Architecture of reconfigurable systems : Partial reconfiguration mechanisms

J. Lorandel

jordane.lorandel@u-cergy.fr

Slides available there: https://perso-etis.ensea.fr//lorandel/M2_ESI_ASR.php

UNIVERSITÉ
de Cergy-Pontoise

Organization

Outline

1. Resources
2. Partial Reconfiguration
   - Reconfiguration modes in FPGA
   - Why use (dynamic) partial reconfiguration?
3. Methodology
   - Different ways to perform PR
   - Module-based partial reconfiguration
   - Difference-based partial reconfiguration
4. Conclusion

UNIVERSITÉ de Cergy-Pontoise

Resources

- This course uses a Xilinx FPGA board. There are plenty of resources available out there (mostly online)

- From Xilinx Inc.
  - Xilinx Zynq-7000
  - http://www.xilinx.com/products/zynq-7000/third-party-documentation.htm
    - See : http://www.xilinx.com/support/documentation/data_sheets/ds180_7series_0verview.pdf
    - Reference Manual : Zynq-7000 All Programmable Technical Reference Manual **UG585**

Resources

Partial Reconfiguration – Xilinx

- Overall Xilinx documentation page for partial reconfiguration:

https://www.xilinx.com/products/design-tools/vivado/implementation/dynamic-function-exchange.html#documentation

- See also :
  - User Guide: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2017_1/ug909-vivado-partial-reconfiguration.pdf
  - Tutorial: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2017_4/ug947-vivado-partial-reconfiguration-tutorial.pdf

Resources

Partial Reconfiguration – Altera

- Overall page:

https://www.intel.com/content/www/us/en/programmable/products/design-software/fpga-design/quartus-prime/features/partial-reconfiguration.html

- See also:
  - User Guide: https://www.intel.com/content/www/us/en/programmable/products/design-software/fpga-design/quartus-prime/user-guides.html
  - Tutorial: https://github.com/intel/fpga-partial-reconfig/tree/master/tutorials
  - Intel PR page: https://01.org/fpga-partial-reconfiguration

UNIVERSITÉ
de Cergy-Pontoise

❑ Partial Reconfiguration

UNIVERSITÉ
de Cergy-Pontoise

WHY ??

- Lack of hardware resources
  - Not enough room to include the logic of all possible hardware functions in the target
  - Need to reduce the size of the resulting device

- Flexibility
  - Implementation of various features with different algorithms
  - Implementation of various protocols to achieve the same end-goal

UNIVERSITÉ
de Cergy-Pontoise

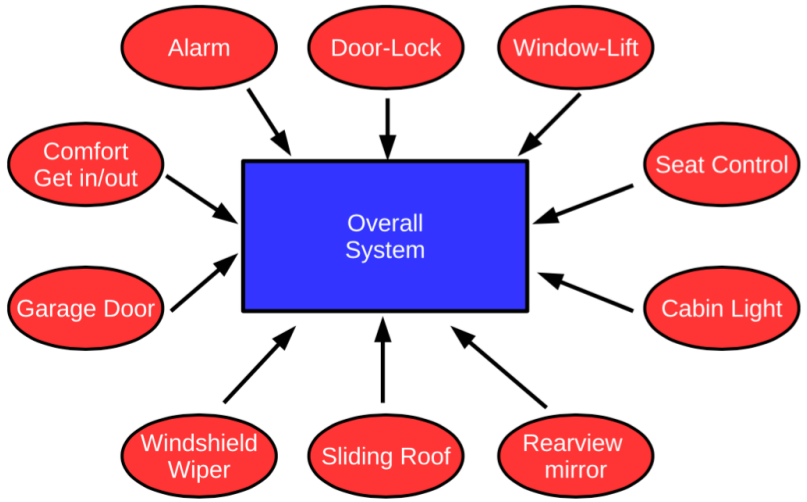
WHY ??

- Reliability and security
  - Ensure the overall system can withstand faults due to external factors
    - Bit flips due to solar flares, radiation, etc.
  - Guarantee some circuits functions are only available to specific users/devices
  - Adapt cryptographic protocols to context
- Increase reconfiguration speed when needed
  - i.e., instead of stopping the whole circuit before reconfiguring it

UNIVERSITÉ de Cergy-Pontoise

WHY ??

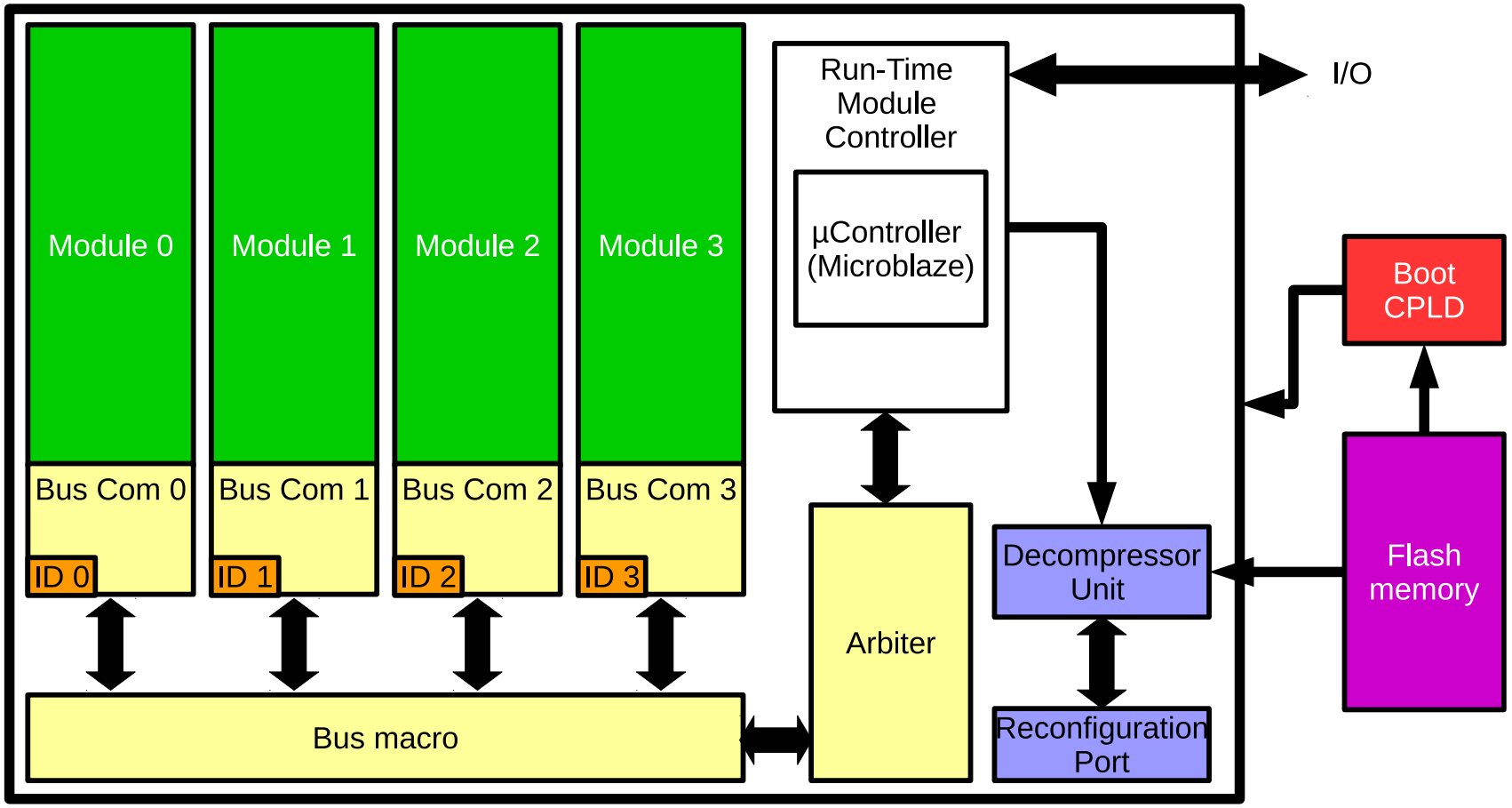
- Some examples
  - Cognitive radio
    - Adapt the modulation and coding scheme and type depending on the environment; [Rihani16]
      - ➢ Modulation, FEC (Reed-Solomon, turbo-code,…), data rate
    - Adapt the wireless communication standard and resource allocation ?
  - Automotive industry [BeckerEtAl07]
    - Reduce the amount of logic embed in a vehicle by reconfiguring HW task (depending on criticality)



Some of the cabin functions of a modern vehicle

UNIVERSITÉ de Cergy-Pontoise

WHY ??



Set of tasks to perform:

- **Communication management**
  - Delivering messages to/from the functions
  - Storing messages
- **Reconfig. Management – context switching**
  - Finding available slot
  - Saving current state
  - Sending instruction block
  - Restoring state
- **Resource Management**
  - Bookkeeping of busy/idle modules
  - Management of message buffers for each function
  - Storing state variables

UNIVERSITÉ
de Cergy-Pontoise

WHY ??

Let's play with your imagination

-> Autonomous vehicle, personalization?

-> IoT object with support of multiple wireless communications standards?

-> Energy-efficient reconfigurable devices?

…

UNIVERSITÉ de Cergy-Pontoise

Configuration types

❑Static Reconfiguration

- Load the whole bitstream on the FPGA
- Stop the execution to do the loading

❑Static Partial Reconfiguration

- Only a portion of the bitstream is load
- Stop the execution to load the partial bistream
  - ➢ But the wait time is much shorter

❑Dynamic Partial Reconfiguration (DPR)

- Part of the FPGA is being reconfigured while the rest is still running

UNIVERSITÉ de Cergy-Pontoise

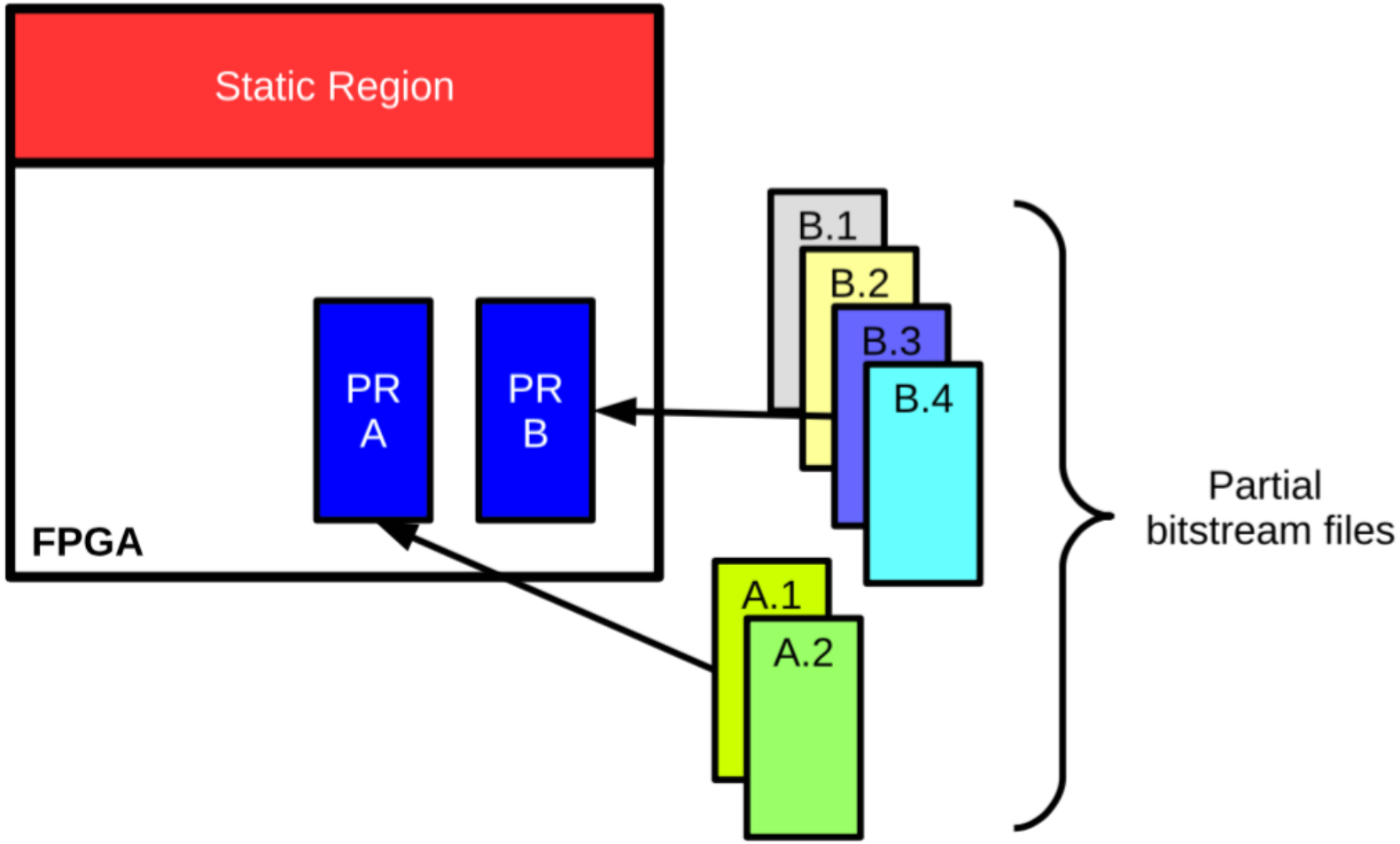
Configuration types

This course tries to provide general concept but some mechanisms are vendor-specific

- We'll mostly discuss Xilinx FPGAs for PR
  - Xilinx is one of the first vendors to propose mechanisms for (D)PR
- Altera FPGAs follow a similar path, but their toolchain and capabilities are much more recent than Xilinx'
  - PR features are mostly available for high-end FPGAs
  - There is a clear will of Intel/ltera to make it more accessible/available to everyone
  - They are not there yet

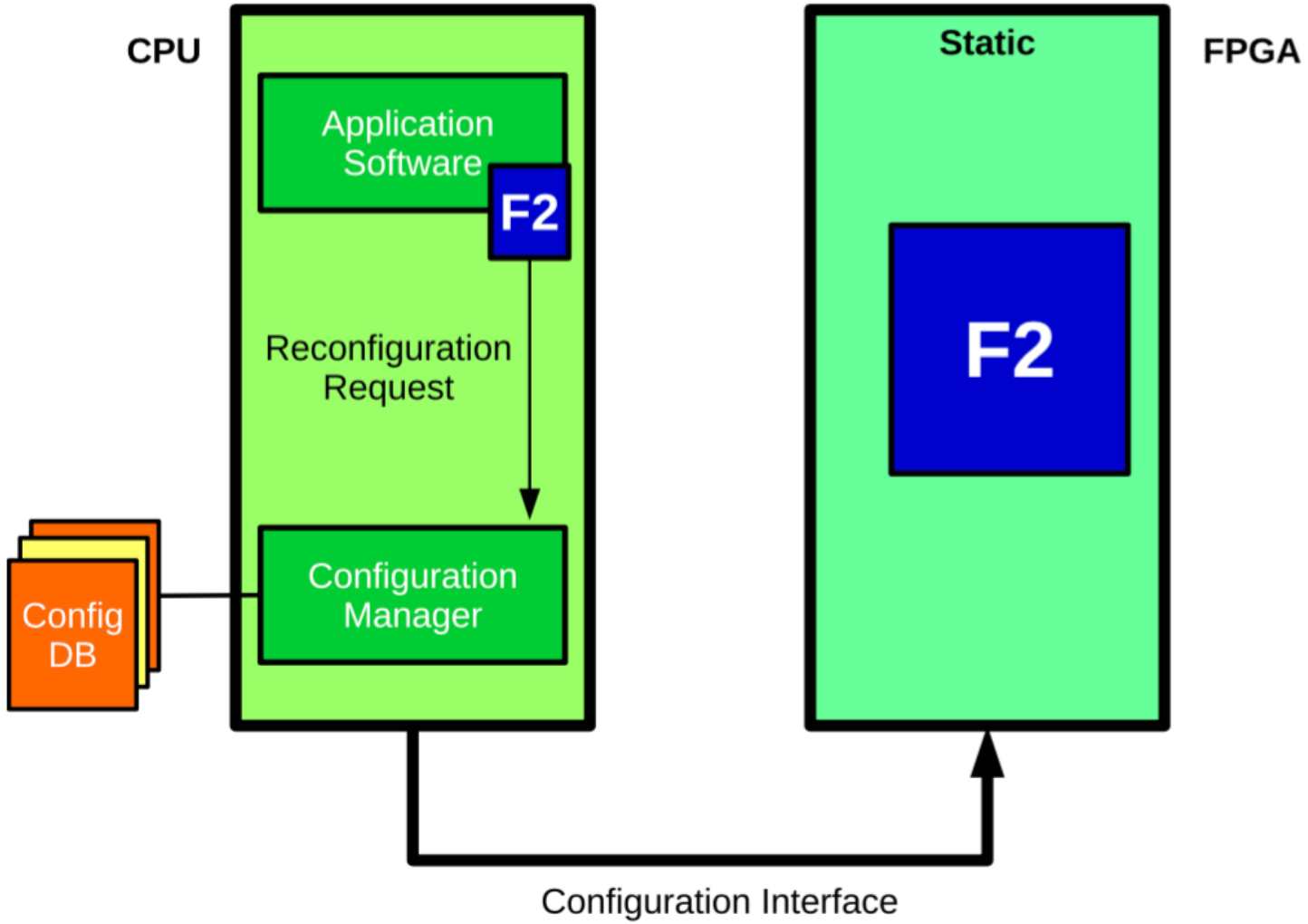
UNIVERSITÉ
de Cergy-Pontoise

Partial Reconfiguration I



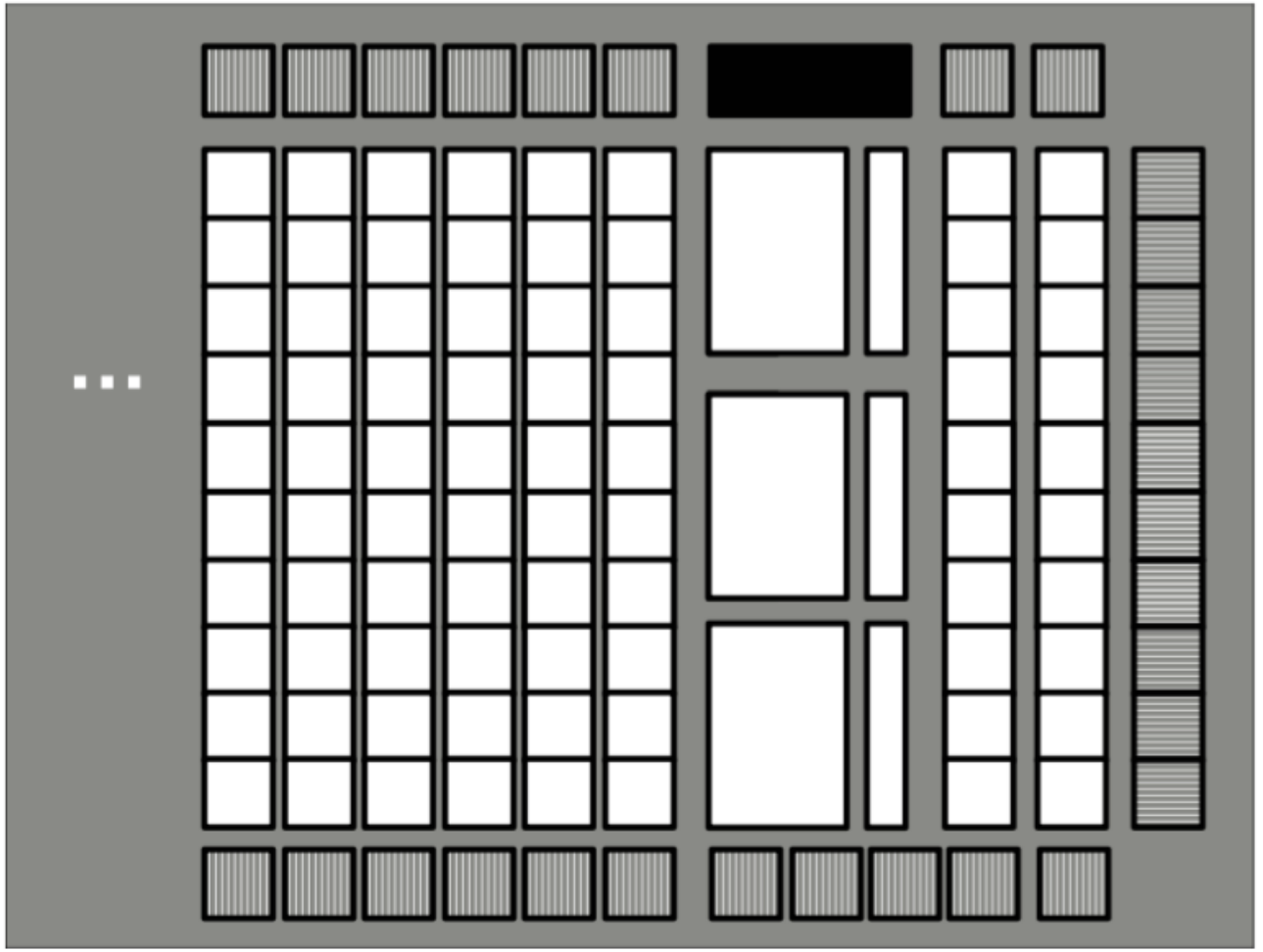
UNIVERSITÉ
de Cergy-Pontoise

Partial Reconfiguration II



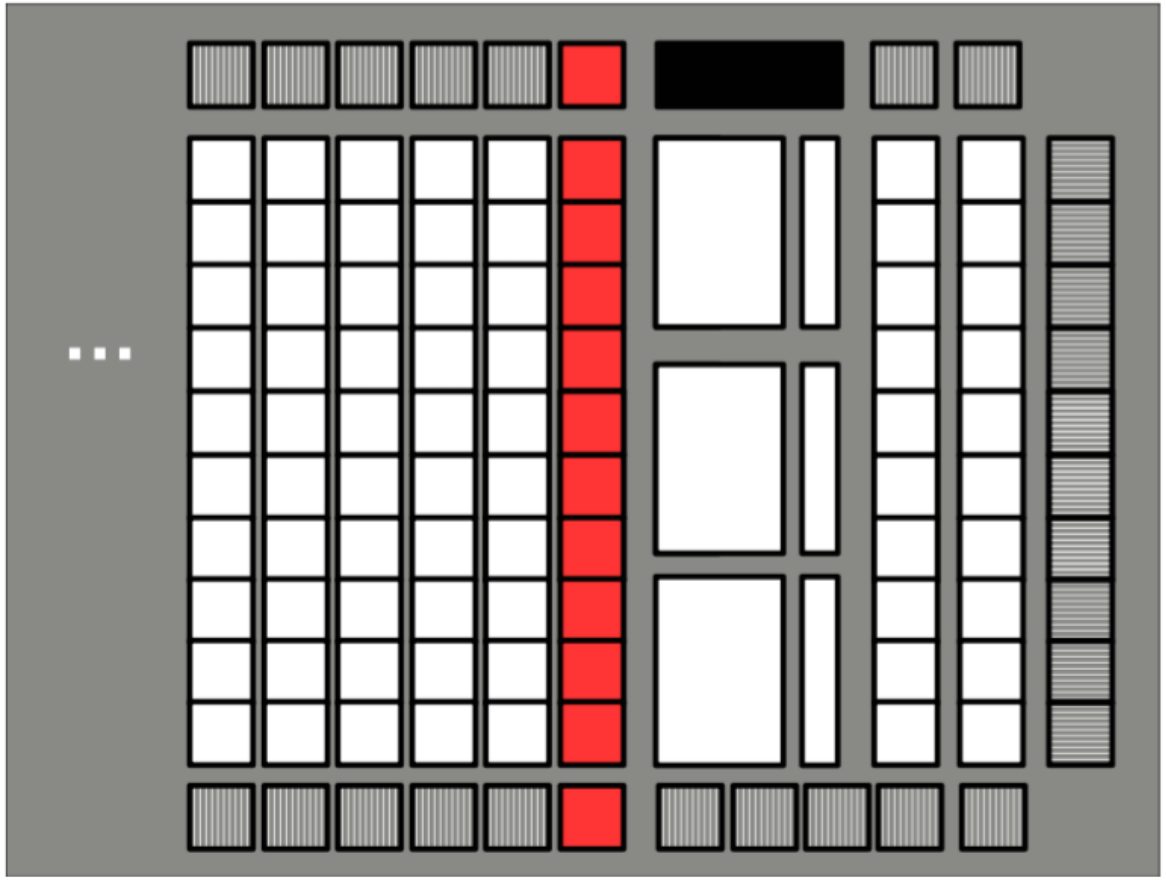
UNIVERSITÉ
de Cergy-Pontoise

Partial Reconfiguration III



UNIVERSITÉ
de Cergy-Pontoise

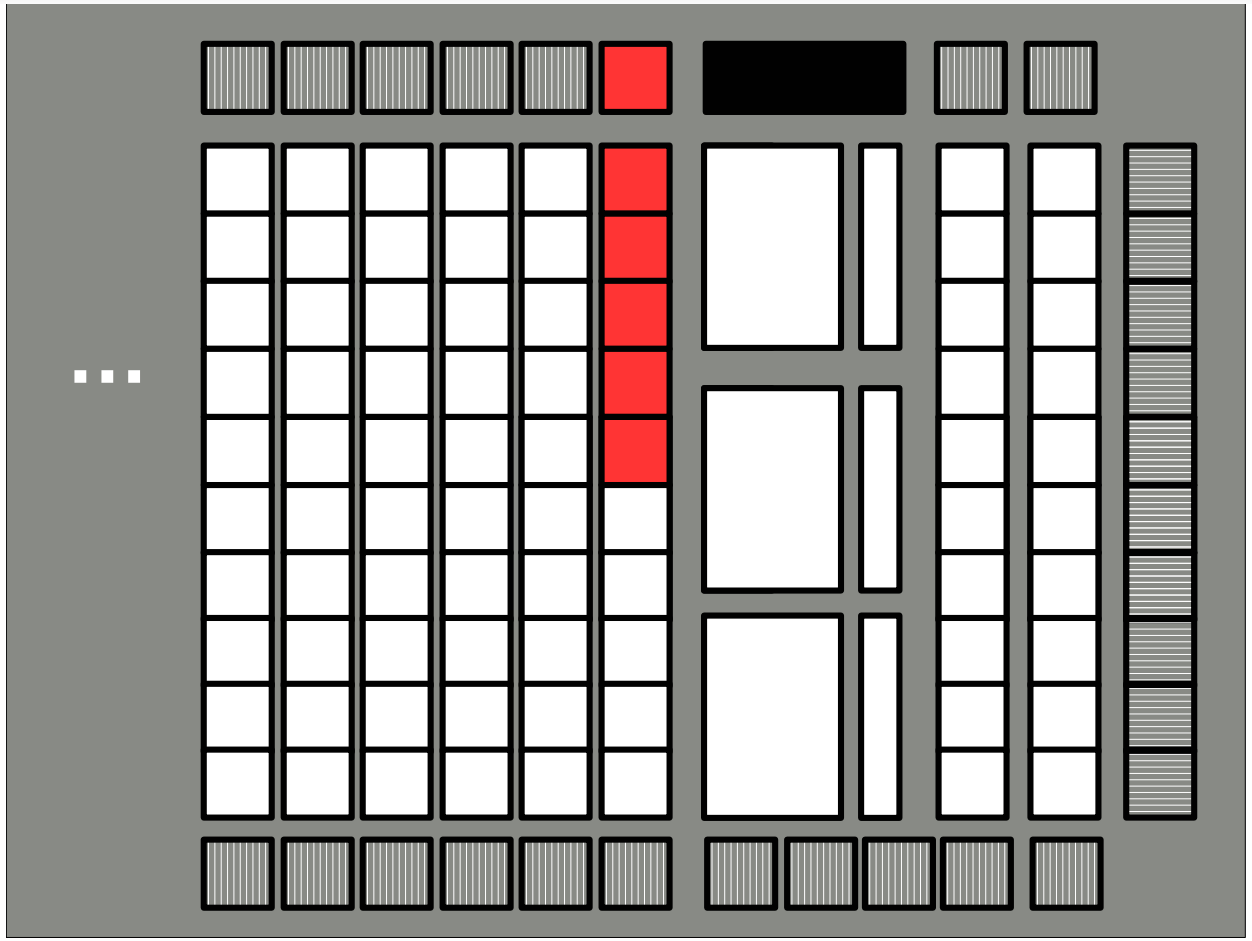
Partial Reconfiguration IV



For older generation : the whole column must be selected to create a partial reconfiguration module

UNIVERSITÉ
de Cergy-Pontoise

Partial Reconfiguration V



Newer generations of FPGAs allow to use a rectangular area, with a fixed minimal number of rows

UNIVERSITÉ
de Cergy-Pontoise

Some metrics

- Reconfiguration speed
  - The reconfigurable time is a function of the size and the organization of the partial reconfigurable regions
  - Modern FPGAs allow for arbitrary-shaped regions (a.k.a. rectangular shape)

UNIVERSITÉ
de Cergy-Pontoise

Bitstream

- **Useful documentation:** 7 Series FPGAs Configuration, UG470, p104
https://www.xilinx.com/support/documentation/user_guides/ug470_7Series_Config.pdf
https://www.kc8apf.net/2018/05/unpacking-xilinx-7-series-bitstreams-part-1/

- **File formats:**
  - .BIT Binary file containing BIT header followed by raw bitstream
  - .RBT ASCII file with text header followed by raw bitstream written as literal '0' and '1' characters for each bit
  - .BIN Raw bitstream
  - .MCSPROM file format (includes address and checksum info)

  Even though a BIN contains all the necessary data for programming a part, BIT is the default format generated by Vivado's *write_bitstream* command and is what I'll focus on.

- **Bit Header:**
  - Information (design name, build date/time, target part name) usually ignored by the chip. Such information is required by other programming tools (Vivado, openocd)

- **Bitstream file**

UNIVERSITÉ de Cergy-Pontoise

Bitstream

- **Configuration packets**
  - All data formats are described in 32-bit, big-endian words
  - All 7 series FPGA bitstreams commands are executed by reading or writing to the configuration registers
  - The FPGA bitstream consists of two packets types: Type 1 & Type 2

Type 1 packet Header (followed by data)

Header Type	Opcode	Register Address	Reserved	Word Count
[31:29]	[28:27]	[26:13]	[12:11]	[10:0]
001	xx	RRRRRRRRRxxxxx	RR	xxxxxxxxxx

Opcode	Name
00	NOP
01	Read
10	Write
11	Reserved

Only 5 out of 14 register address bits are used in 7 series FPGAs

⇒ Define a control register

Name	Read/Write	Address	Description
CRC	Read/Write	00000	CRC Register
FAR	Read/Write	00001	Frame Address Register
FDRI	Write	00010	Frame Data Register, Input Register (write configuration data)
FDRO	Read	00011	Frame Data Register, Output Register (read configuration data)
CMD	Read/Write	00100	Command Register
CTL0	Read/Write	00101	Control Register 0

...

UNIVERSITÉ
de Cergy-Pontoise

Bitstream

- **Frame Address Register I**
  - The 7 series devices are divided into two halves, the top and the bottom. All frames in 7 series devices have a fixed, identical length of 3,232 bits (101 32-bit words).
  - The Frame Address Register (FAR) is divided into five fields:

Address Type	Bit Index	Description
Block Type	[25:23]	Valid block types are CLB, I/O, CLK (000), block RAM content (001), and CFG_CLB (010). A normal bitstream does not include type 011.
Top/Bottom Bit	22	Select between top-half rows (0) and bottom-half rows (1).
Row Address	[21:17]	Selects the current row. The row addresses increment from center to top and then reset and increment from center to bottom.
Column Address	[16:7]	Selects a major column, such as a column of CLBs. Column addresses start at 0 on the left and increase to the right.
Minor Address	[6:0]	Selects a frame within a major column.

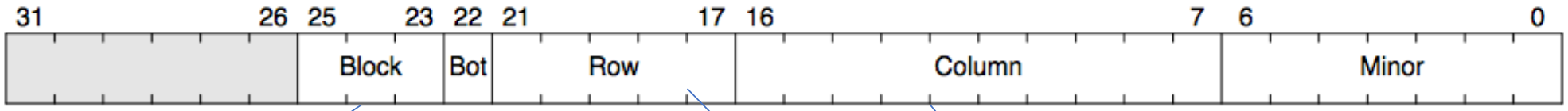
The address can be written directly or can be auto-incremented at the end of each frame. The typical bitstream starts at address 0 and auto-increments to the final count.

Source : https://www.xilinx.com/support/documentation/user_guides/ug470_7Series_Config.pdf

UNIVERSITÉ de Cergy-Pontoise

Bitstream

- **Frame Address Register II**



Xilinx 7-Series Frame Address Register

Code	Block Type
000	CLB,I/O,Clock
001	BRAM content
010	CFG_CLB

Bottom/Up regarding a center-line

Row and column index (0 on left)

Minor frames are contained within a column

Source : https://www.kc8apf.net/2018/05/unpacking-xilinx-7-series-bitstreams-part-2/

UNIVERSITÉ
de Cergy-Pontoise

Bitstream

- **Configuration packets**

Type 2 packet Header (must follow a type 1 packet)

⇒ used to write long blocks. No address is presented here because it uses the previous Type 1 packet address. The header section is always a 32-bit word.

⇒ Following the Type 2 packet header is the Type 2 Data section, which contains the number of 32-bit words specified by the word count portion of the header

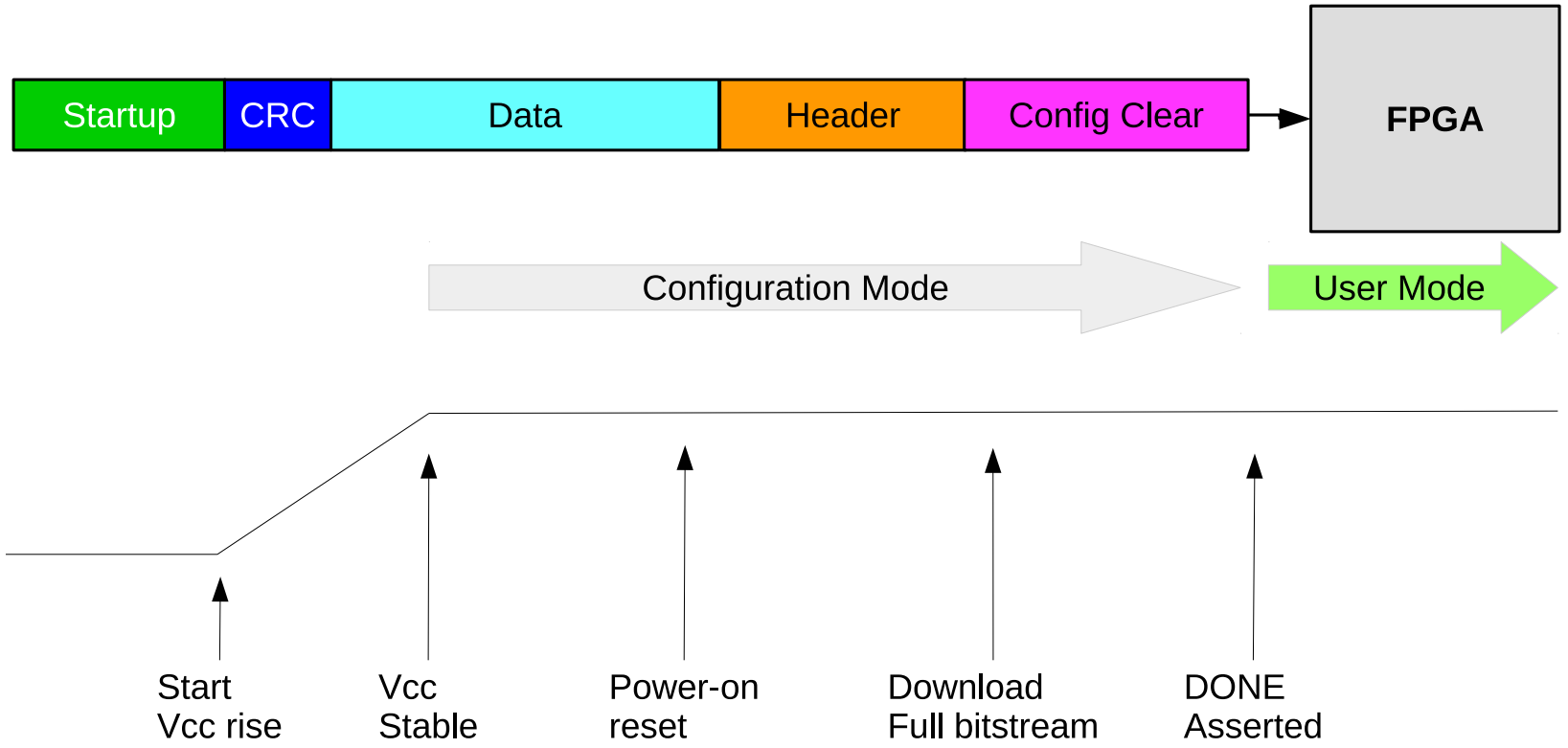
Header Type	Opcode	Word Count
[31:29]	[28:27]	[26:0]
010	XX	xxxxxxxxxxxxxxxxxxxxxxxxxx

Number of data word to transmit

UNIVERSITÉ
de Cergy-Pontoise

Download Bitstream

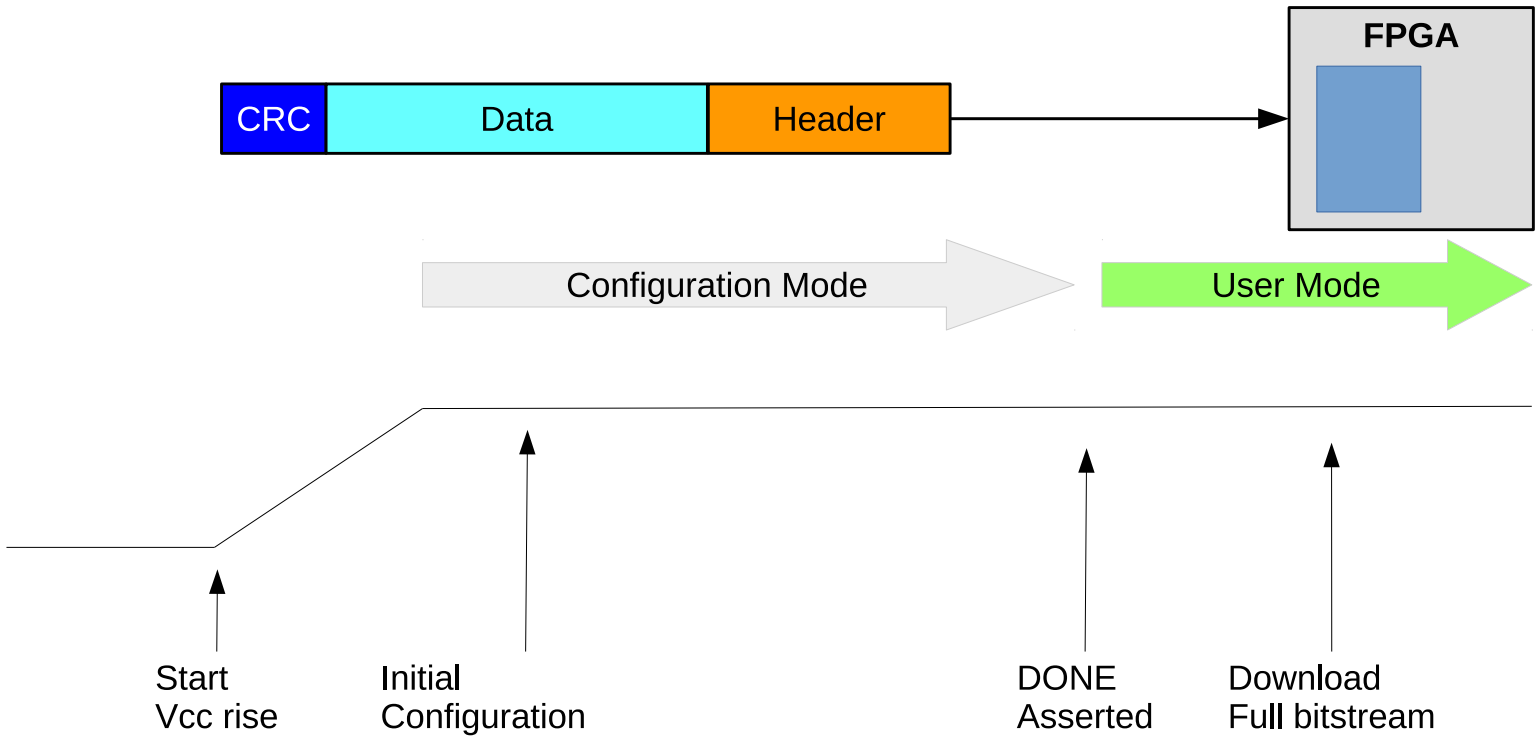
- Downloading bit files to FPGA



UNIVERSITÉ
de Cergy-Pontoise

Download Bitstream

- Downloading **partial** bit files to FPGA



UNIVERSITÉ de Cergy-Pontoise

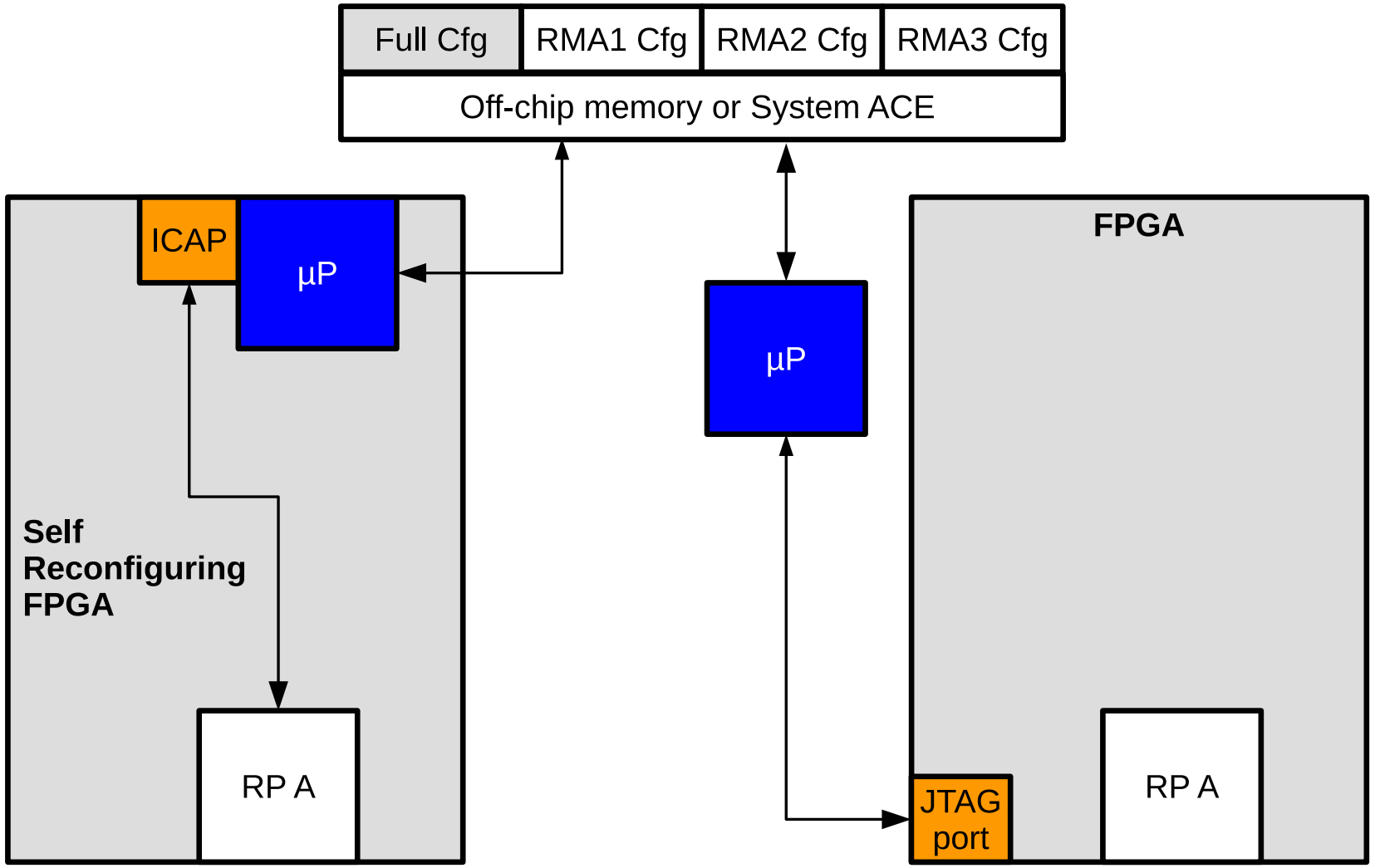
Xilinx PR Interface

- The Partial Reconfiguration control logic can either reside in an **external processor** or in the **fabric of the FPGA** device to be reconfigured.
  - **Externally:** Uses Serial configuration port, JTAG port, Processor Configuration Access Port (PCAP), or MCAP*
  - **Internally:** Uses Internal configuration access port (**ICAP**), Internal configuration can consist of either a custom state machine, or an embedded processor such as MicroBlaze™ processor.

The MCAP is a dedicated link to the ICAP from one specific PCIe block (Ultrascale only,)

UNIVERSITÉ
de Cergy-Pontoise

PR Overview



UNIVERSITÉ de Cergy-Pontoise

Methodology

- Two major ways to perform PR
  - **Module-based PR**
    - Implement reconfigurable modules separately
    - Constrain components to be placed at given location
    - Complete bitstream built as sum of all partial bitstreams
  - **Difference-based PR**
    - Implement complete bitstreams separately
    - Implement fixed-parts and reconfigurable parts with components constrained at same location within all bitstreams
    - Compute the difference of two bitstreams to obtain partial bitstream needed to move from one configuration to the next context

Focus on this type in this course

UNIVERSITÉ
de Cergy-Pontoise

Methodology

- Component enabling PR
  - Basic architectural elements
    - Fixed physical implementation of part of function
    - Stable interface:
      - Inputs specifically targeted at changing functionality
      - Inputs to be manipulated by the component

UNIVERSITÉ de Cergy-Pontoise

Methodology

- Configuration frame (Xilinx-specific)
  - Smallest addressable segments of FPGA configuration memory space
  - 1 frame = {CLB, BRAM, DSP}
    - On more modern/complex FPGAs (UltraScale, etc.), more components may be part of frames. . .
  - Bitstream files specify which logic to configure according to frame boundaries

	7-series	**UltraScale(+)**
CLB	50×1	60×1
DSP48	10×1	24×1
Block RAM	10×1	12×1
IOB / Clock	N/A	$52\,I/O \times 1$
Gb transceiver	N/A	4 high

UNIVERSITÉ
de Cergy-Pontoise

Methodology

- Specific components for Xilinx FPGAs
  - Partition – logical section of the design
    - Defined at hierarchical boundaries
  - Partition Definition (PD) – set of Partial Reconfiguration Modules (PRM) associated with a module instance
    - A PD is applied to all instances of the module
  - Bus macros – communication between modules
  - Partition pin – logical & physical connections between static and PR logic
    - Replace bus macros…
  - Partial reconfigurable module (PRM) – netlist or HDL description implemented within reconfiguration partition
  - Static logic – top-level logic, never partially reconfigured
    - Active during PR

Methodology

❑ Specific components for Xilinx FPGAs II

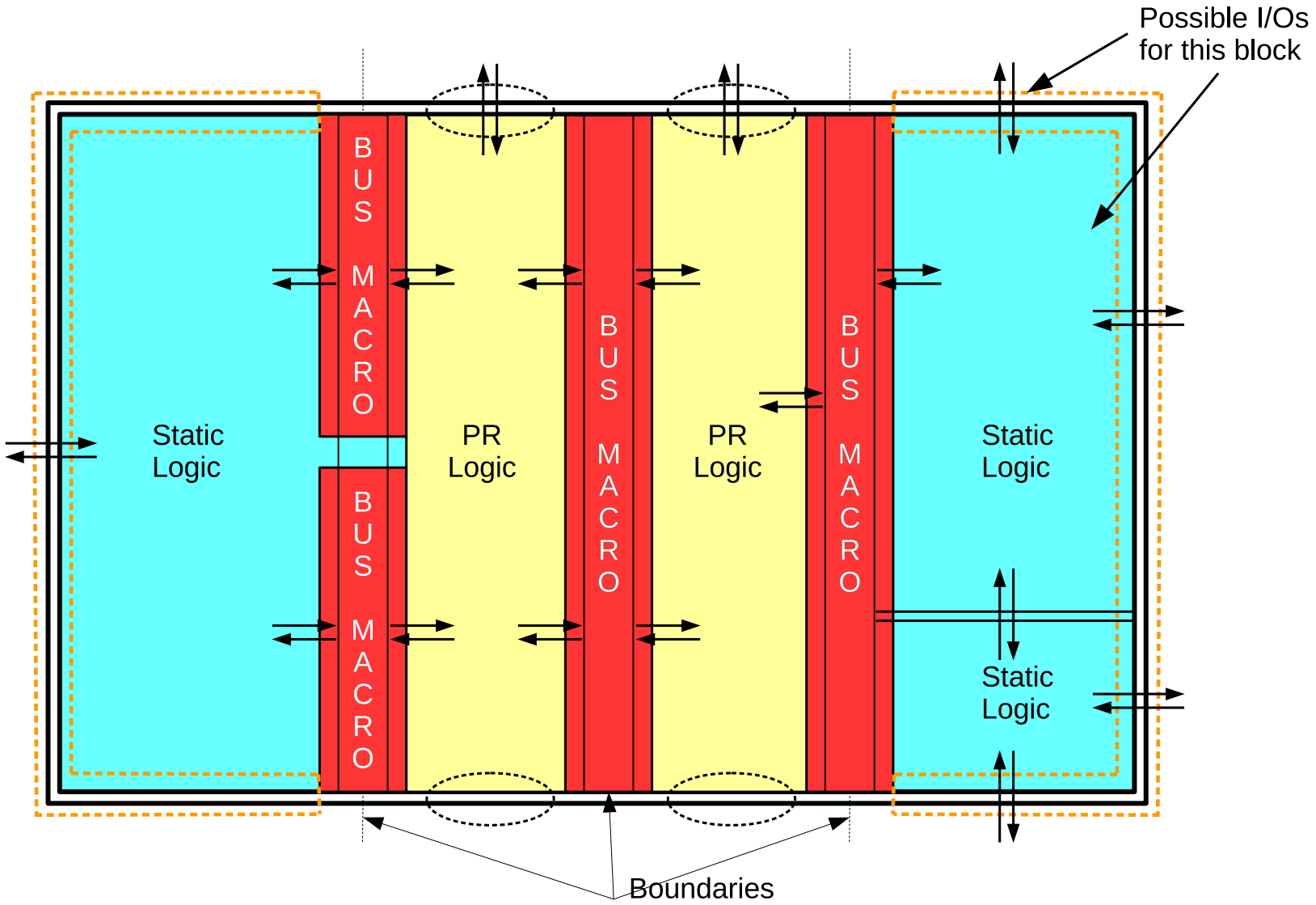
- Static design– part of design that doesn't change during PR
  - ➢ Includes top-level & all modules not defined as reconfigurable
  - ➢ Static design "=" static logic + static routing

- Physical block (pblock) – used to group logic of a reconfigurable module
  - ➢ Rectangular area with constraints
  - ➢ Motivation: minimize routing between pblocks
  - ➢ One pblock should represent less than 20% of the total floor plan

UNIVERSITÉ
de Cergy-Pontoise

Methodology

- Bus macro I



UNIVERSITÉ
de Cergy-Pontoise

Methodology

❑ Bus macro II

Bus macros provide guaranteed fixed-communication channels among PRMs and static modules

- Designer must ensure signals are routed on the right paths after reconfiguration
- Routing resources must remain static for inter-module communications
- HDL code must be written so that only bus macros are used to have modules communicating with each other
- Each macro: 4-bit inter-communication channel
  - ➢ E.G., to enable a 16-bit communication channel, 4 bus macros must be enabled.

UNIVERSITÉ de Cergy-Pontoise

Methodology

❑ Bus macro III -> Partition pins in current FPGAs

New Xilinx FPGAs and tools do not require the designer to explicitly specify bus macros at FPGA boundaries

- ➢ **Bus macros are replaced by partition pins**

  These are the physical interface points between static and reconfigurable logic. They are anchor points within an interconnect tile through which each I/O of the Reconfigurable Module must route (= virtual I/O between RM)

  - The tools do the work automatically
  - The user can create specific pin partitions when needed
    - ➢ Use HD.PARTPIN_RANGE to do so

- ➢ Modules must still communicate through adjacent IOBs

UNIVERSITÉ de Cergy-Pontoise

Methodology

- Physical blocks (pblocks)
  - Snapping mode: ensures that drawn pblocks are legal (will go to the right row/column boundaries)
  - Partitions make it harder to optimize the overall design
    - Don’t abuse the use of pblocks
  - Reconfigurable frames: pblocks ensure that specific layout constraints are met
  - When creating a pblock, one must ensure that
    - **PRM & RP are big enough to include the right amount of BRAM/DSP48 components**
    - **The most complex and/or biggest PRM is designed/outlined first**

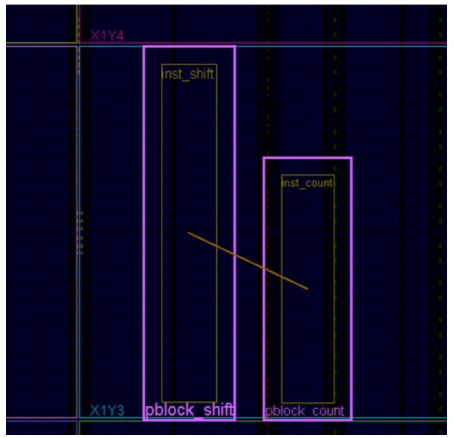
Methodology

❑ Physical blocks (pblocks)

- Reset after reconfig: *specific feature (7 series FPGA) allowing to held the reconfiguring region in a steady state during PR, and then all the logic of the RM is initialized to its starting value.*
  - ➢ Very important in synchronous design (FSM, …)
  - ➢ The user can specify this pblock feature using :

    Set_property RESET_AFTER_RECONFIG true [get_pblocks <reconfig_pblock_name>]

  - ➢ Pay attention to:
    - Pblock constraints must align to reconfigurable frames (vertically). Because the GSR affects every synchronous element within the region !



Left ok
Right ko

Using the SNAPPING_MODE constraint automatically creates legal, reconfigurable Pblocks

UNIVERSITÉ
de Cergy-Pontoise

Methodology

❑ Additional constraints on modular design

- The size & position of a design cannot be changed
- I/O blocks can only be accessed by contiguous modules
- PRMs/RMs can only communicate with neighbor modules
  - ➢ Must be done through bus macros /partition pins

➢ No global signals allowed

- ➢ E.G., global reset signals are forbidden
- ➢ Exception: clocks which use different bitstreams & routing channels

UNIVERSITÉ de Cergy-Pontoise

Vivado Software flow

❑Module-based PR using Vivado

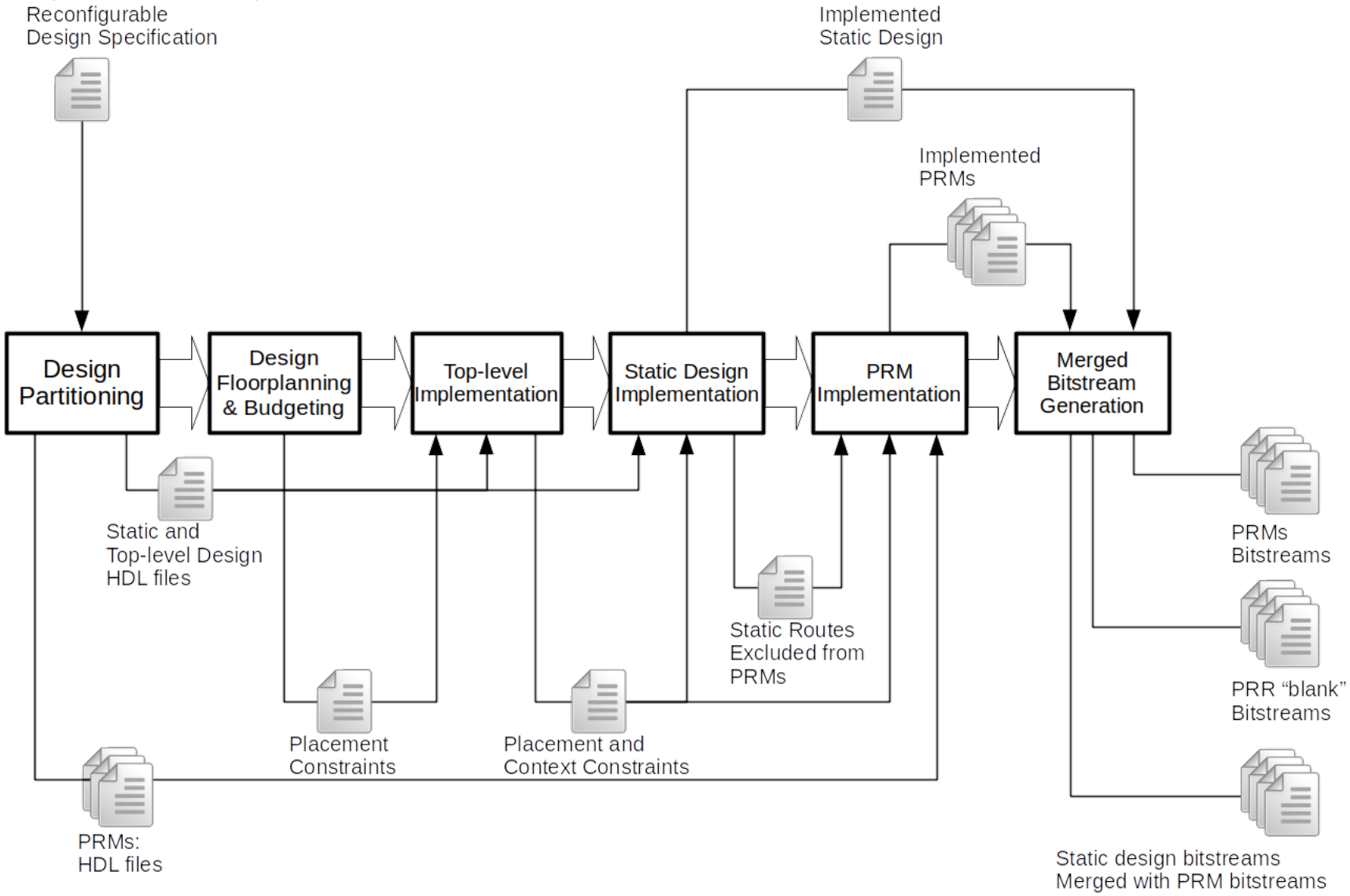
1. Synthesize the static &Reconfigurable Modules separately.
2. Create physical constraints (Pblocks) to define the reconfigurable regions.
3. Set the HD.RECONFIGURABLE property on each Reconfigurable Partition.
4. Implement a complete design (static and one Reconfigurable Module per Reconfigurable Partition) in context.
5. Save a design checkpoint (DCP) for the full routed design.
6. Remove Reconfigurable Modules from this design and save a static-only design checkpoint.
7. Lock the static placement and routing.
8. Add new Reconfigurable Modules to the static design and implement this new configuration, saving a checkpoint for the full routed design.
9. Repeat Step 8 until all Reconfigurable Modules are implemented.
10. Run a verification utility (pr_verify) on all configurations.
11. Create bitstreams for each configuration
    - ➢ One full bitstream for the static design
    - ➢ One partial bitstream for each PRM

https://www.xilinx.com/support/documentation/sw_manuals/xilinx2017_2/ug909-vivado-partial-reconfiguration.pdf

UNIVERSITÉ
de Cergy-Pontoise

Vivado Software flow

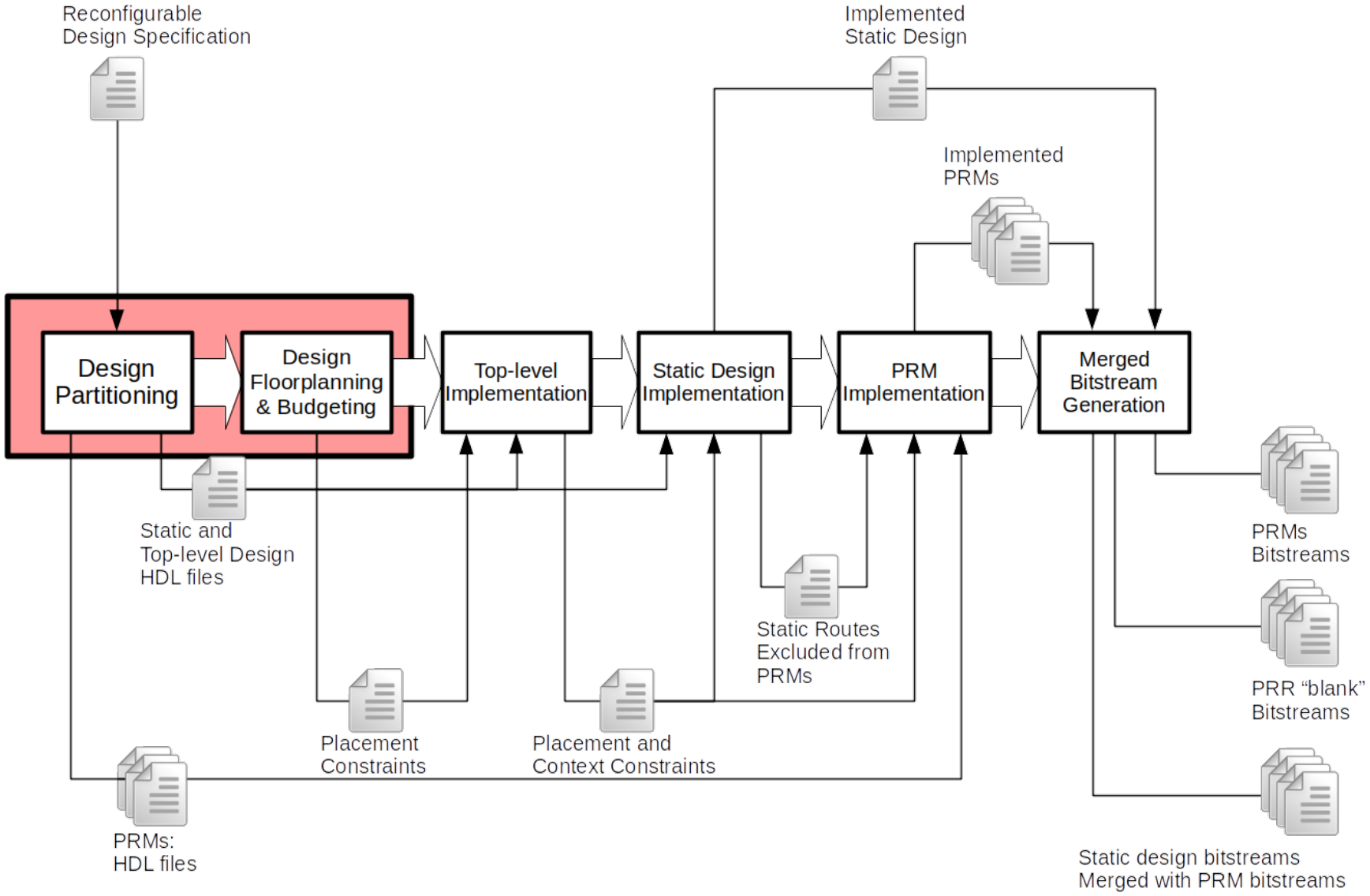
- A graphical representation of the PR flow



UNIVERSITÉ
de Cergy-Pontoise

Vivado Software flow

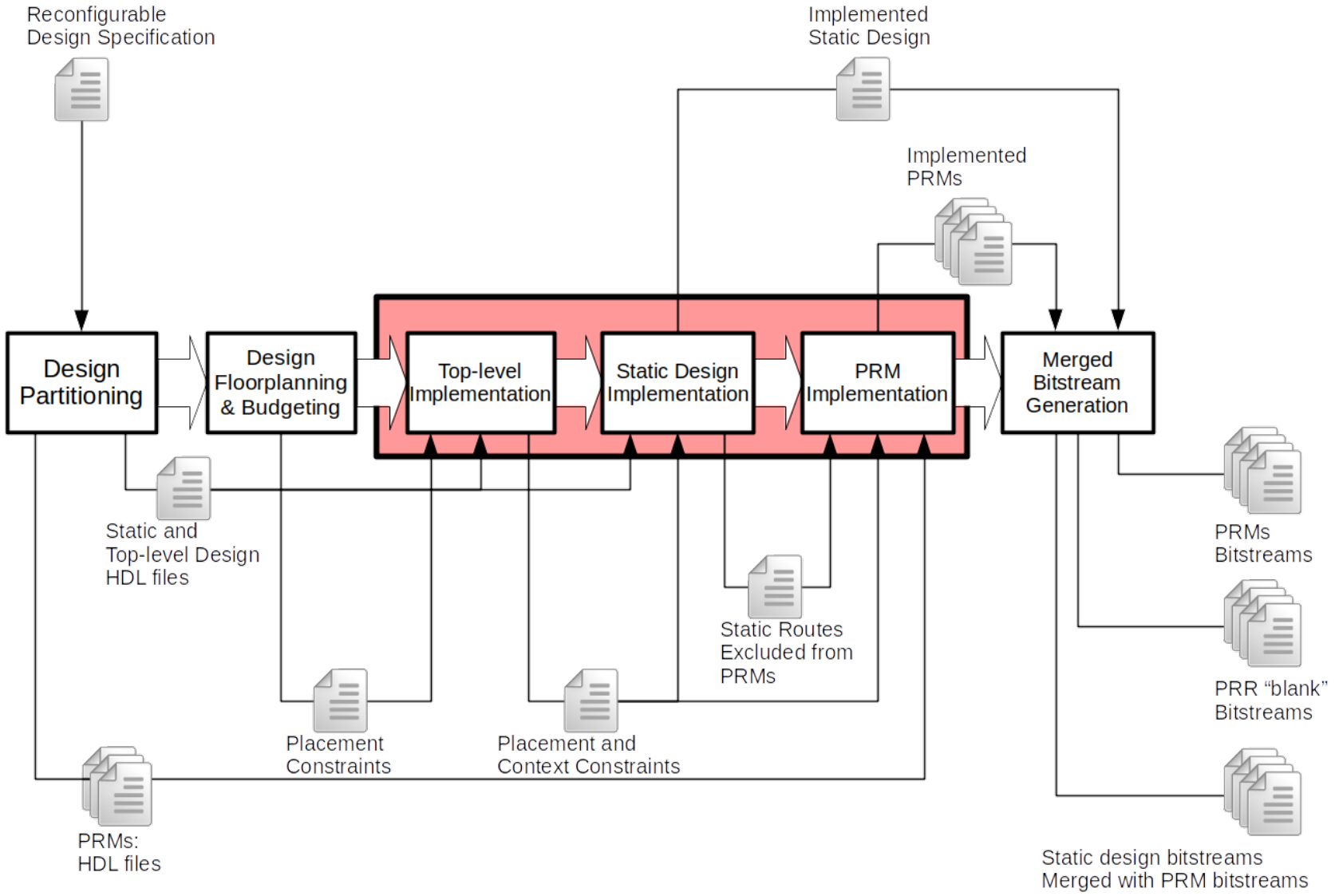
- A graphical representation of the PR flow



UNIVERSITÉ
de Cergy-Pontoise

Vivado Software flow

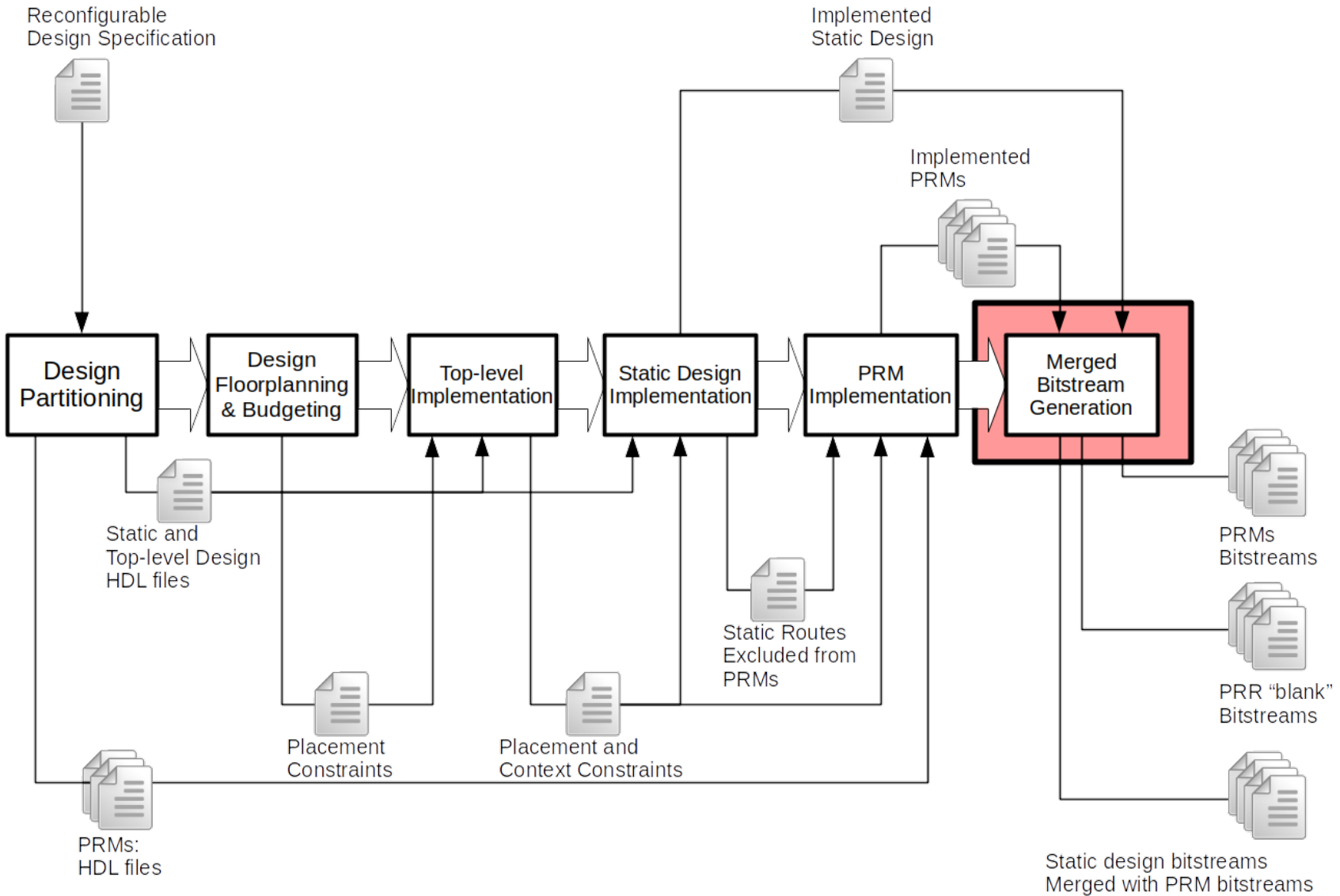
- A graphical representation of the PR flow



UNIVERSITÉ
de Cergy-Pontoise

Vivado Software flow

- A graphical representation of the PR flow



UNIVERSITÉ
de Cergy-Pontoise

Criteria

- Criteria for dynamic PR in FPGAs
  - Speeds and methods;
  - Design hierarchy limitations w.r.t. PR modules (PRMs)
  - Number of allowed PR regions (PRRs)
  - Software support to generate PR designs

UNIVERSITÉ
de Cergy-Pontoise

Difference-based PR I

- Overview
  - Small changes on the FPGA configuration
  - Manually done using the FPGA editor
  - Relies on full bitstream files
    - The reconfiguration occurs when the new file is selected for reconfiguration
    - A difference is computed between the current and the new file to see what to reconfigure

UNIVERSITÉ
de Cergy-Pontoise

Difference-based PR II

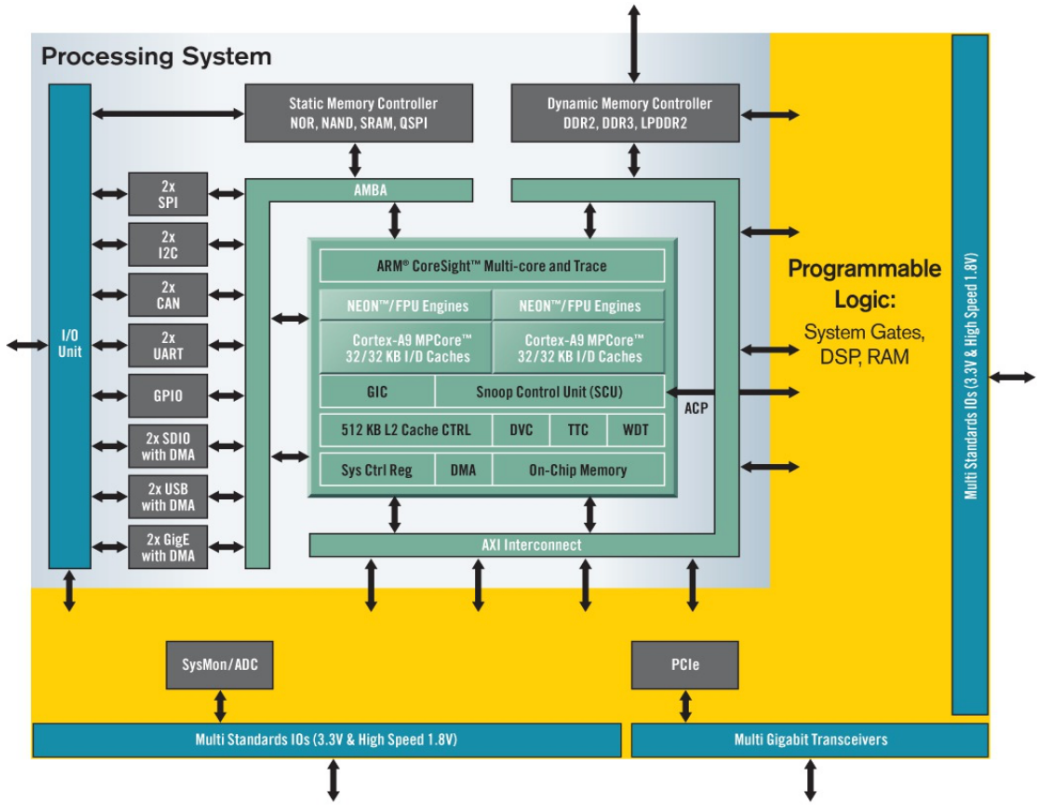
- What can be modified ?
  - LUT equations
  - BRAM contents and BRAM write modes
  - I/O standard pull-ups/pull-downs on external pins
  - MUXes to invert polarity
  - Flipflop initialization and reset values

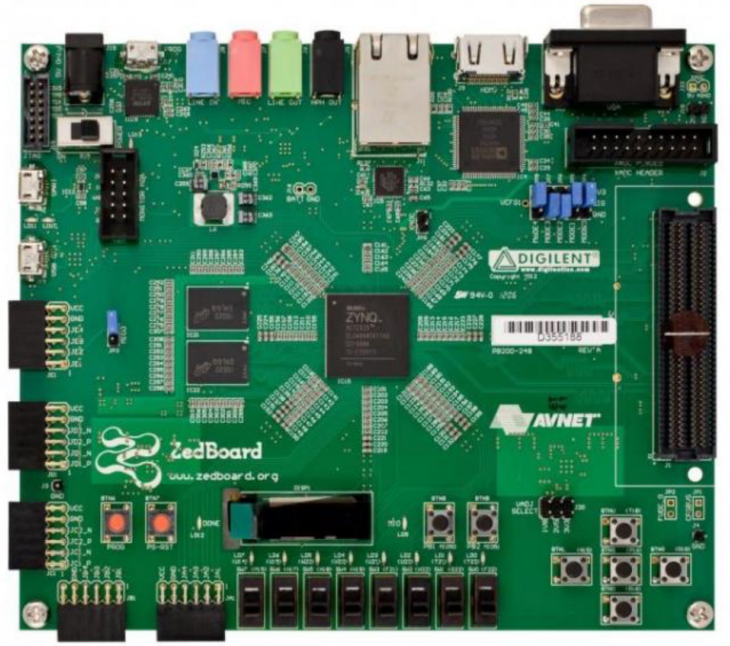
- What can NOT be modified ?
  - Routing—very dangerous: internal contention

UNIVERSITÉ
de Cergy-Pontoise

PRATICAL WORK

- During the TP,
  - Realization of the PR design flow using Vivado
  - Target platform is zedboard – Zynq-7000
  - Dynamic partial reconfiguration using ARM processor and PCAP interface

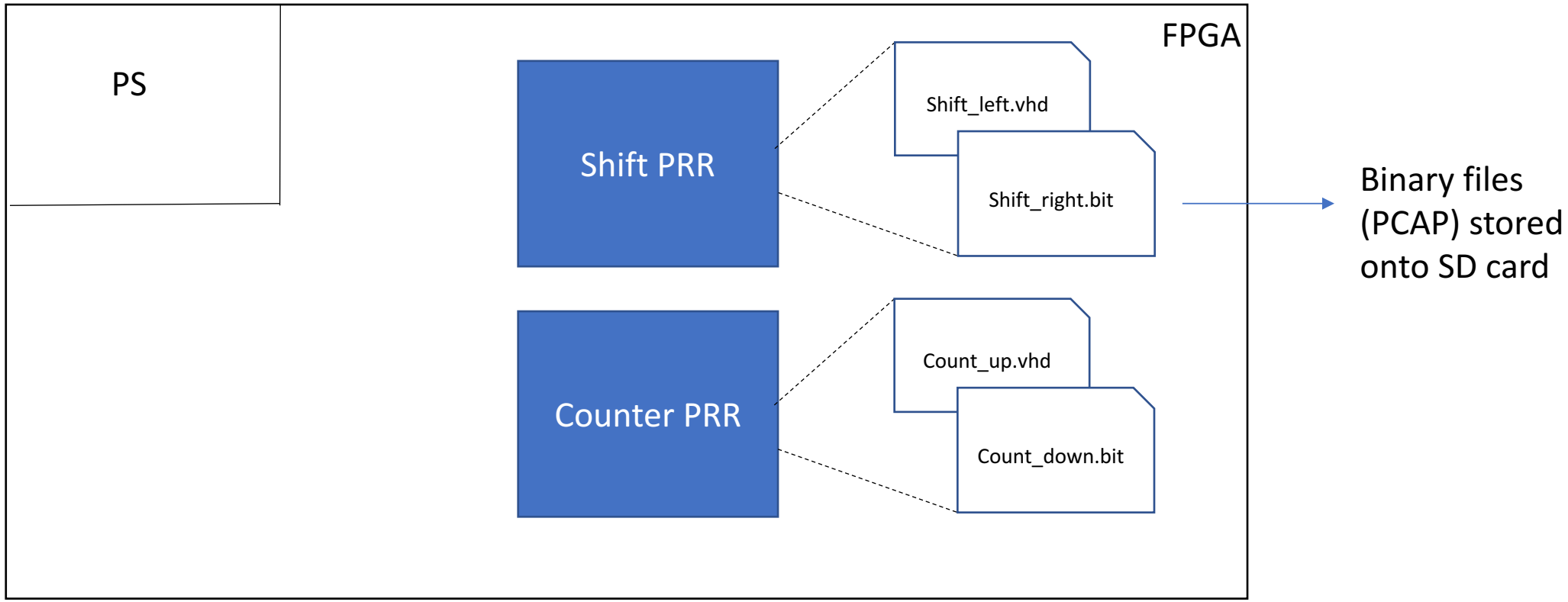




UNIVERSITÉ de Cergy-Pontoise

PRATICAL WORK

- During the TP, 2 PRRs and 4 bitstreams will be used
- DPR Management using PCAP and DevCFG
- Bitstreams (*.bin) onto SD card



UNIVERSITÉ
de Cergy-Pontoise

Conclusion

- Partial reconfiguration in FPGAs offers a flexible way to adapt a device to specific situations
- PR can be
  - ➢ static “\stop the world” while the system gets reconfigured, even partially
  - ➢ dynamic “only a target portion paused while it is being reconfigured” everything else continues to run;
- Advantages include
  - ➢ lower power consumption
  - ➢ better runtime adaptation
  - ➢ possibility to enhance fault-tolerance
  - ➢ etc.
- When considering dynamic PR, the designer must take into account reconfiguration times
  - ⇒ In real-time environments, one must ensure deadlines can be met i.e., reconfiguration time should be included when considering tasks latencies