

## Dynamic Partial Reconfiguration

Julio Dondo Gazzano

Jdondo@unsl.edu.ar

Juliodaniel.dondo@uclm.es

# FPGA Partial reconfiguration

Partial Reconfiguration (PR) is the ability to time multiplex hardware dynamically on a single FPGA

PARTIAL RECONFIGURATION means:

- ❑ The modification of a **reconfigurable region** in the FPGA fully configured by loading a partial configuration file (Partial Bitstream).

DYNAMICALLY means:

- ❑ These **Partial BIT files** can be downloaded when the FPGA is in running mode without compromising the integrity of the applications running on the rest of the FPGA

# FPGA Partial reconfiguration

## Benefits and uses

- Reduce power consumption (use it when you need it)
  - Less resources needed (again, less power consumption)
  - Shorter reconfiguration time
  - HW components interchangeability
- 
- Adaptive design (Reduce implementation costs)
  - Encryption key management (security)
  - Self monitoring service (Having a DUT change test pattern on the fly)
  - Reconfigurable grids or clusters (HPC)

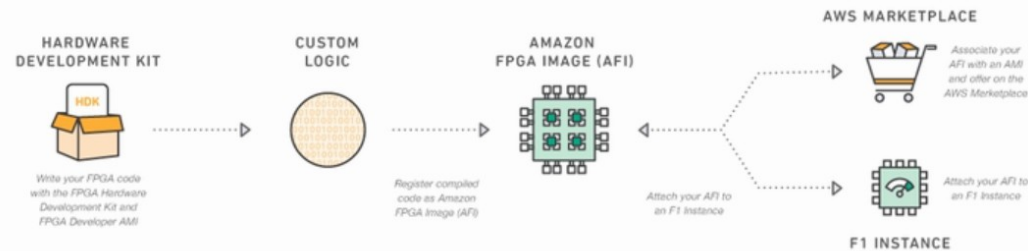
# FPGA Partial reconfiguration

## Success with Partial Reconfiguration

*Computing Acceleration in the Cloud*

### ➤ Amazon Web Services EC2 F1 Instances

- Powered by the Xilinx Reconfigurable Acceleration Stack on UltraScale+
- Deploy acceleration kernels in the cloud across many F1 instances



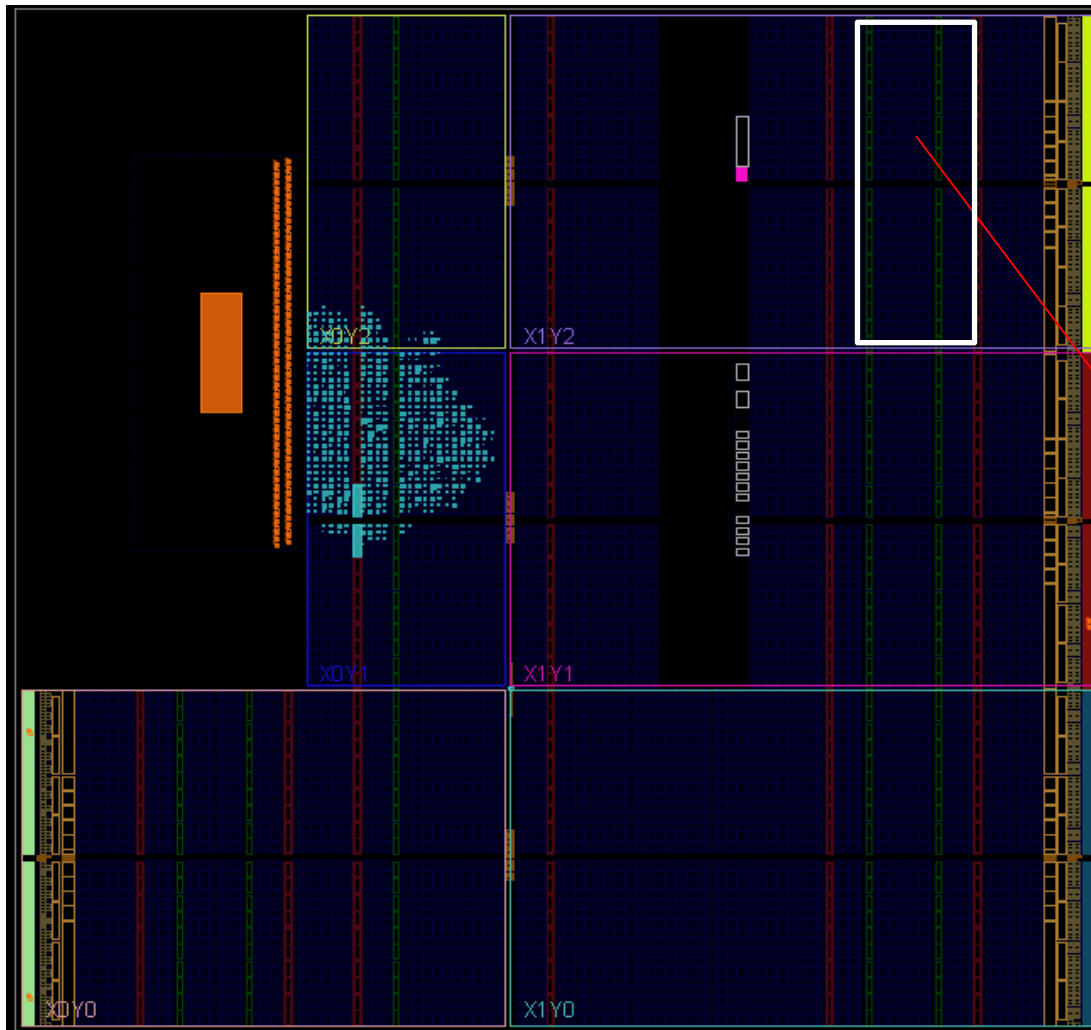
# FPGA Partial reconfiguration

Ok, everything is fantastic, but

How easy is to design PR systems?

Let's define some terms first.

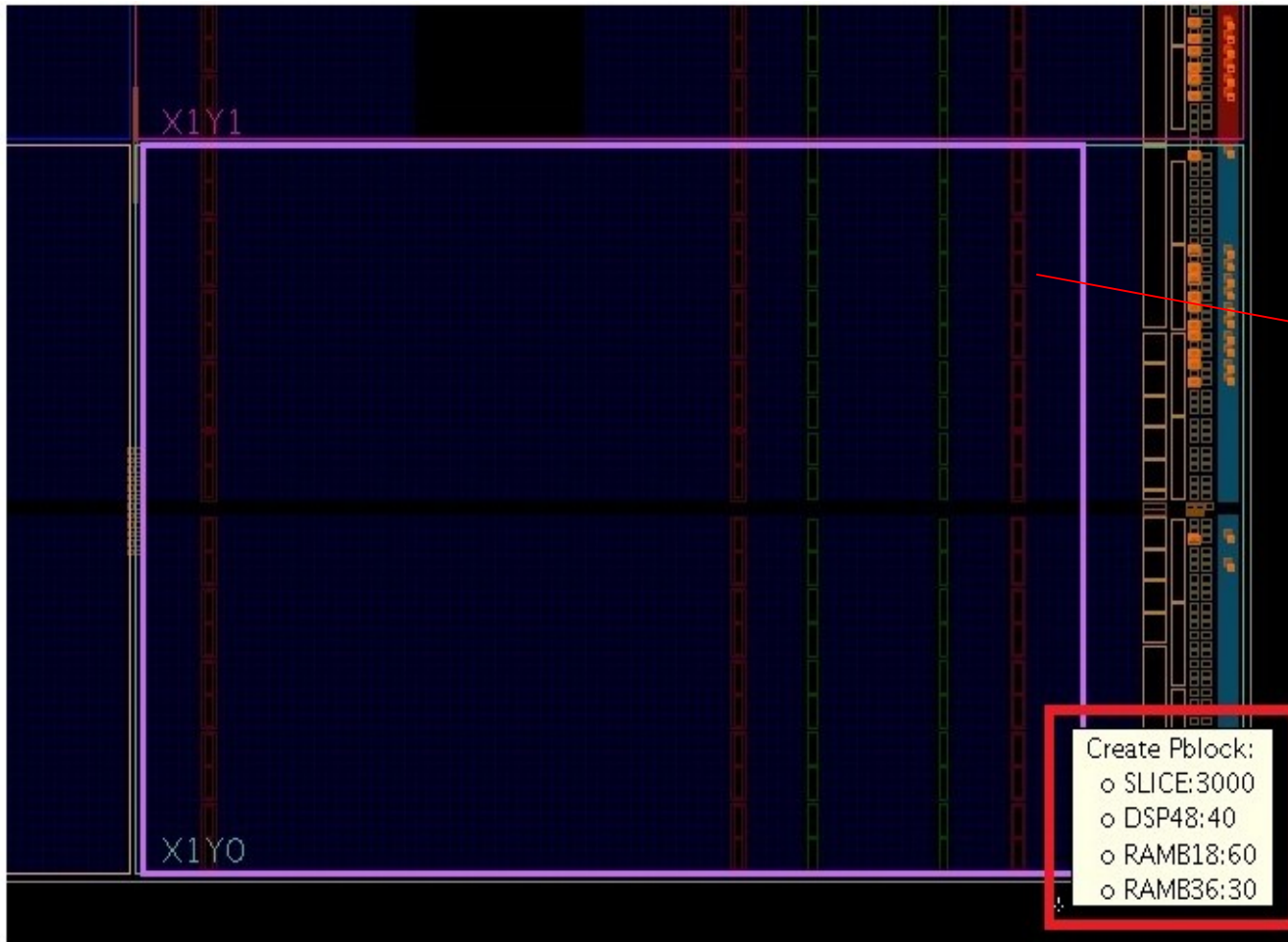
# FPGA Partial reconfiguration



## Reconfigurable Partition

A Partition is a logical section of the design, user-defined at a hierarchical boundary, to be considered for design reuse.

# FPGA Partial reconfiguration



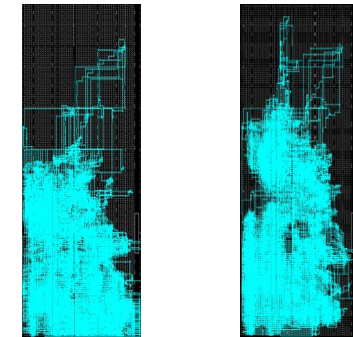
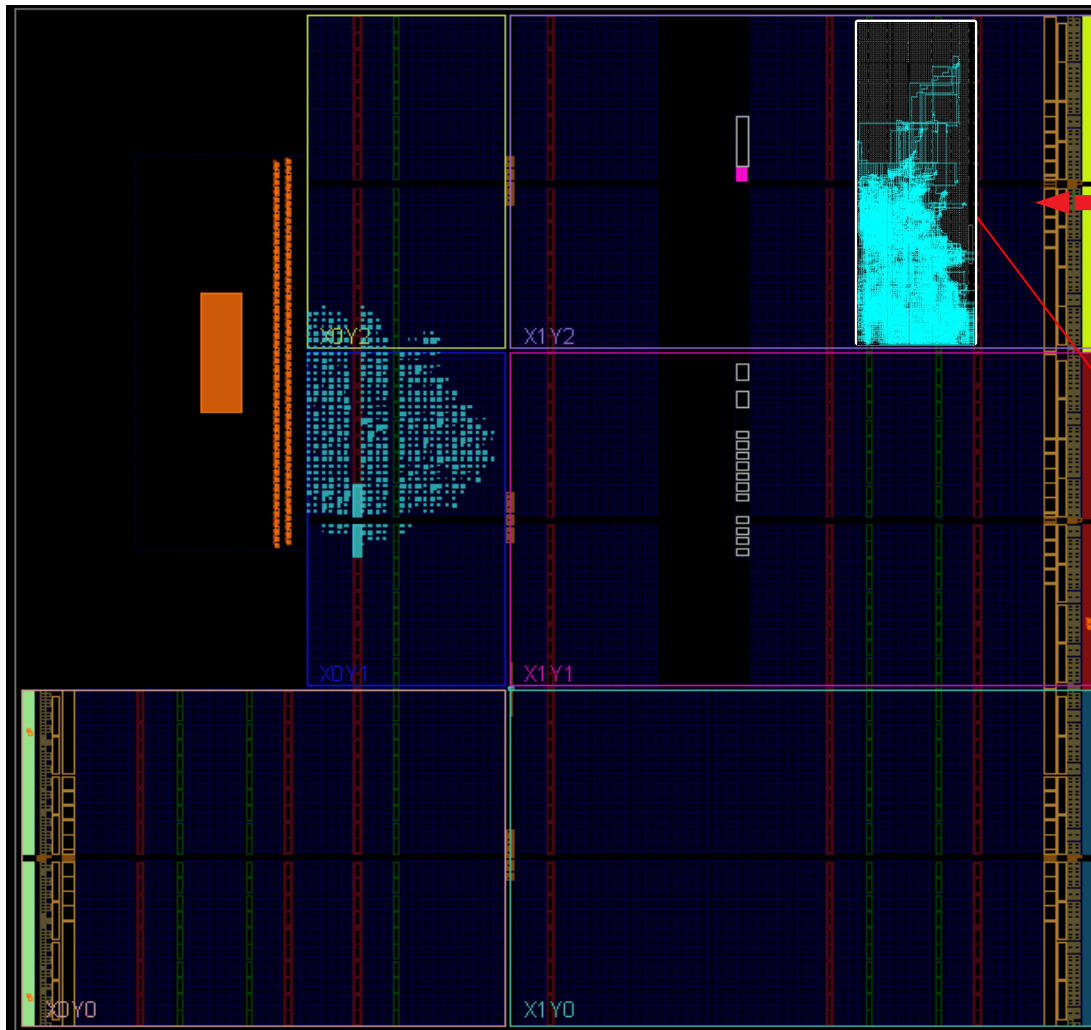
**Reconfigurable Partition**  
Floorplanning

Draw PBlock

In a 7 series device a reconfigurable frame is 50 CLBs high by 1 CLB wide.

It is good practice to frame-align the Reconfigurable Partition to achieve best place and route results.

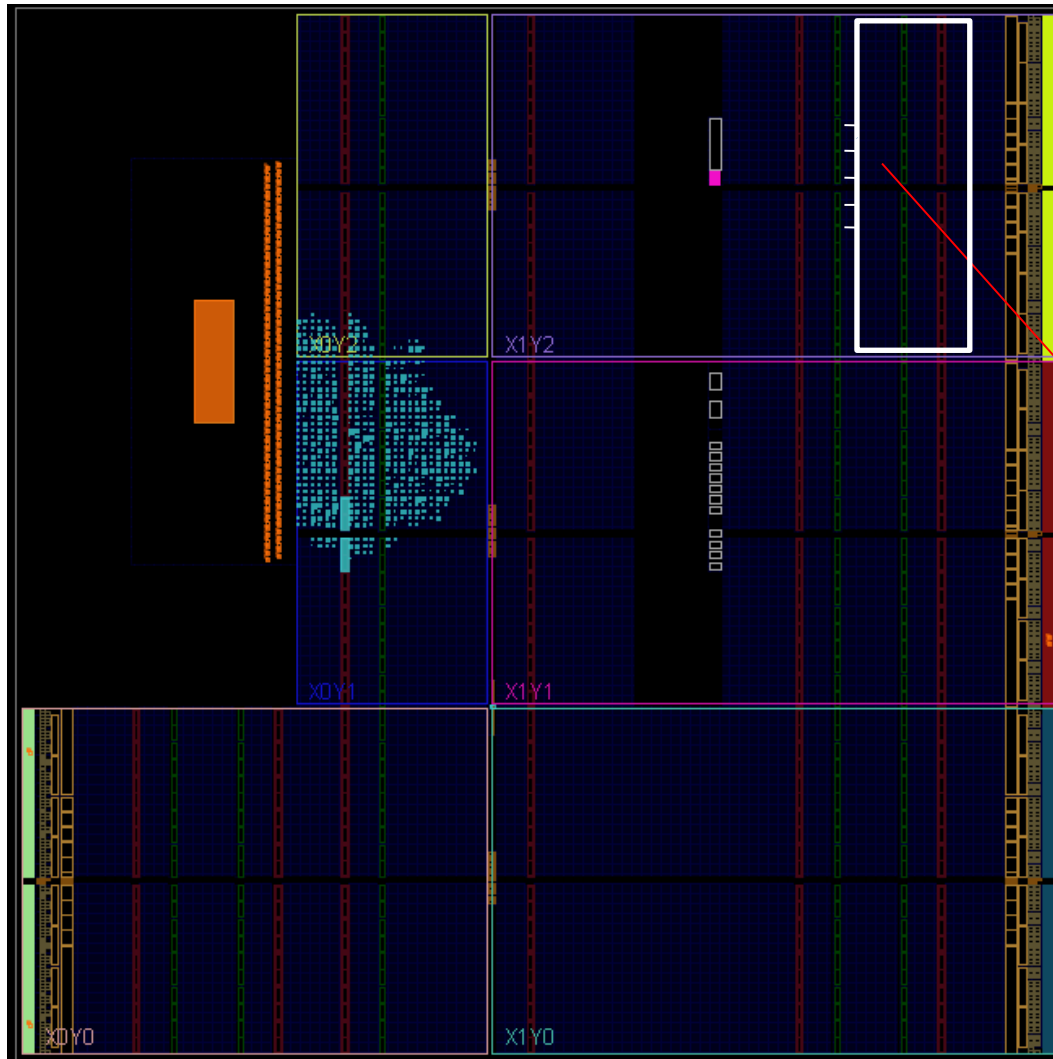
# FPGA Partial reconfiguration



**Reconfigurable Module (RM)**  
RM is the netlist or HDL description that is implemented within a reconfigurable partition.



# FPGA Partial reconfiguration



**Partition Pin**  
Partition pins are the logical and physical connection between static logic and reconfigurable logic

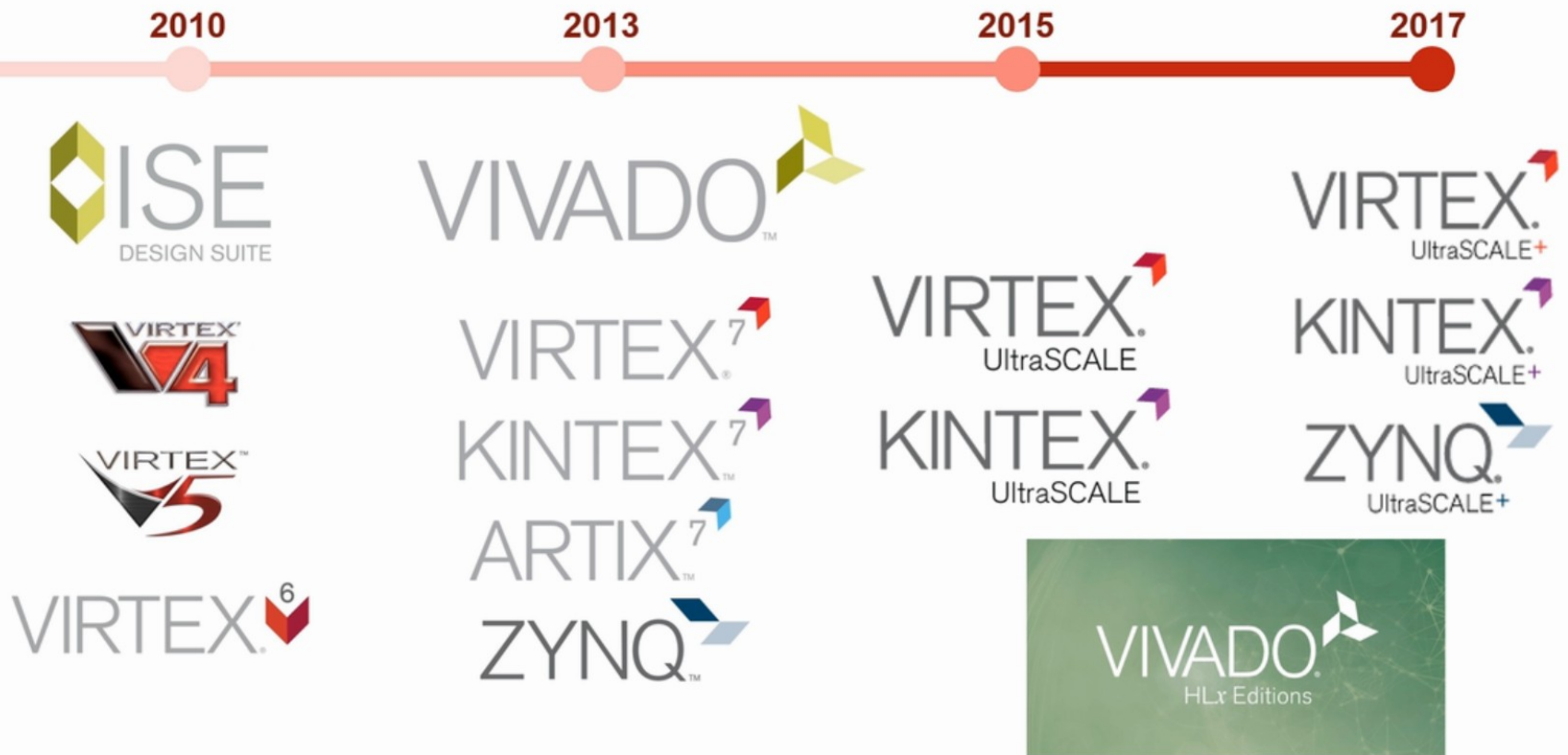
# FPGA Partial reconfiguration

- Partial Reconfiguration (PR) is an expert flow within the Vivado Design Suite
- PR is supported through Tcl or by command line only; there is no project support at this time.
- Device support in Vivado Design Suite 2018.3:
  - 7 Series: All Artix-7, Kintex-7, Virtex-7, and Zynq-7000 SoC devices, Ultrascale, Ultrascale+.

For partial reconfiguration in Virtex-6, Virtex-5 and Virtex-4 devices use ISE Design Suite

# FPGA Partial reconfiguration

## Partial Reconfiguration Technology Timeline



# FPGA Partial reconfiguration

However, besides tool's issues, there are several aspects to address during design

Design considerations

Componing your design

Testing your design

Vivado Design Flow

Managing Partial Reconfiguration Process

Let's go into details

# FPGA Partial reconfiguration Design Considerations

## Where to start?

### Defining which part of your design will be static

Managing one more degree of freedom.

- Hw/Sw partitioning (modeling and profiling)
- Static/dynamic partitioning

**Static Design, part of the design that remains the same during design life cycle.**

**Dynamic design, part of the design that can change many times during design life cycle.**

# FPGA Partial reconfiguration Design Considerations

## Designing Reconfigurable Modules

- All modules sharing the same reconfigurable zone must have the **same interface** to the static part of the design. These interfaces must have the same name, width and direction,

# FPGA Partial reconfiguration Design Considerations

## Designing Reconfigurable Modules

- All modules sharing the same reconfigurable zone must have the **same interface** to the static part of the design. These interfaces must have the same name, width and direction,
- The name of the module must be the same in all instances. Then, to identify each reconfigurable module from the others it is necessary to create different folders, each one for the corresponding reconfigurable module.

# FPGA Partial reconfiguration Design Considerations

## Designing Reconfigurable Modules

- All modules sharing the same reconfigurable zone must have the **same interface** to the static part of the design. These interfaces must have the same name, width and direction,
- The name of the module must be the same in all instances. Then, to identify each reconfigurable module from the others it is necessary to create different folders, each one for the corresponding reconfigurable module.
- Outputs of the reconfigurable modules must be ignored during partial reconfiguration process.
  - Vivado design tools provides with a PR Decoupler Ip that allows users to insert Muxes to decouple custom interfaces, AXI-Lite and AXI4-Stream interfaces.
  - You can do also by your own adding registers or muxes to the outputs of reconfigurable modules on the static side of the interfaces.



# FPGA Partial reconfiguration Design Considerations

## Designing Reconfigurable Modules

Other aspects to be considered

To start and to Stop RM execution

- Initial State (Reset After Reconfiguration)
- Final State (Consistency checkpoint for RM replacement)
- Persistence (Management of internal data)
- Take into account the technology used
  - Ultrascale need that RM must be cleared before reconfiguration

# FPGA Partial reconfiguration Design Considerations

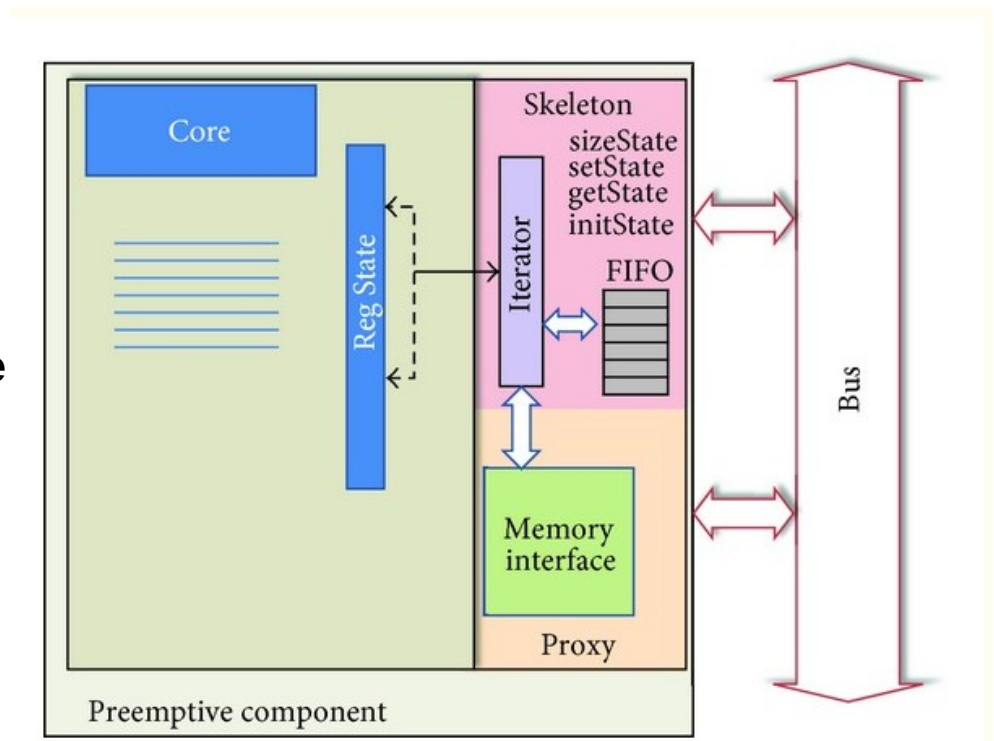
## Designing Reconfigurable Modules

Other aspects to be considered

Persistence (Management of Internal data)

$$T_{\text{stop\_RM}} = T_{\text{stop\_meth}} + T_{\text{ack}} + T_{\text{state\_trf}}$$

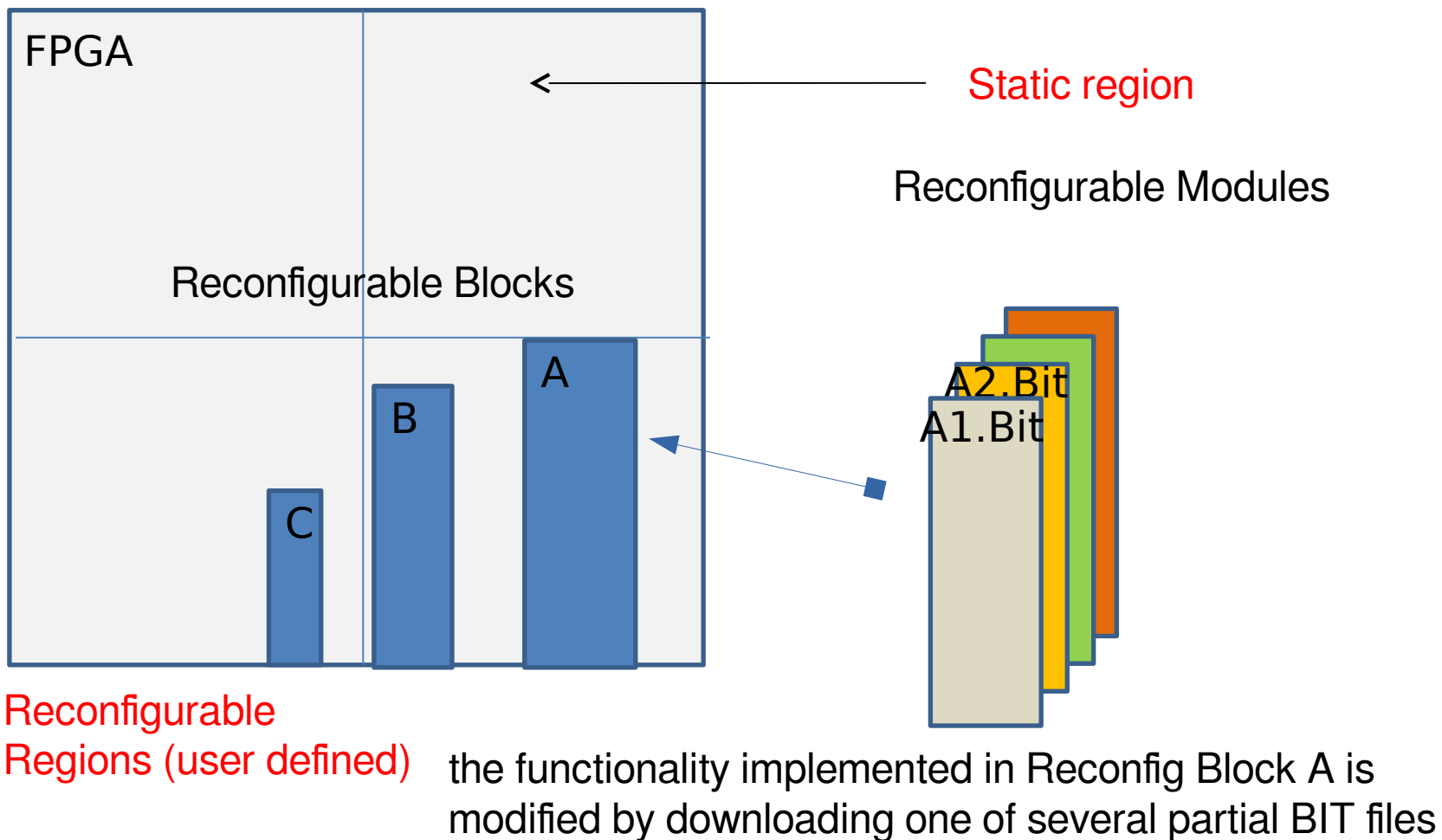
$$T_{\text{state\_trf}} = \frac{StSize}{Chwidth} * Ncycles + init\_time$$



# FPGA Partial reconfiguration Design Considerations

Once you have designed reconfigurable modules and synthesised them it is necessary **to define a reconfigurable region** with enough reconfigurable resources ( CLB, DSP, blockRAMs) to place the biggest reconfigurable module.

# FPGA Partial reconfiguration Design Considerations



# FPGA Partial reconfiguration Design Considerations

## How to define these Reconfigurable regions?

### Depending on the components

Size of the components

Resources needed (CLB, DSP, BlockRAMs)

Interfaces between static and dynamic parts

- Same interface with the static part for all dynamic designs
- Modules with different interfaces (adapters)

# FPGA Partial reconfiguration Design Considerations

## How to define these Reconfigurable regions?

### Depending on the technology

For 7 series, it is recommended to vertically align Pblocks with frame/clock region boundaries.

- A Reconfigurable Frame is the smallest size physical region that can be reconfigured.
- In a 7 series device a reconfigurable frame is 50 CLBs high by 1 CLB wide.
- Clocks including BUFG, BUFR, MMCM, PLL, I/O, components, transceivers and components that can modify the architecture such as BSCAN, XADC, ICAP, STARTUP can not be configured and must be placed in static region.

# FPGA Partial reconfiguration Design Considerations

## How to define these Reconfigurable regions?

### Depending on the technology

For UltraScale and UltraScale+ devices, the list of reconfigurable component types is more extensive:

- CLB, block RAM, and DSP component types as well as routing resources
- Clocks and clock modifying logic, including BUFG, MMCM, PLL, and similar components
- I/O and I/O related components (ISERDES, OSERDES, IDELAYCTRL)
- Serial transceivers (MGTs) and related components
- PCIe, CMAC, Interlaken, and SYSMON blocks

## Componing your design



# FPGA Partial reconfiguration

## Componing your design

- At top level you will have the static design plus a black box for each reconfigurable region.
- Create a design with one of the RM as it was all static (non PR flow) to evaluate functionality, performance, etc.
- Repeat with each one of the RM
- Consider evaluate each posible configuration when managing several Reconfigurable Regions

## Testing your design

# FPGA Partial reconfiguration

## Testing your design

Each reconfigurable module must be tested in order to meet with the performance and timing constraints of the whole design. A static timing analysis must be done for each reconfigurable module in the overall design to verify that all constraints are met.

# FPGA Partial reconfiguration

## Testing your design

Each reconfigurable module must be tested in order to meet with the performance and timing constraints of the whole design. A static timing analysis must be done for each reconfigurable module in the overall design to verify that all constraints are met.

Once we have designed all reconfigurable modules, it is necessary to create static designs with each one of the modules for testing functionality, timing and physical constraints of the design. That means you have many flat designs as reconfigurable modules you have.

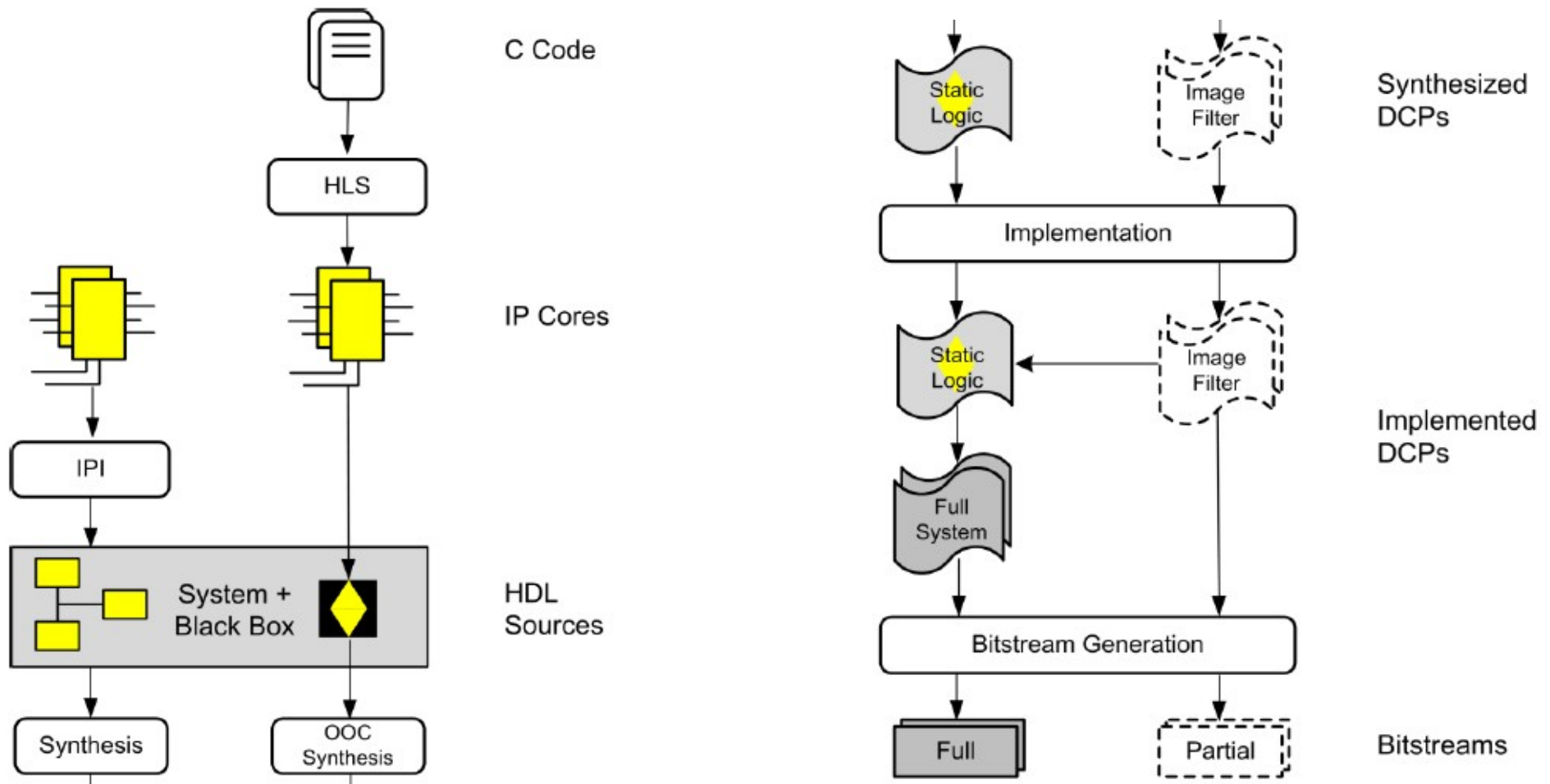
# FPGA Partial reconfiguration

Having all these stuffs already defined, let's see the

Partial Reconfiguration Design Flow

# FPGA Partial reconfiguration Vivado Design Flow (TCL-based)

## Partial reconfiguration design flow



Source: XAPP1231 (v1.1) March 20, Xilinx 2015

X14559

# FPGA Partial reconfiguration Vivado Design Flow (TCL-based)

## *Synthesis*

### *Synthesizing the Top Level*

You must have a top-level netlist with a black box for each Reconfigurable Partition (RP).

This requires the top-level synthesis to have module/entity declarations for the partitioned instances, but no logic; the module is empty

### *Synthesizing Reconfigurable Modules*

Each Reconfigurable Module must be instantiated in the same black box in the static design, so the different versions must have identical interfaces.

Each module (including Static) needs to be synthesized bottom-up so that a netlist or checkpoint exists for static and each Reconfigurable Module.

# FPGA Partial reconfiguration Vivado Design Flow (TCL-based)

## ***Implementation (I)***

***Create physical constraints (Pblocks) to define the Reconfigurable regions.***

***Set the HD.RECONFIGURABLE property on each Reconfigurable Partition.***

***Implement a complete design (static and one Reconfigurable Module per Reconfigurable Partition) in context.***

***Save a design checkpoint for the full routed design.***



# FPGA Partial reconfiguration Vivado Design Flow (TCL-based)

## *Implementation (II)*

*Remove Reconfigurable Modules from this design and save a static-only design checkpoint.*

*Lock the static placement and routing.*

*Add new Reconfigurable Modules to the static design and implement this new configuration, saving a checkpoint for the full routed design.*

*Repeat last Step until all Reconfigurable Modules are implemented.*

# FPGA Partial reconfiguration Vivado Design Flow (TCL-based)

## **Verification**

*Run a verification utility (`pr_verify`) on all configurations.  
Remove Reconfigurable Modules from this design and save a static-only design checkpoint.*

## **Generate Bitstreams.**

*Create bitstreams for each configuration, and partial bitstreams*

# FPGA Partial reconfiguration

## Dynamic Reconfiguration Management

How to manage the reconfiguration process?

Xilinx provides different configuration modes depending on the technology

# FPGA Partial reconfiguration Dynamic Reconfiguration Management

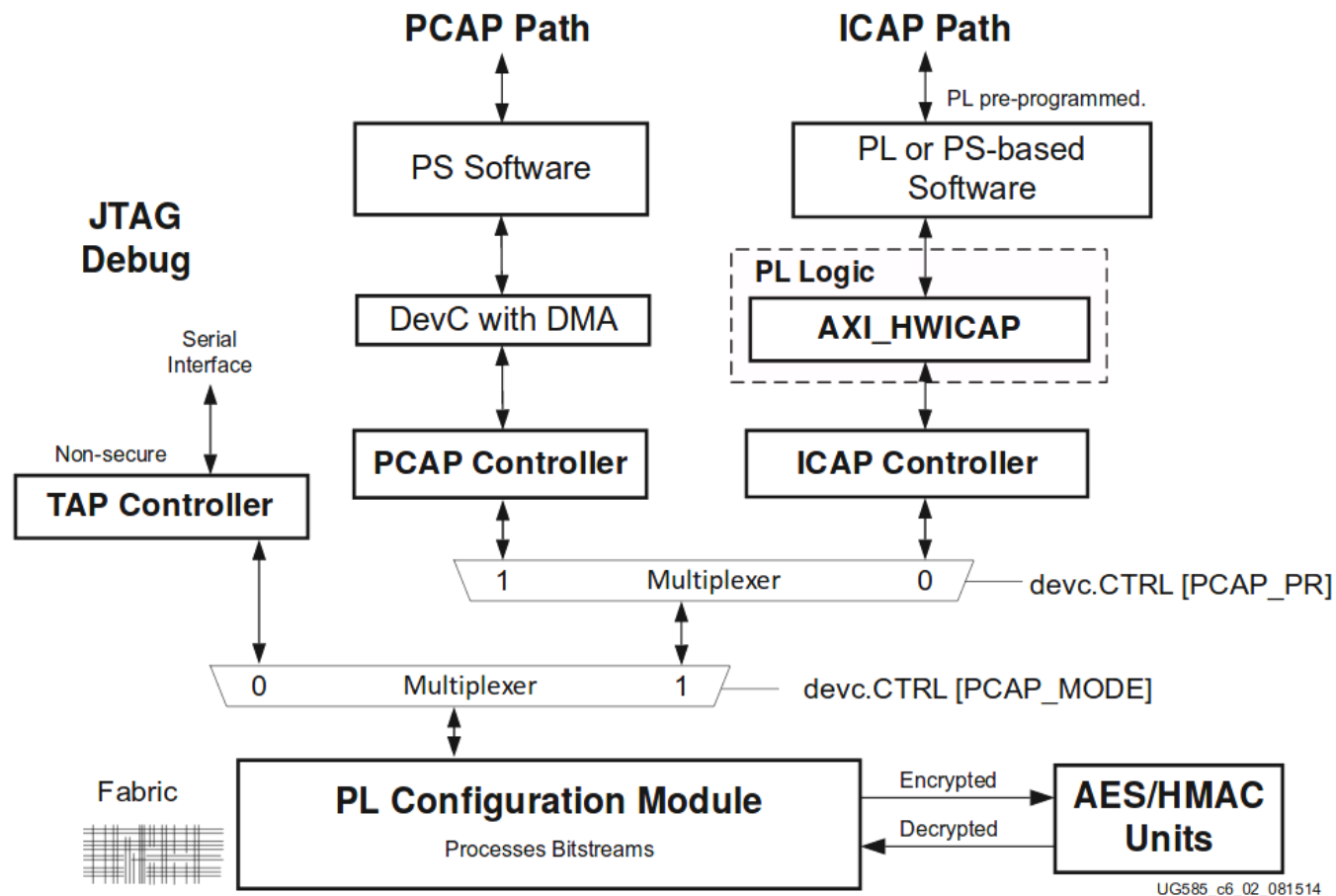
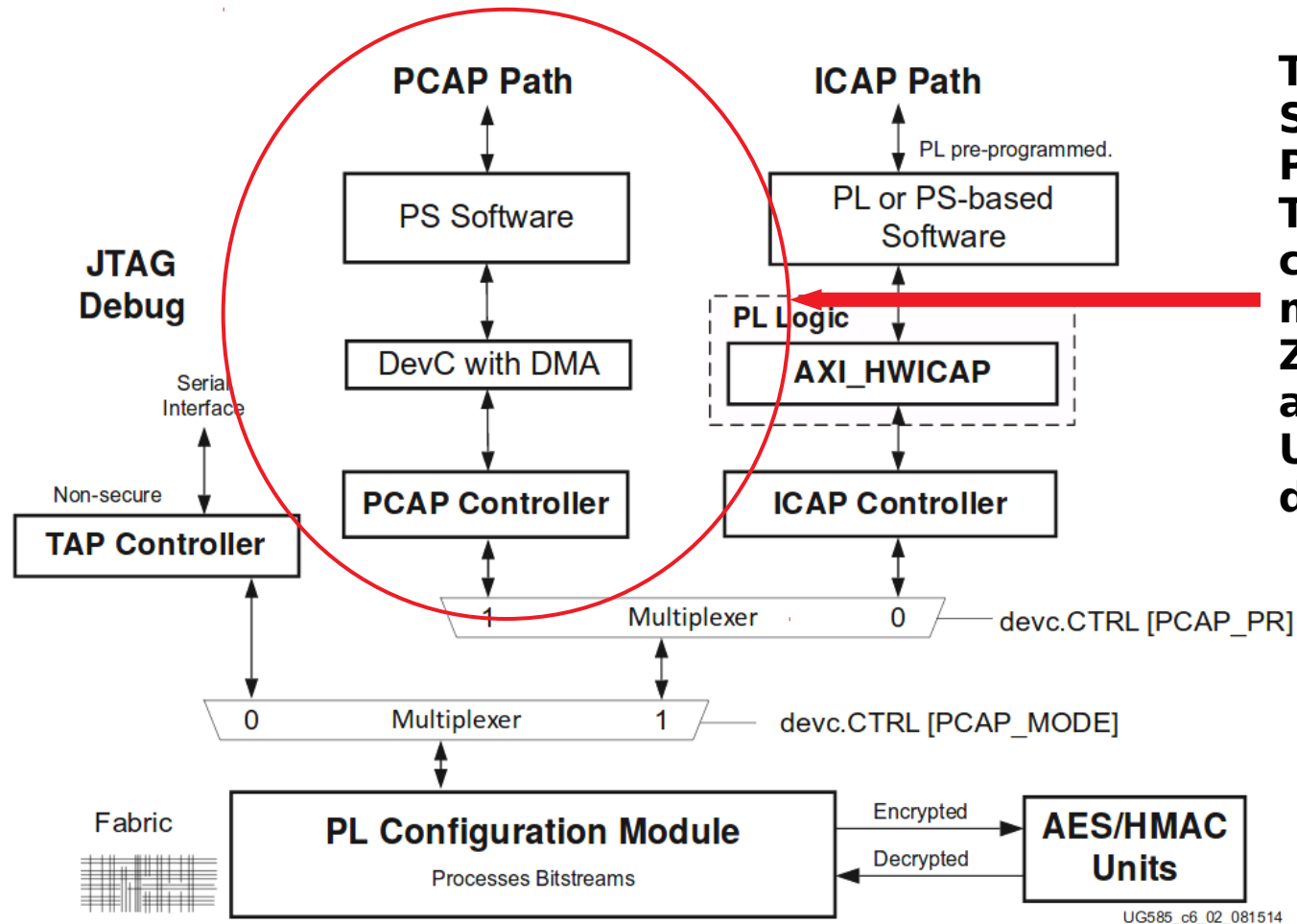


Figure 6-2: PL Configuration Paths

Source: [https://www.xilinx.com/support/documentation/user\\_guides/ug585-Zynq-7000-TRM.pdf](https://www.xilinx.com/support/documentation/user_guides/ug585-Zynq-7000-TRM.pdf)

# FPGA Partial reconfiguration Dynamic Reconfiguration Management



Through dedicated Software in the PS.  
The primary configuration mechanism for Zynq-7000 SoC and Zynq UltraScale+ designs

Figure 6-2: PL Configuration Paths

Source: [https://www.xilinx.com/support/documentation/user\\_guides/ug585-Zynq-7000-TRM.pdf](https://www.xilinx.com/support/documentation/user_guides/ug585-Zynq-7000-TRM.pdf)

# FPGA Partial reconfiguration Dynamic Reconfiguration Management

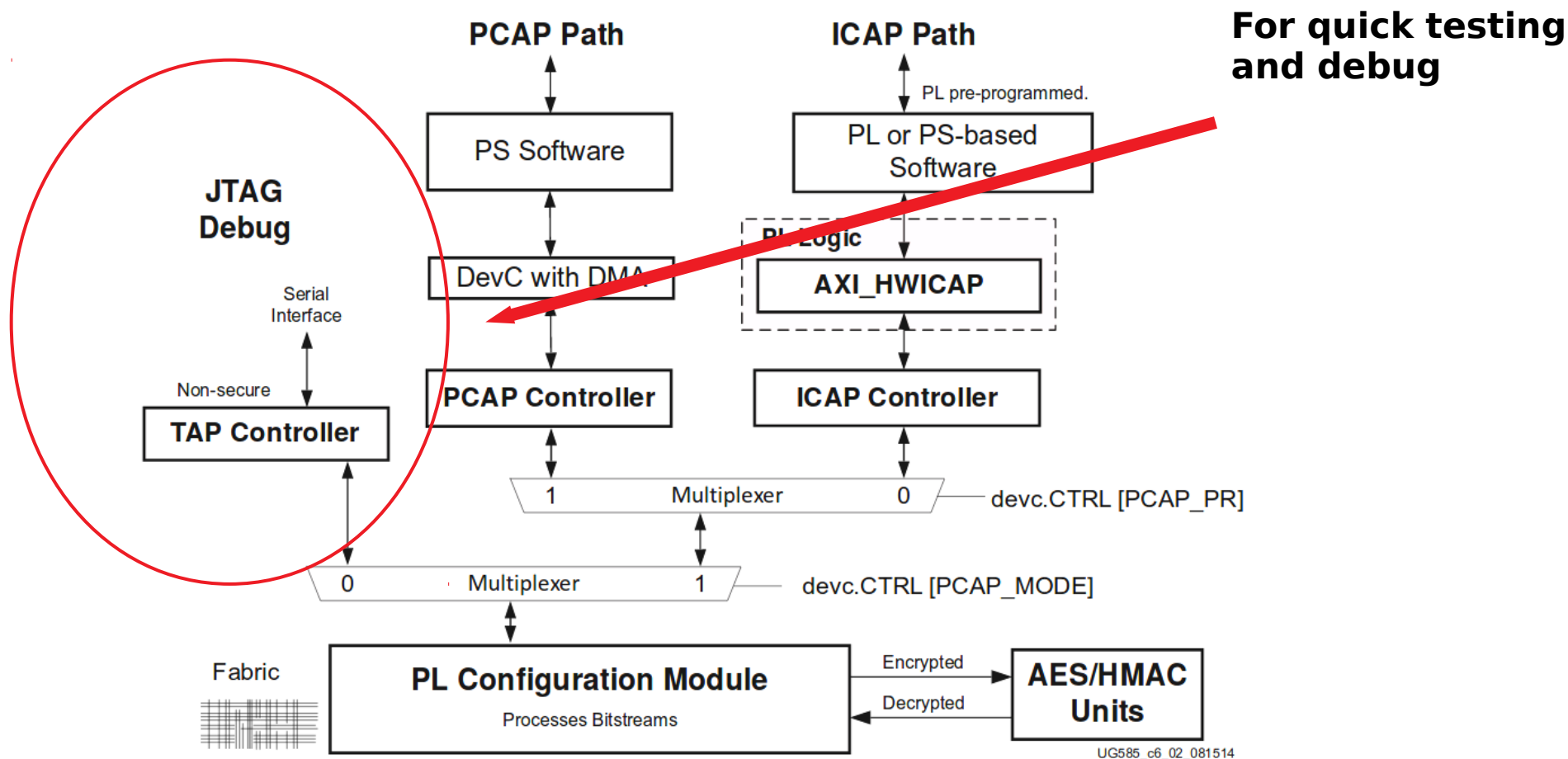


Figure 6-2: PL Configuration Paths

Source: [https://www.xilinx.com/support/documentation/user\\_guides/ug585-Zynq-7000-TRM.pdf](https://www.xilinx.com/support/documentation/user_guides/ug585-Zynq-7000-TRM.pdf)

# FPGA Partial reconfiguration Dynamic Reconfiguration Management

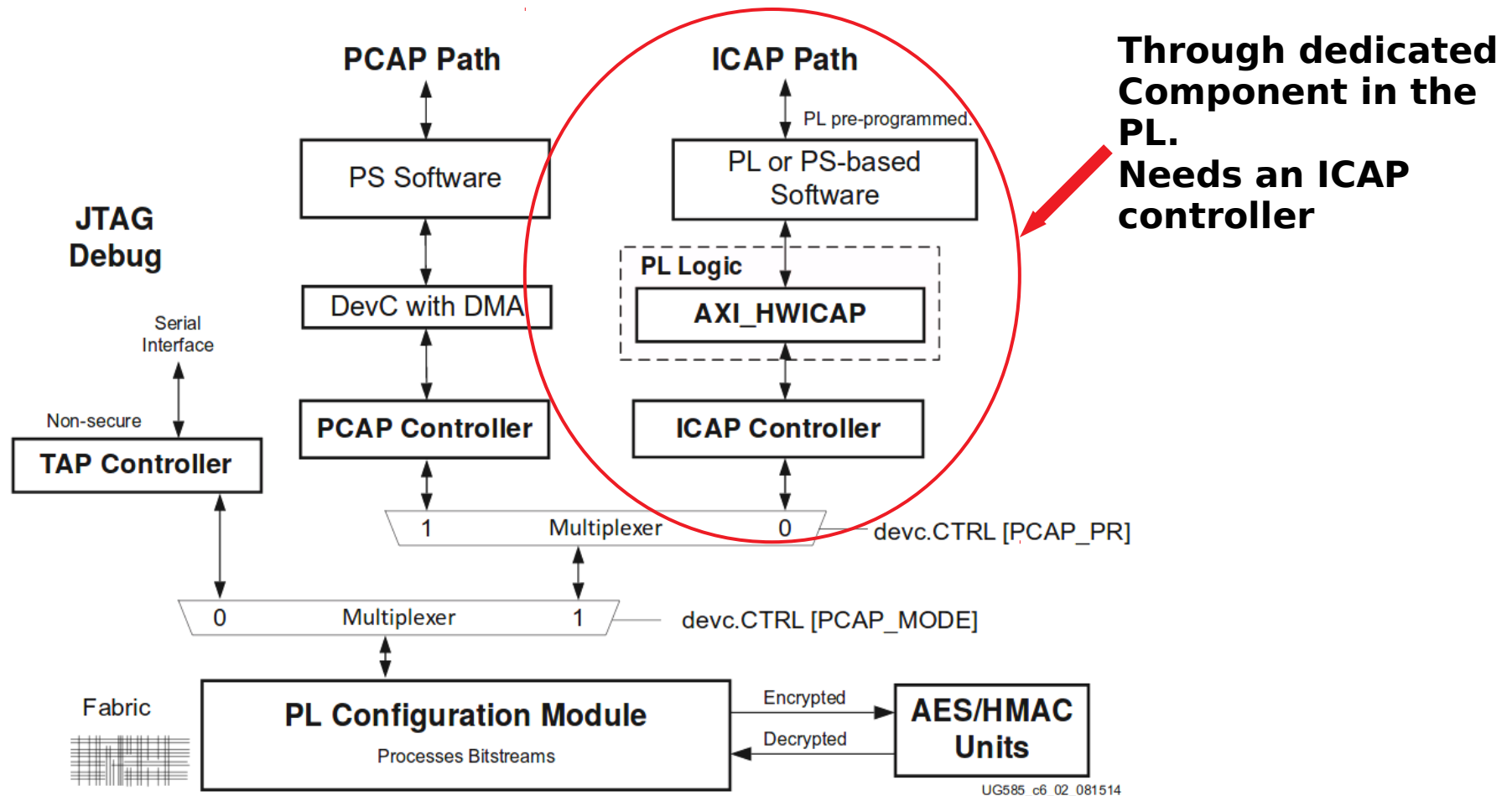


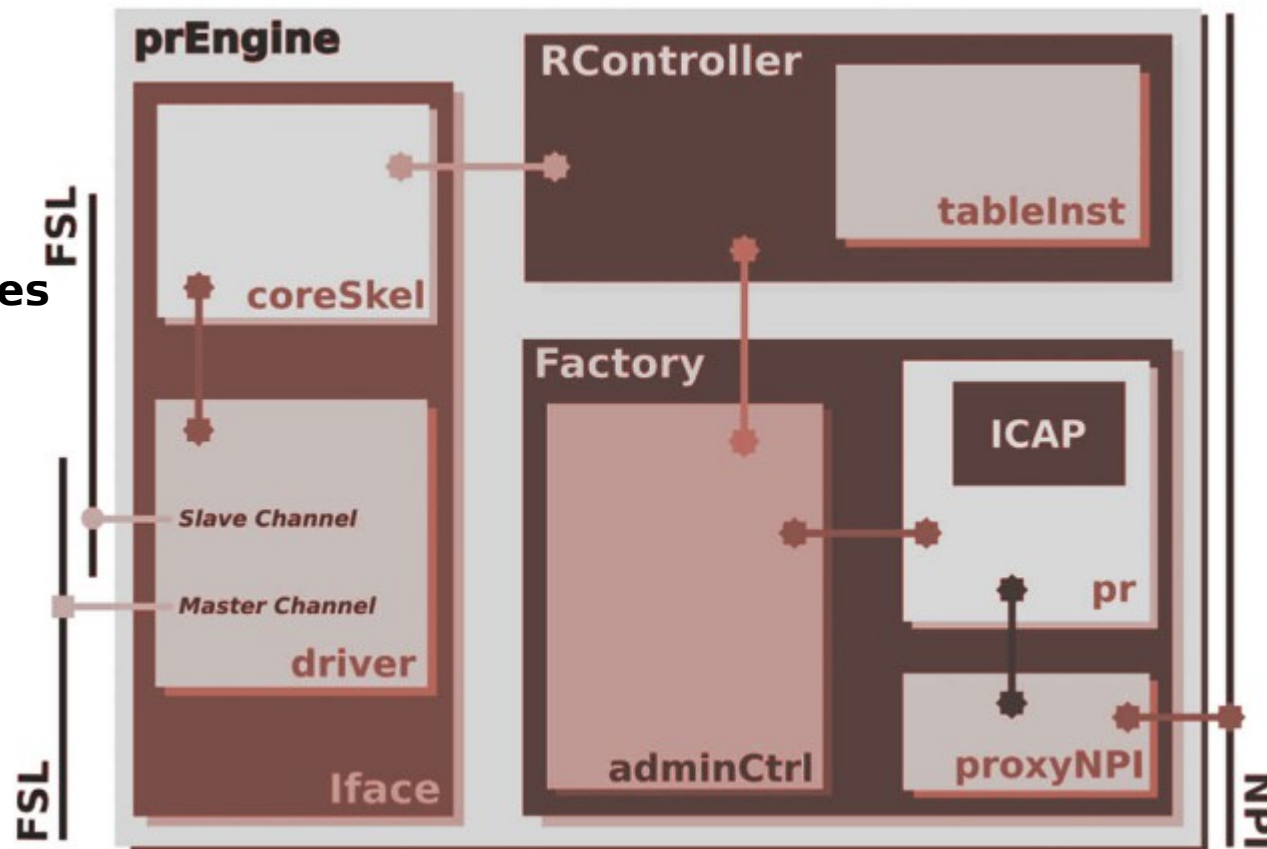
Figure 6-2: PL Configuration Paths

Source: [https://www.xilinx.com/support/documentation/user\\_guides/ug585-Zynq-7000-TRM.pdf](https://www.xilinx.com/support/documentation/user_guides/ug585-Zynq-7000-TRM.pdf)

# FPGA Partial Reconfiguration Dynamic Reconfiguration Management

## Example of Dedicated Hw Component for ICAP management

### Reconfiguration Engine



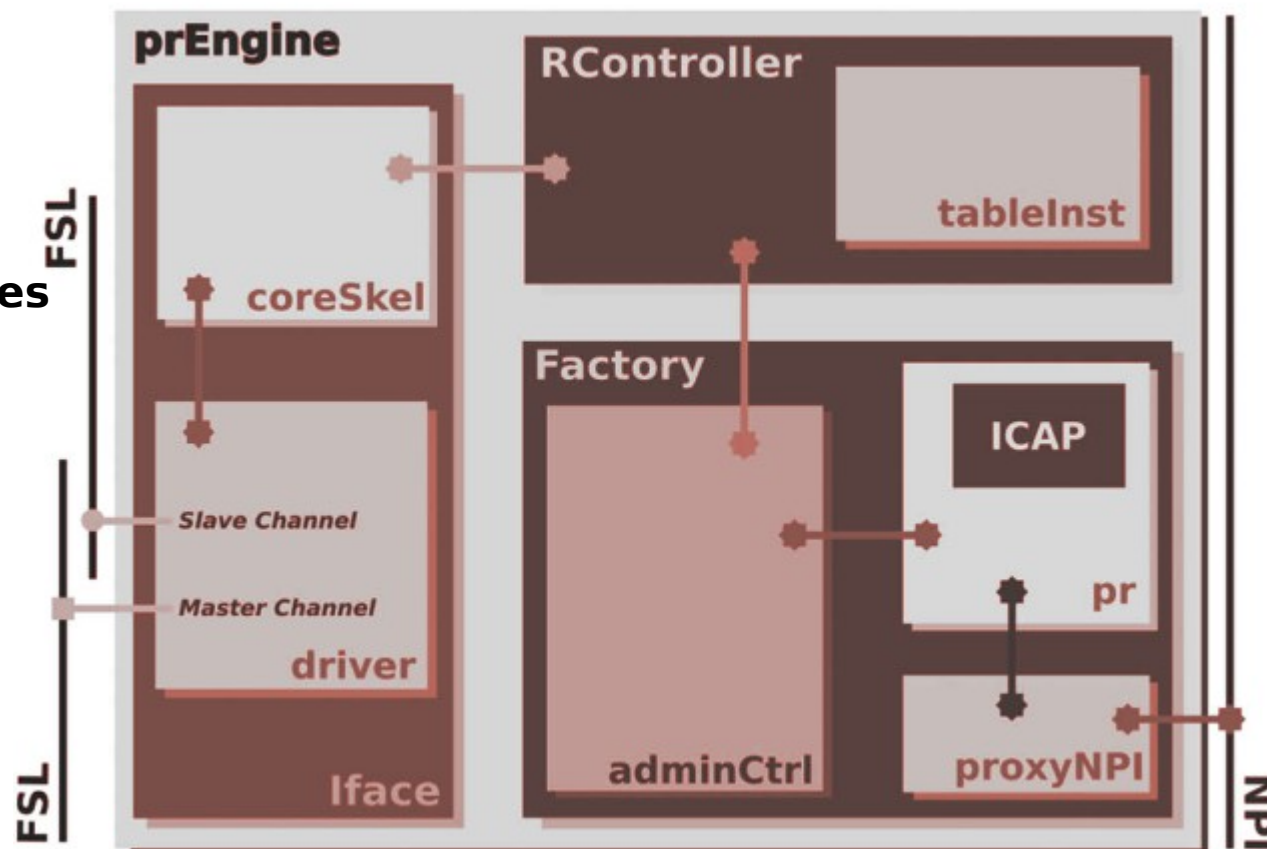
Source: T. G. Cervero et al., "A Scalable and Dynamically Reconfigurable FPGA-Based Embedded System for Real-Time Hyperspectral Unmixing," in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 8, no. 6, pp. 2894-2911, June 2015.



# FPGA Partial Reconfiguration Dynamic Reconfiguration Management

## Example of Dedicated Hw Component for ICAP management

### Reconfiguration Engine



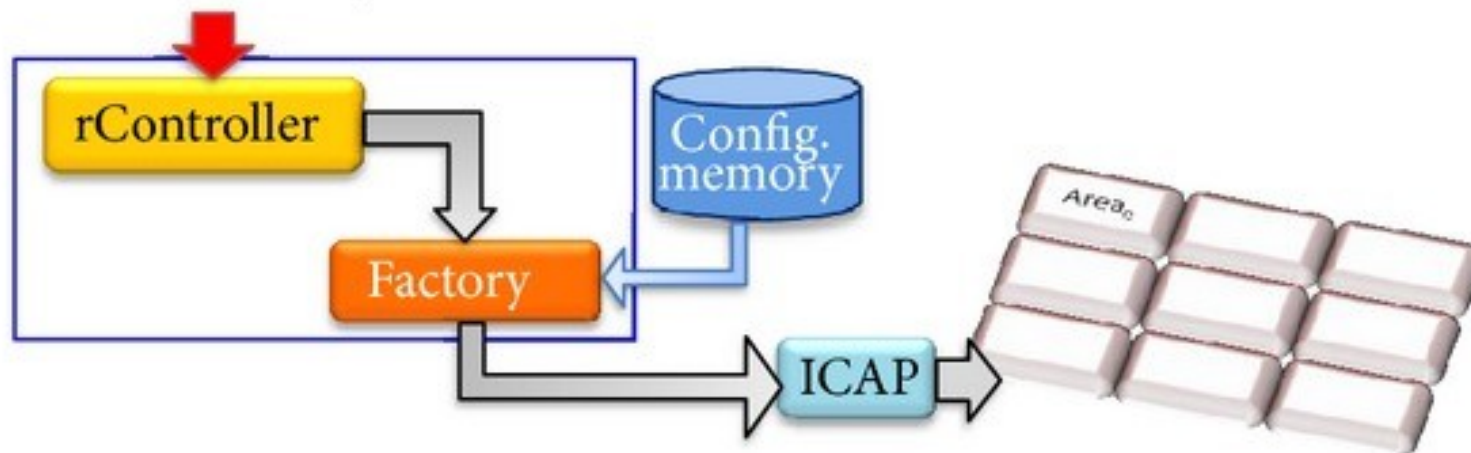
It is responsible for offering a set of reconfiguration services through a simple bus interface, which is based on a bidirectional FSL and a unidirectional Native Port Interface (NPI)

Source: T. G. Cervero et al., "A Scalable and Dynamically Reconfigurable FPGA-Based Embedded System for Real-Time Hyperspectral Unmixing," in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 8, no. 6, pp. 2894-2911, June 2015.

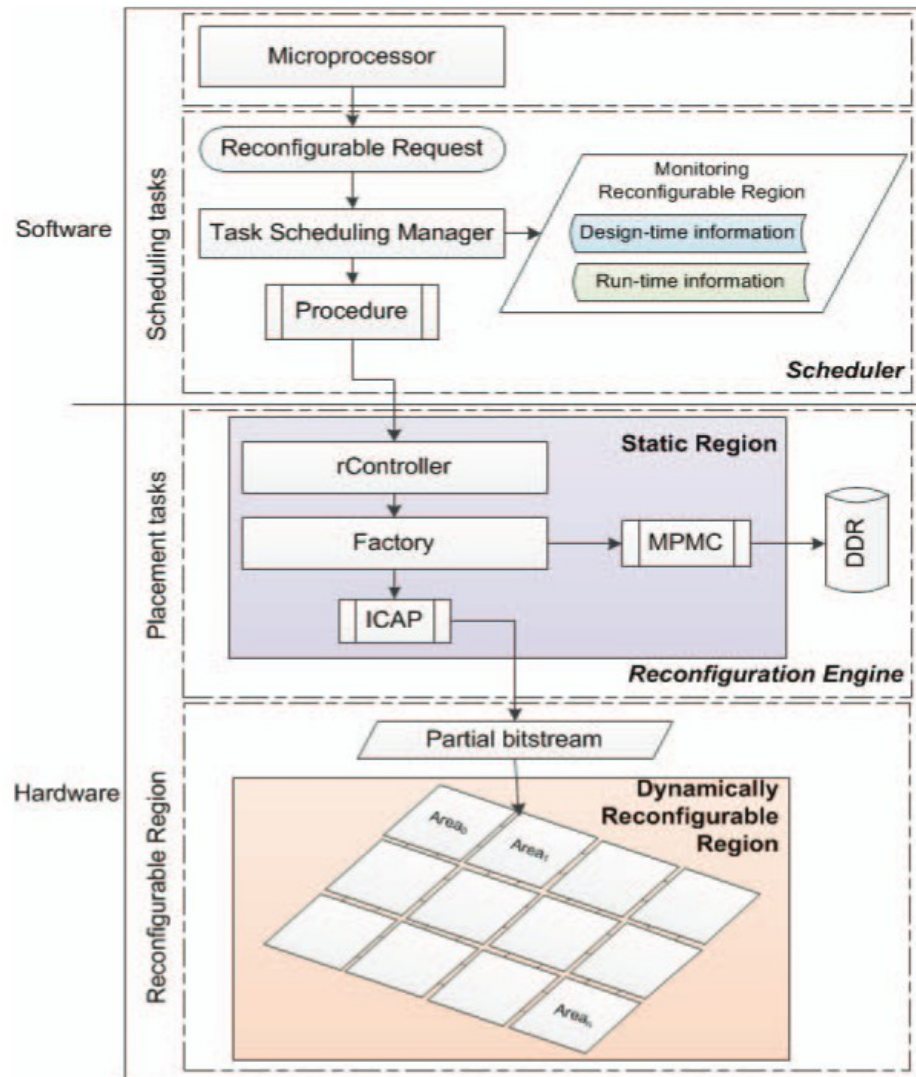
# FPGA Partial Reconfiguration Dynamic Reconfiguration Management

## Reconfiguration Engine

Scheduler task request



# FPGA Partial Reconfiguration Dynamic Reconfiguration Management



## Planification of Partial reconfiguration

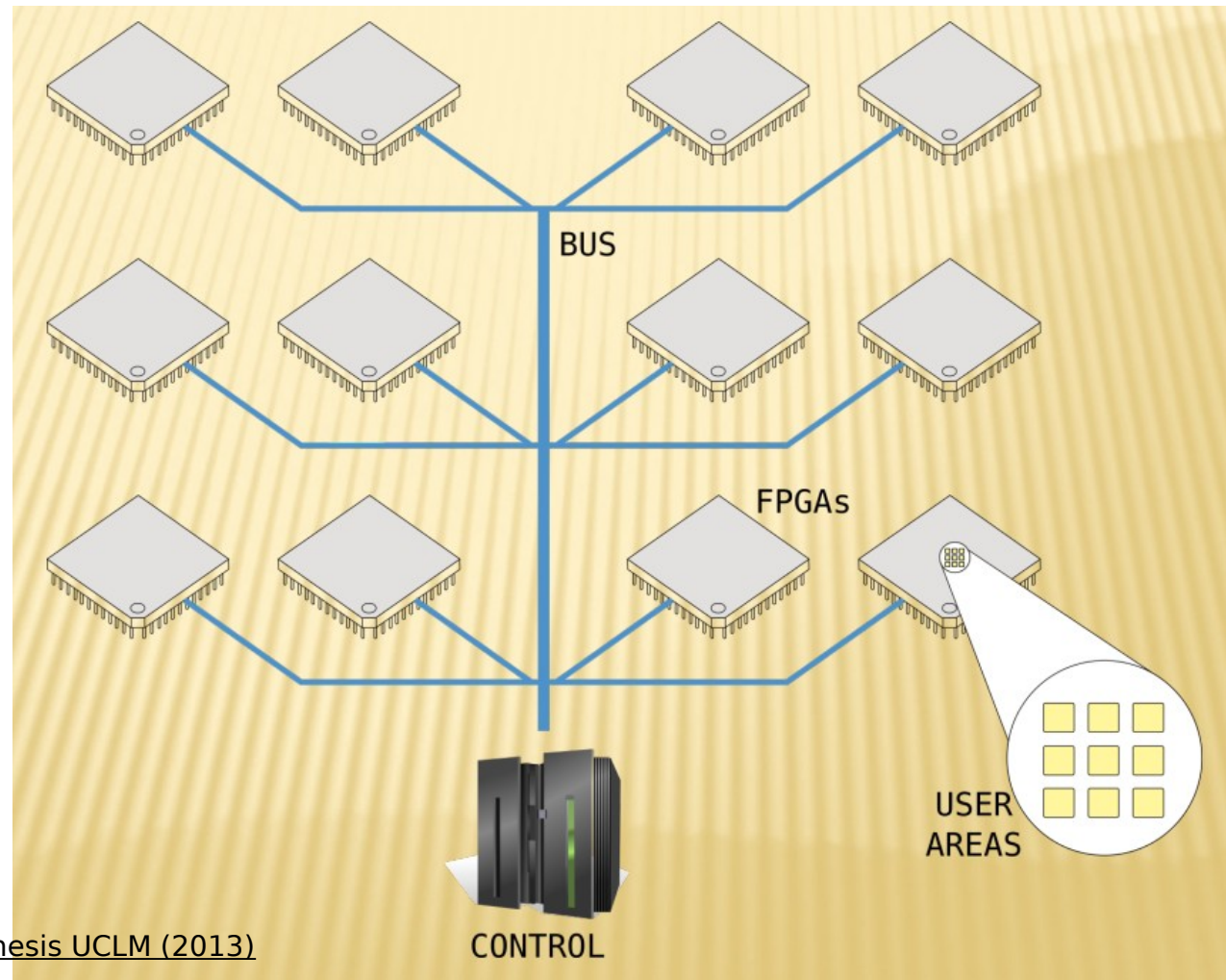
- Selecting the new reconfiguration process
- Which component will be evicted
- Planning when to stop this component
- Which area will be used ( for those components that belongs to different RP)

Source: T. Cervero et al., "A Resource Manager for Dynamically Reconfigurable FPGA-Based Embedded Systems," 2013 Euromicro Conference on Digital System Design, pp 633-640.

# FPGA Partial reconfiguration

Some examples using Partial Reconfiguration

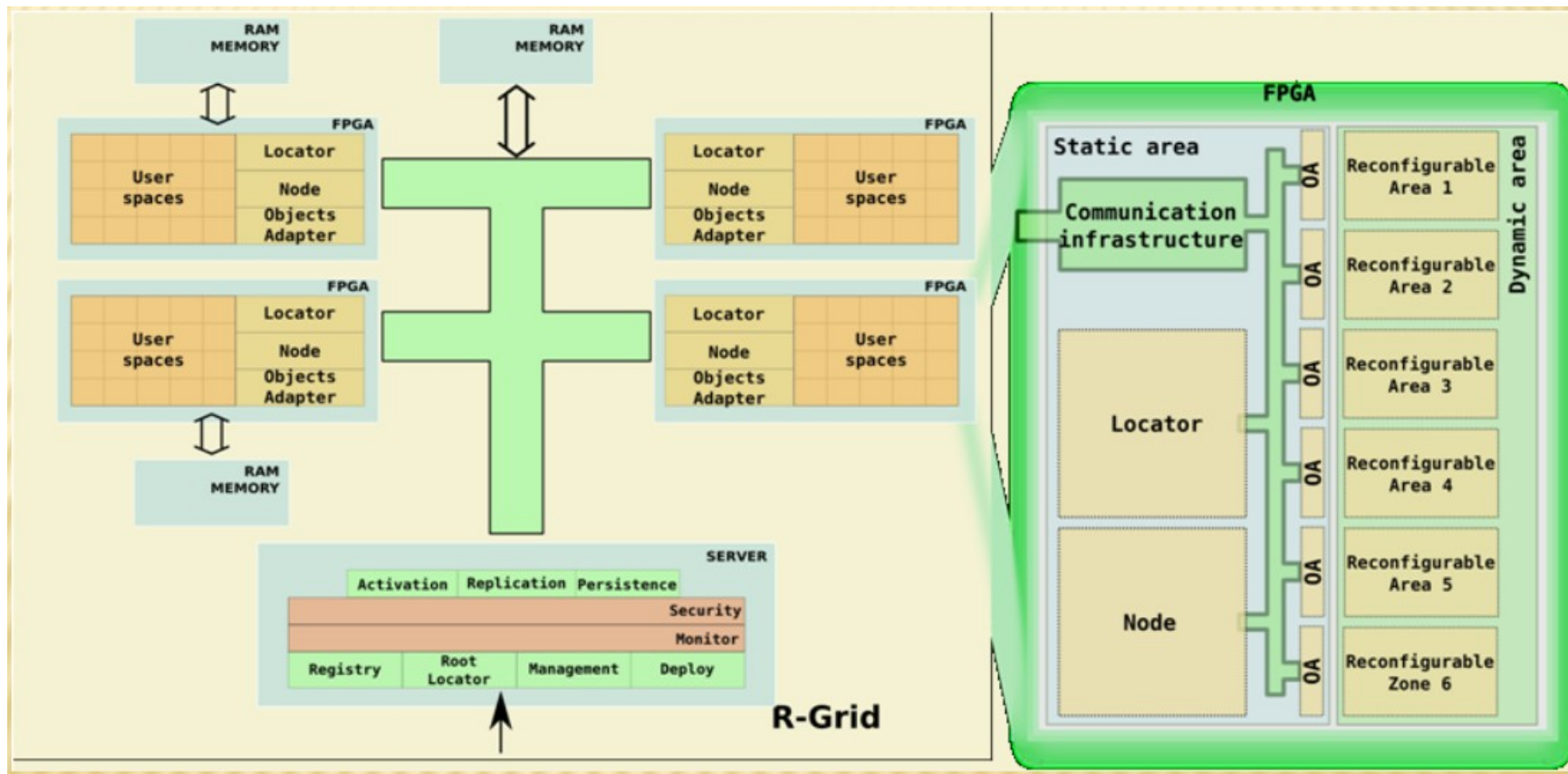
# FPGA Partial reconfiguration R-GRID



Francisco Sanchez Ph.D Thesis UCLM (2013)

# FPGA Partial reconfiguration

## R-Grid FPGA details



Francisco Sanchez Ph.D Thesis UCLM (2013)

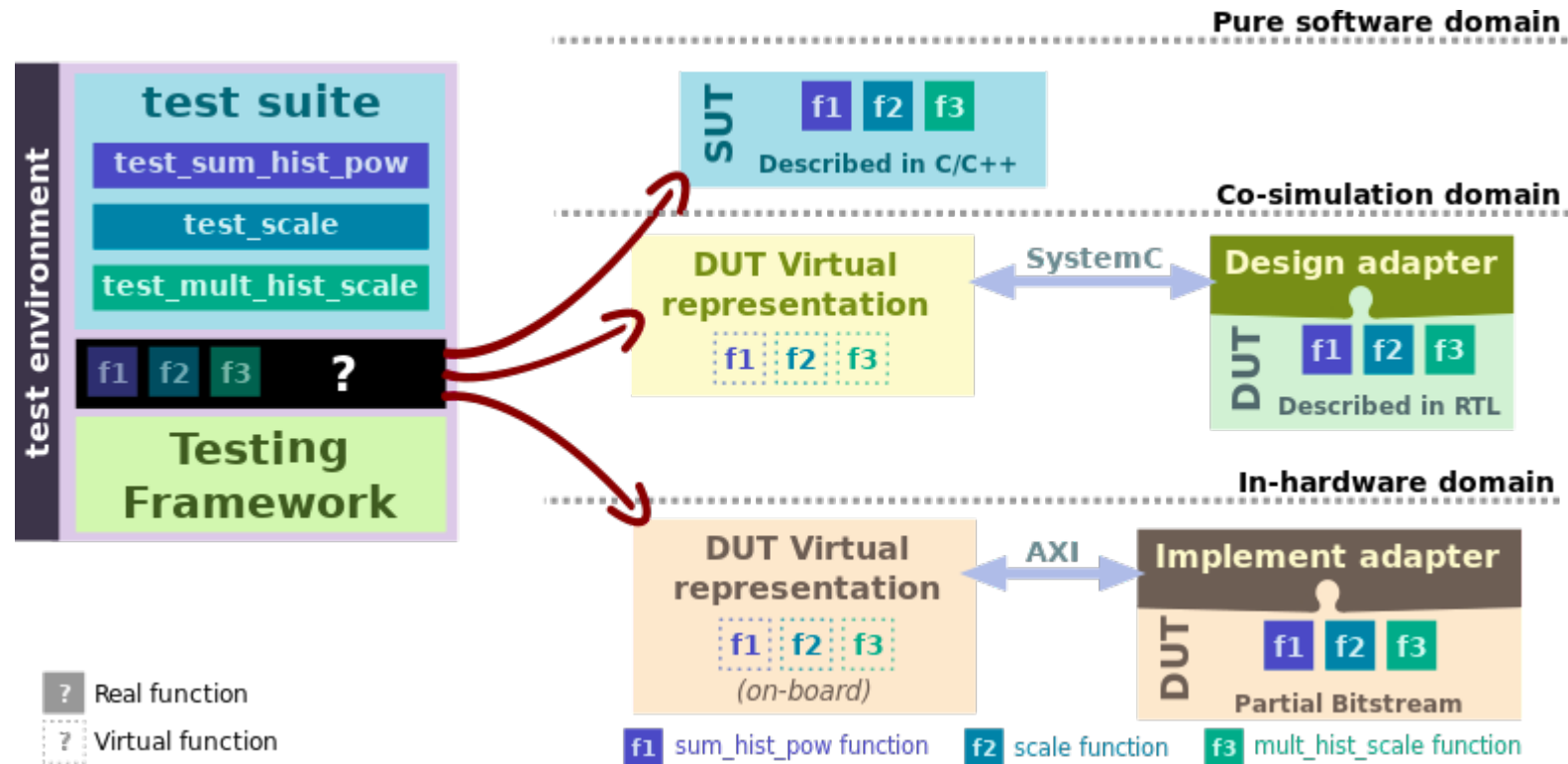
# FPGA Partial reconfiguration Testing Framework for FPGA design

Functional and timing in-hardware  
verification of FPGA-based designs using  
unit testing frameworks

First Prize: **Xilinx Open Hardware Contest 2017**

Julian Caba Ph.D Thesis UCLM (2017)

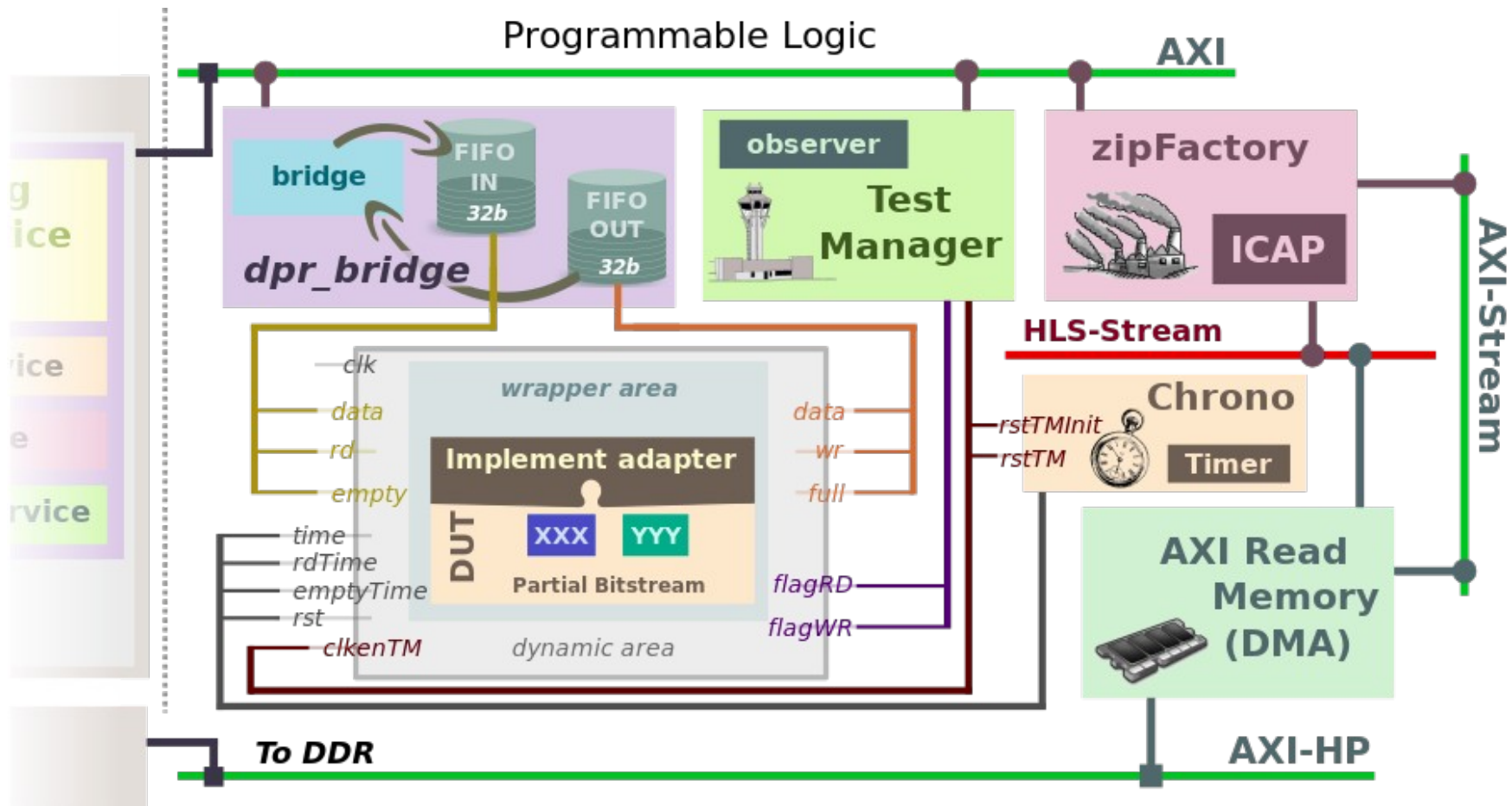
# FPGA Partial reconfiguration Testing Framework for FPGA design



Julian Caba Ph.D Thesis UCLM (2017)



# FPGA Partial reconfiguration Testing Framework for FPGA design



Julian Caba Ph.D Thesis UCLM (2017)

# FPGA Partial reconfiguration Testing Framework for FPGA design

a) **0011 2233 4455 6677 8899 AABB CCDD EEFF**

```

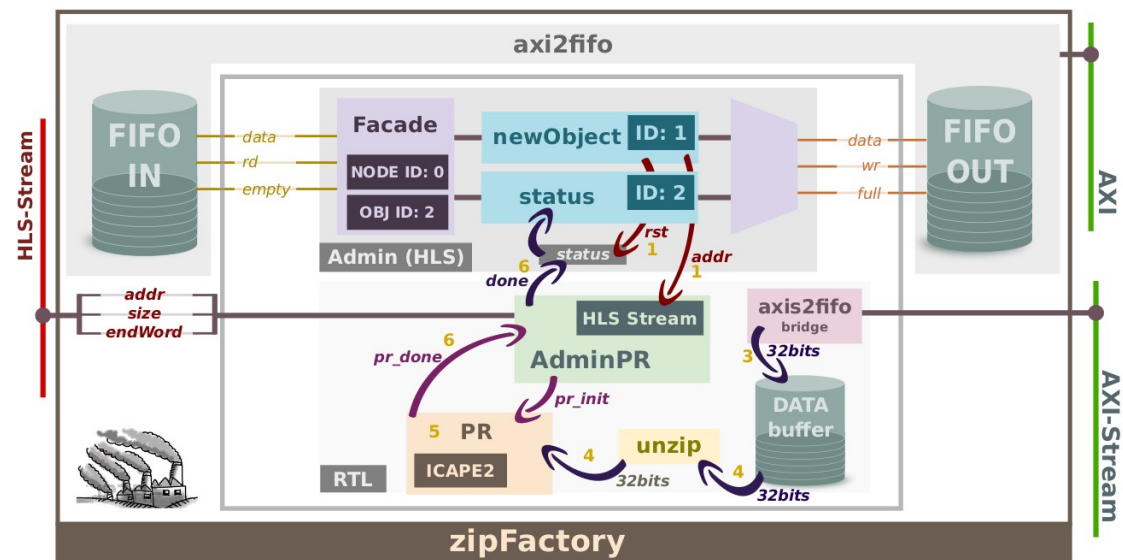
00000000: 3834 0063 000B 3230 3136 2F30 322F 3236
00000010: 0064 0009 3135 3A30 383A 3037 0065 003D
00000020: BAFC FFFF FFFF FFFF FFFF FFFF FFFF FFFF
00000030: FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF
00000040: FFFF 0000 00BB 1122 0044 FFFF FFFF FFFF
00000050: FFFF AA99 5566 2000 0000 3002 2001 0000
00000060: 0000 3002 0001 .....
.....
000007E0: ..... FFFF FFFF FFFF FFFF FFFF FFFF
000007F0: 0301 0000 0000 0000 0000 0000 0000 0000
00000800: 0000 0000 0000 FFFF FFFF FFFF 3000 8001
00000810: 0000 0005 0000 0000 FFFF .....
.....
003F9EE0: ..... 0000 E3C3 ECF5 2000 0000 3000
003F9EF0: 8001 0000 000D 2000 0000 2000 0000 2000
003F9F00: 0000 2000 0000 .....
    
```

b) **0011 2233 4455 6677 8899 AABB CCDD EEFF**

```

00000000: FFFF FFFF FFFF FFFF 0000 00BB 1122 0044
00000010: FFFF FFFF FFFF FFFF AA99 5566 2000 0000
00000020: 3002 2001 0000 0000 3002 0001 .....
.....
00000540: ..... FFFF FFFF FFFF FFFF FFFF FFFF
00000550: 3003 6001 0000 0005 FFFF FFFF FFFF FFFF
00000560: 3000 8001 0000 0005 0000 0000 .....
.....
00001CA0: ..... E313 81E5 2000 0000 3000 8001
00001CB0: 0000 000D
    
```

Julian Caba Ph.D Thesis UCLM (2017)



# FPGA Partial reconfiguration Documentation

[https://www.xilinx.com/support/documentation/  
sw\\_manuals/xilinx2018\\_1/ug947-vivado-partial-reconfiguration-tutorial.pdf](https://www.xilinx.com/support/documentation/sw_manuals/xilinx2018_1/ug947-vivado-partial-reconfiguration-tutorial.pdf)

[https://www.xilinx.com/support/documentation/  
sw\\_manuals/xilinx2018\\_3/ug909-vivado-partial-reconfiguration.pdf](https://www.xilinx.com/support/documentation/sw_manuals/xilinx2018_3/ug909-vivado-partial-reconfiguration.pdf)

[https://www.xilinx.com/support/documentation-navigation/  
design-hubs/dh0017-vivado-partial-reconfiguration-hub.html](https://www.xilinx.com/support/documentation-navigation/design-hubs/dh0017-vivado-partial-reconfiguration-hub.html)

# FPGA Partial reconfiguration

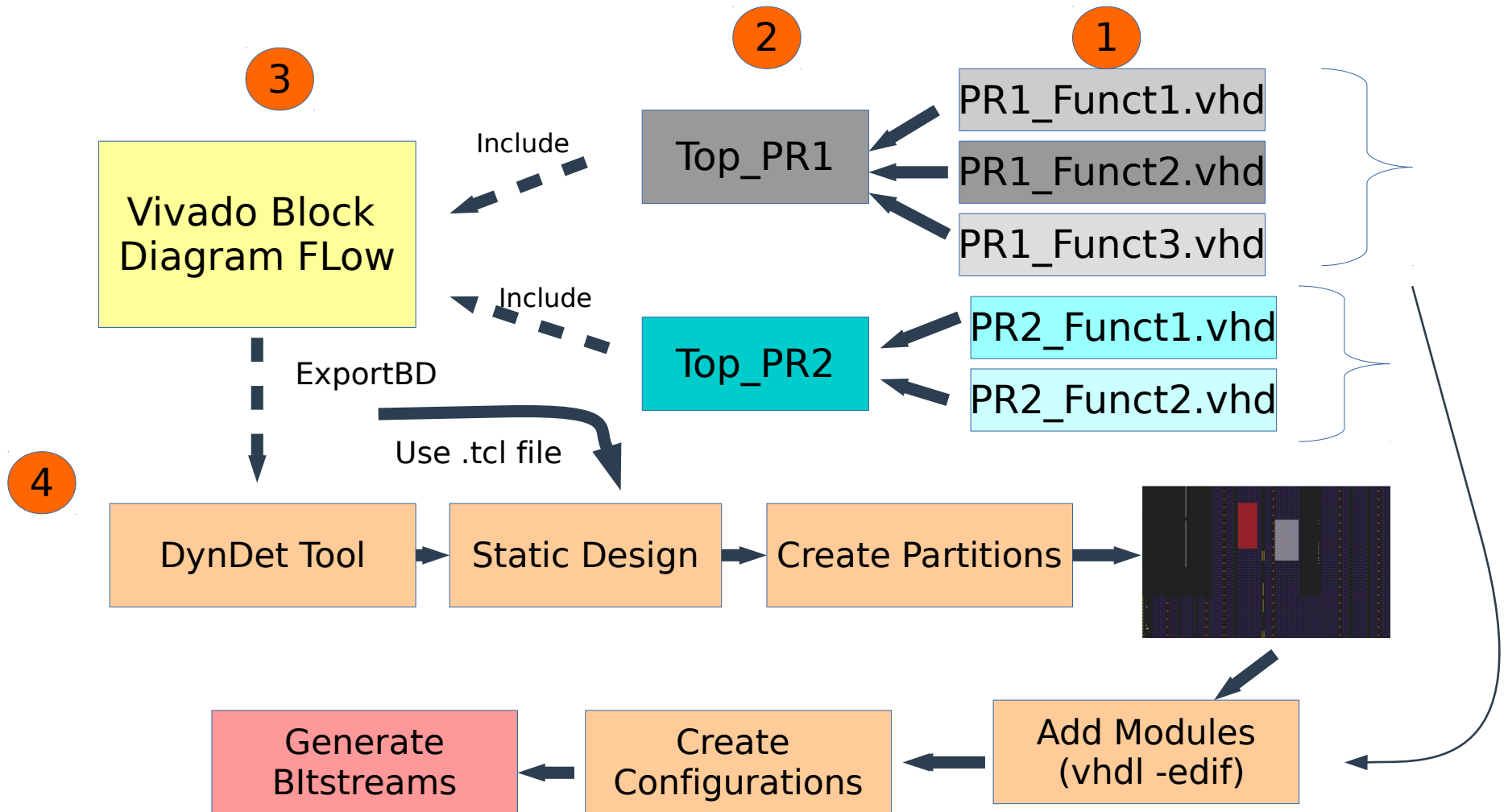
## **DynDeT: Dynamic Partial Reconfiguration Design Tool**

# *DynDet: **D**ynamic **P**artial **R**econfiguration **D**esign **T**ool*

**DynDet** is a QT-based graphical tool created by ARCO Research group of University of Castilla-La Mancha, to manage the design process of dynamically reconfigurable systems

Built over Vivado tools, dynDet tool automatizes the process design of partial reconfigurable system.

# Partial Reconfiguration Flow using dynDet tool




# Partial Reconfiguration using dynDet tool

dynDeT: dynamic partial reconfiguration DDesign Tool

Tools Help

**dynDeT: dynamic partial reconfiguration DDesign Tool**

Output directory:  

**Design flow**

- General information**
- Static part
- Reconfigurable partitions
- Reconfigurable modules
- Configurations
- Bitstreams


FPGA data


Part:

Board:

Vivado project data

Block design name:

Block design file:  

User IPs path (optional):  

# Partial Reconfiguration using dynDet tool

Tools Help

**dynDeT: dynamic partial reconfiguration DDesign Tool**

Output directory:

**Design flow**

- General information
- Static part
- Reconfigurable partio...**
- Reconfigurable modules
- Configurations
- Bitstreams

**Add new partition**

Reconfigurable module name:

Instance name:

Instance path (Vivado Tree):

Hardware location:

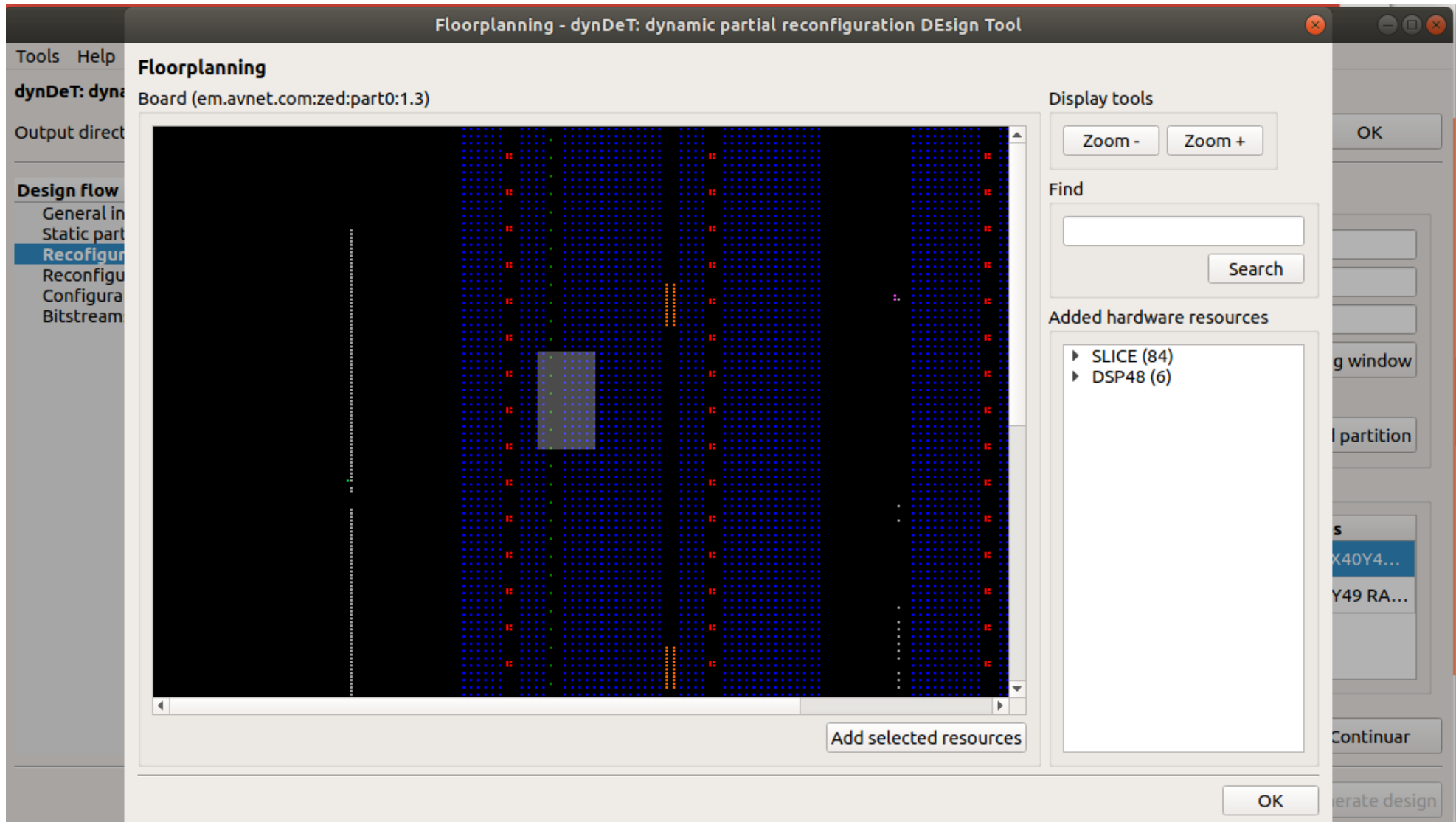
Properties:  RESET\_AFTER\_RECONFIG  SNAPPING\_MODE

**Available partitions**

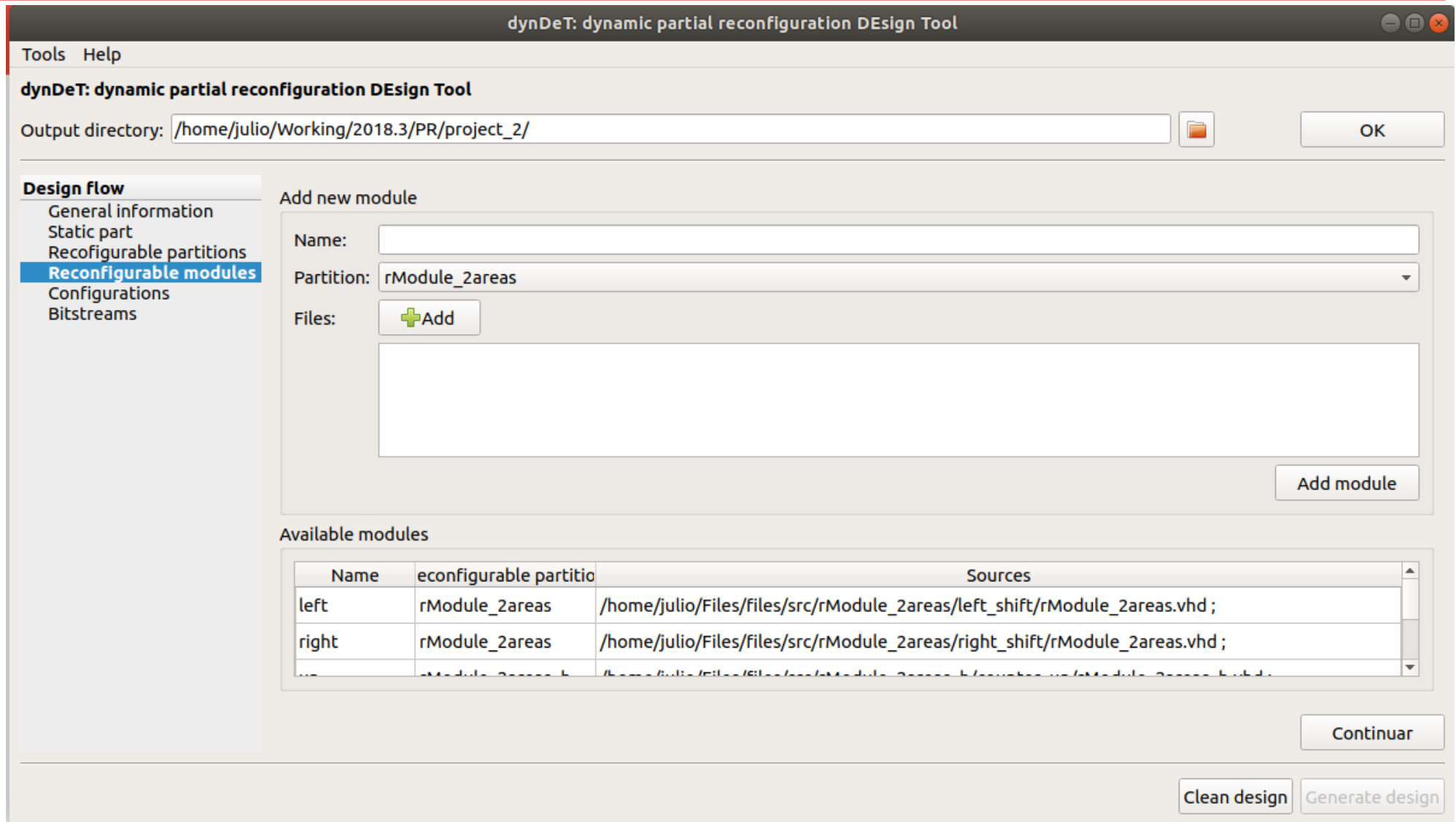
| Name             | Instance    | Source                      | ET_AFTER_RECONF | SNAPPING_MODE | HW resources                 |
|------------------|-------------|-----------------------------|-----------------|---------------|------------------------------|
| rModule_2areas   | rm_2areas   | design_1_i/top_shift_0/U0   | true            | true          | SLICE_X20Y19:SLICE_X40Y4...  |
| rModule_2areas_b | rm_2areas_b | design_1_i/top_counter_0... | true            | true          | SLICE_X5Y0:SLICE_X6Y49 RA... |



# Partial Reconfiguration using dynDet tool



# Partial Reconfiguration using dynDet tool




# Partial Reconfiguration using dynDet tool

dynDeT: dynamic partial reconfiguration DDesign Tool

Tools Help

**dynDeT: dynamic partial reconfiguration DDesign Tool**

Output directory:  

**Design flow**

- General information
- Static part
- Reconfigurable partitions
- Reconfigurable modules
- Configurations**
- Bitstreams

**Add new configuration**

Name:

Modules to implement:

| Partition        | Module              |
|------------------|---------------------|
| rModule_2areas   | right (reference) ▼ |
| rModule_2areas_b | up (reference) ▼    |

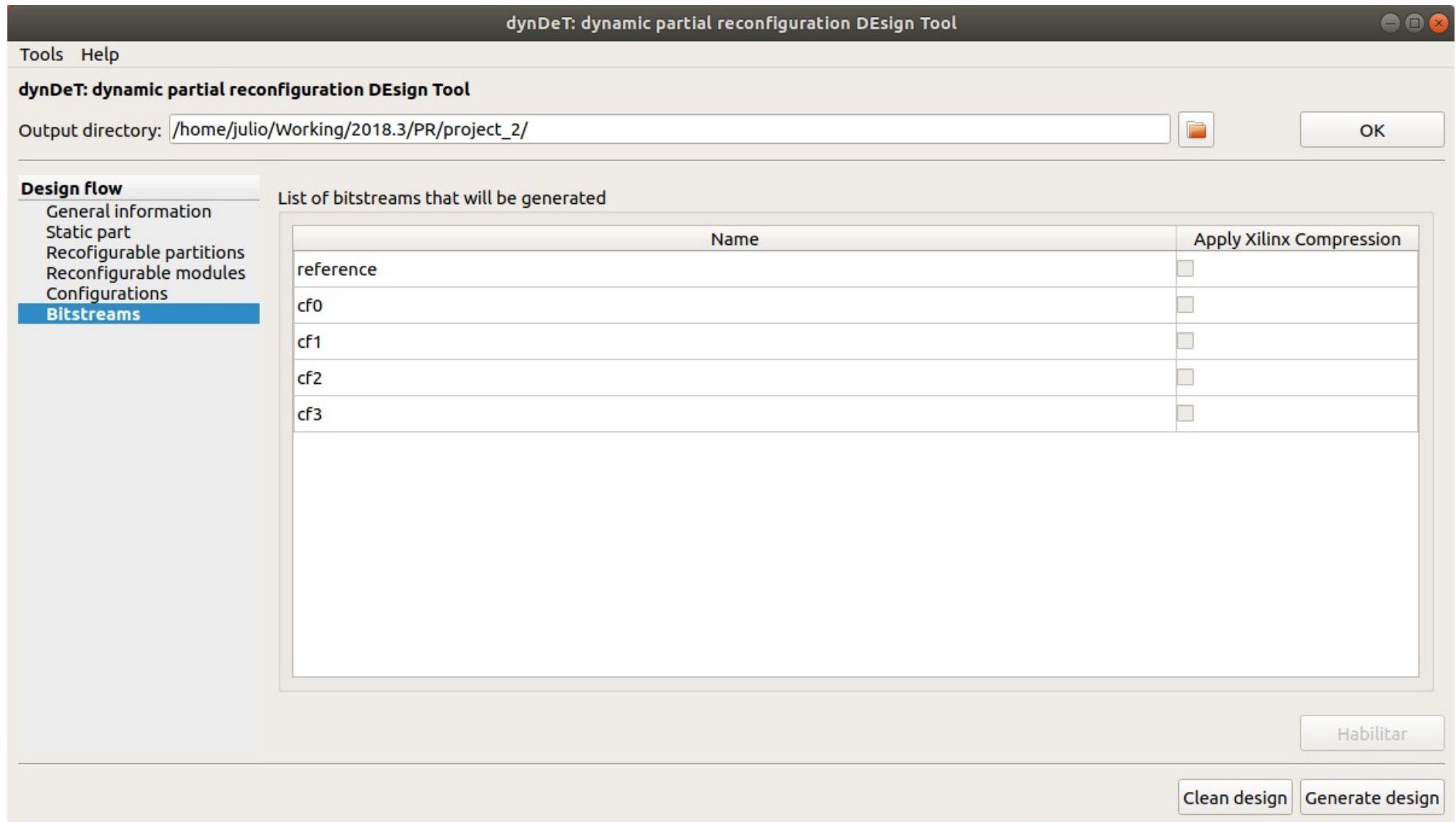
**Configurations**

| Name | Modules to implement                             |
|------|--|
| cf1  | right (rModule_2areas); up (rModule_2areas_b);   |
| cf2  | right (rModule_2areas); down (rModule_2areas_b); |

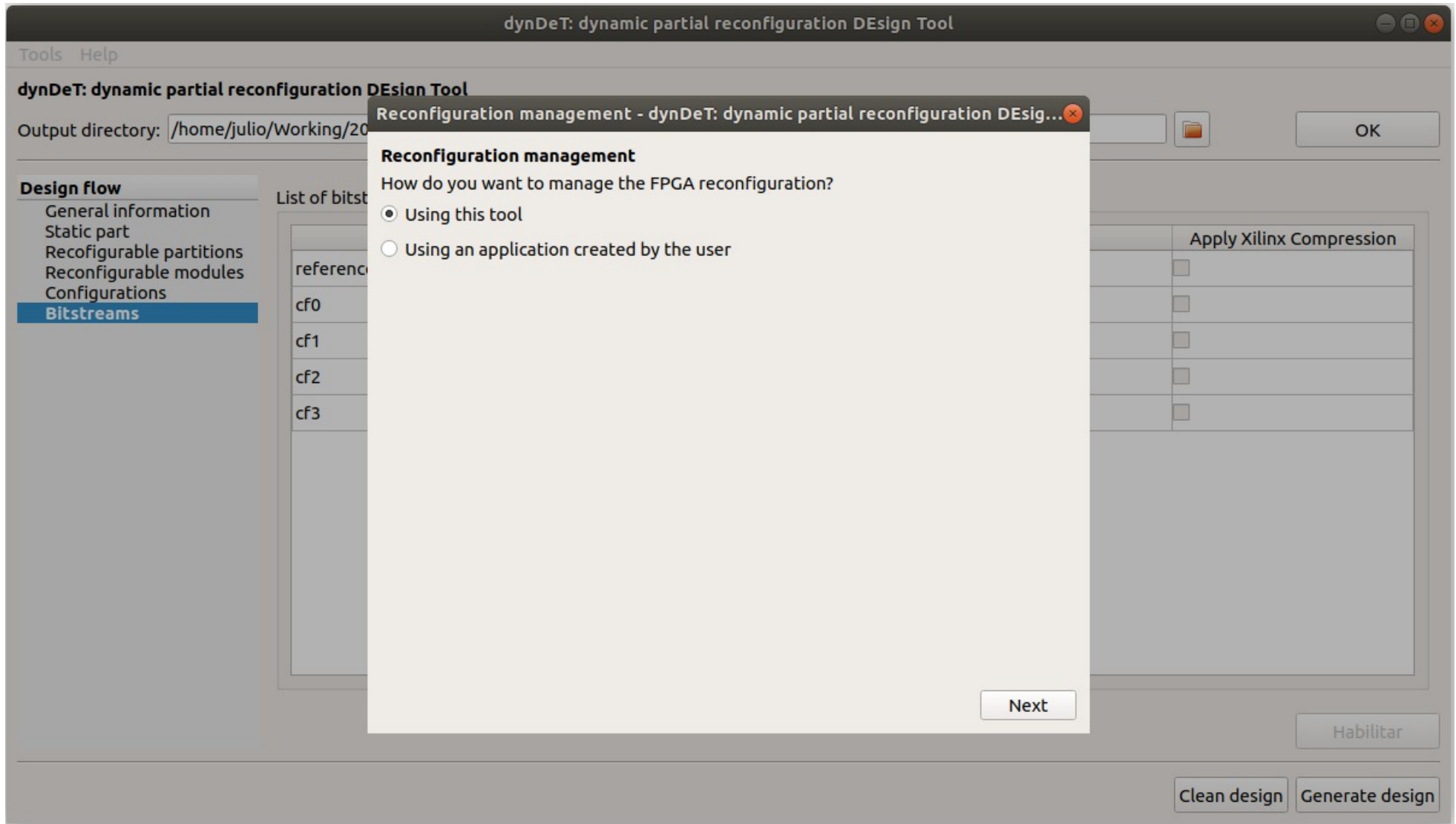
**Other options**

Generate blanking configuration

# Partial Reconfiguration using dynDet tool



# Partial Reconfiguration using dynDet tool



# Partial Reconfiguration using dynDeT tool

**Use dynDeT,**

**<https://github.com/juliancaba/dynDeT>**

# FPGA Partial reconfiguration

**Thanks for your attention!**

**Questions?**