_name1>
<task_name2>
<task_name3>
...
```

Task Example:

```
# Path to directory of circuits to use
circuits_dir=benchmarks/verilog
# Path to directory of architectures to use
archs_dir=arch/timing
# Add circuits to list to sweep
circuit_list_add=ch_intrinsics.v
circuit_list_add=diffeq1.v
# Add architectures to list to sweep
arch_list_add=k6_N10_memSize16384_memData64_stratix4_based_timing_
sparse.xml
# Parse info and how to parse
parse_file=vpr_standard.txt
```

Must be done in VTR tasks location

Lab 4 : Mini challenge

Map the circuit « sha.v » on the given FPGA

Go to folder « Lab4_challenge »

Specifications :

- Use 70% of FPGA logic resources, especially CLBs
- FPGA size MUST be fixed (see the architecture description file)
- Modify route_chan_width value, simulate and display result in VPR graphics

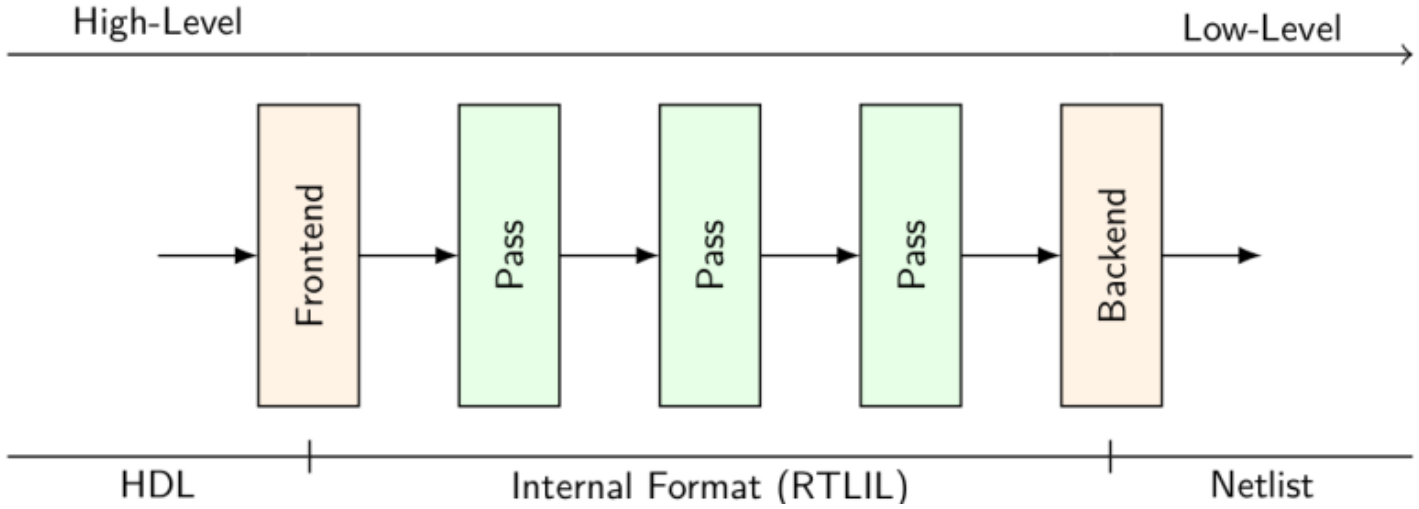
Je vous remercie !

Questions?

Email: sylvain.takougang@lip6.fr

Yosys Data- and Control-Flow

- Three types of commands:
  - **Frontends**: read input files (Verilog)
  - **Passes**: perform transformation on the design
  - **Backends**: write the design to a file



http://www.clifford.at/yosys/files/yosys_presentation.pdf

Annexe 2

Program Components and Data Formats