Introduction to Partial Reconfiguration Methodology

Zynq
Vivado 2015.2 Version

Objectives

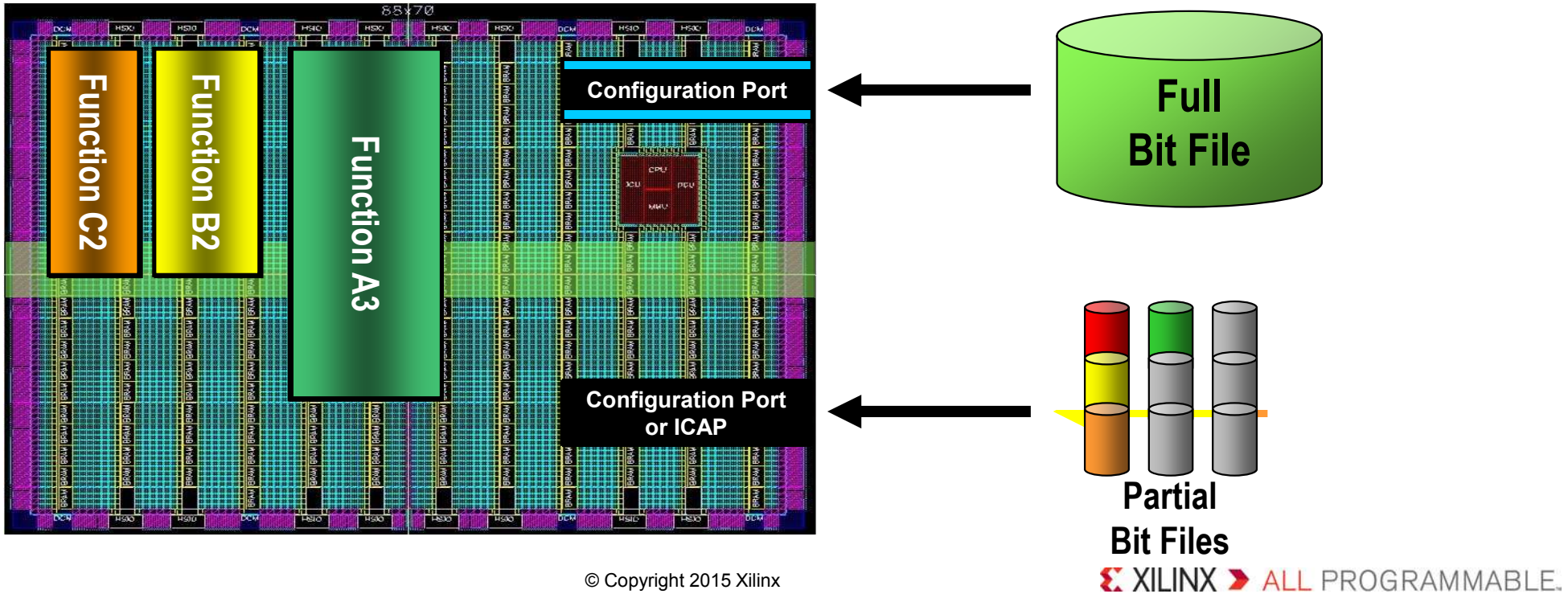
- **After completing this module, you will be able to:**
  - Define Partial Reconfiguration technology
  - List common applications for using Partial Reconfiguration
  - Define Partial Reconfiguration terminology
  - State the Partial Reconfiguration flow

Outline

- *What is Partial Reconfiguration(PR)?*
- PR Technology
- PR Terminology
- PR Design Flow
- Summary

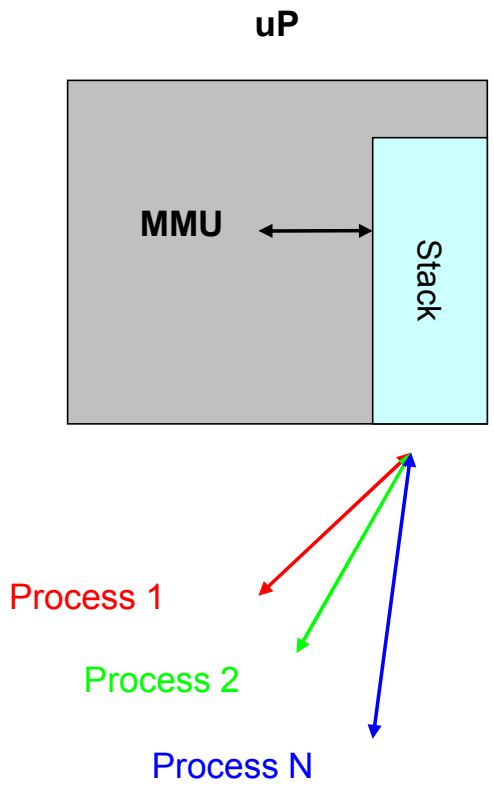
What is Partial Reconfiguration?

- Partial Reconfiguration is the ability to dynamically modify blocks of logic by downloading partial bit files while the remaining logic continues to operate without interruption.



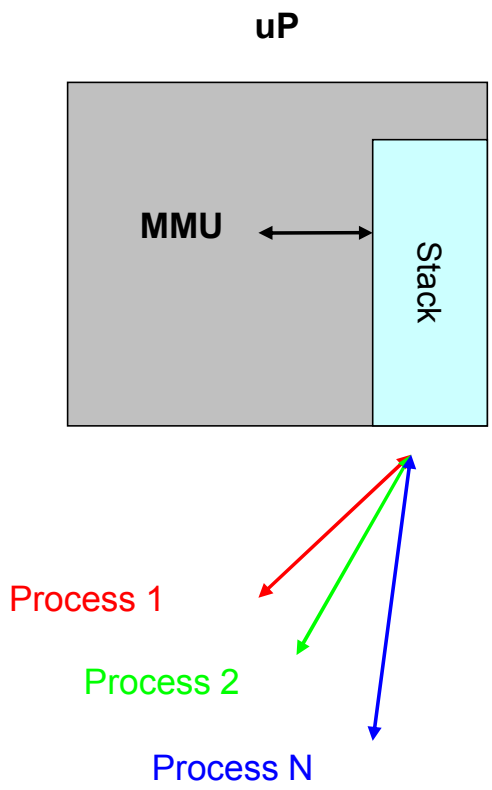
PR Applications Analogy

Processor Context Switch

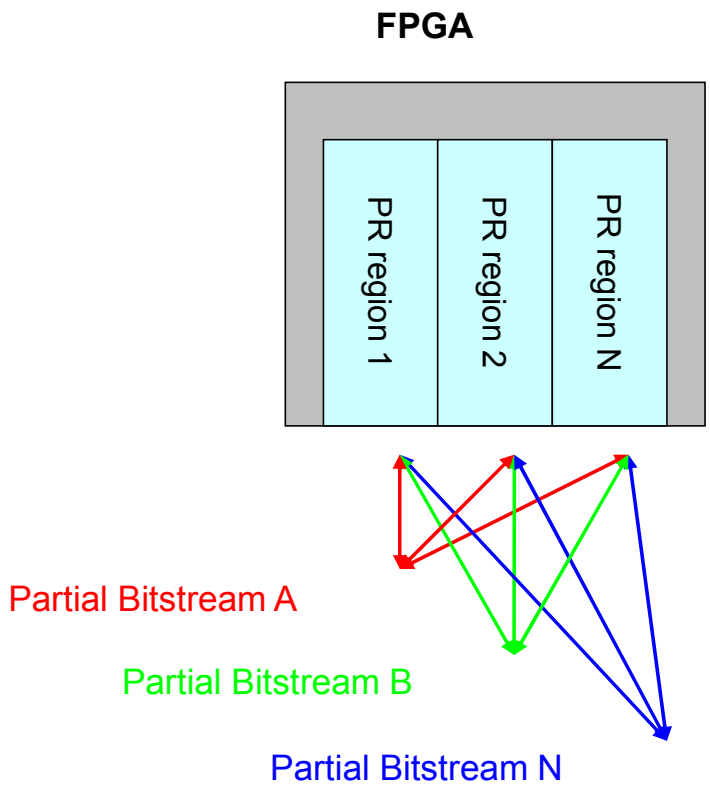


PR Applications Analogy

Processor Context Switch



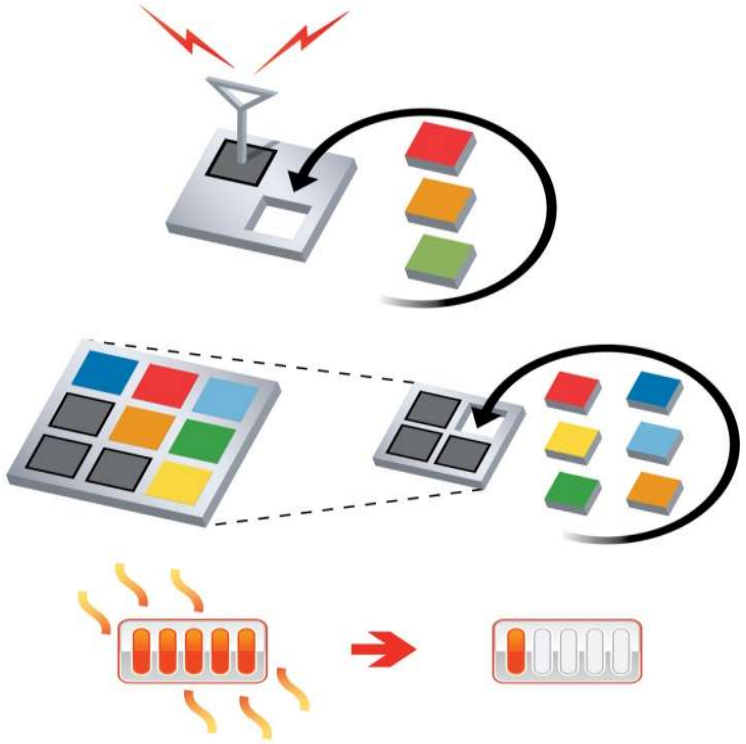
FPGA Configuration Switch



Partial Reconfiguration

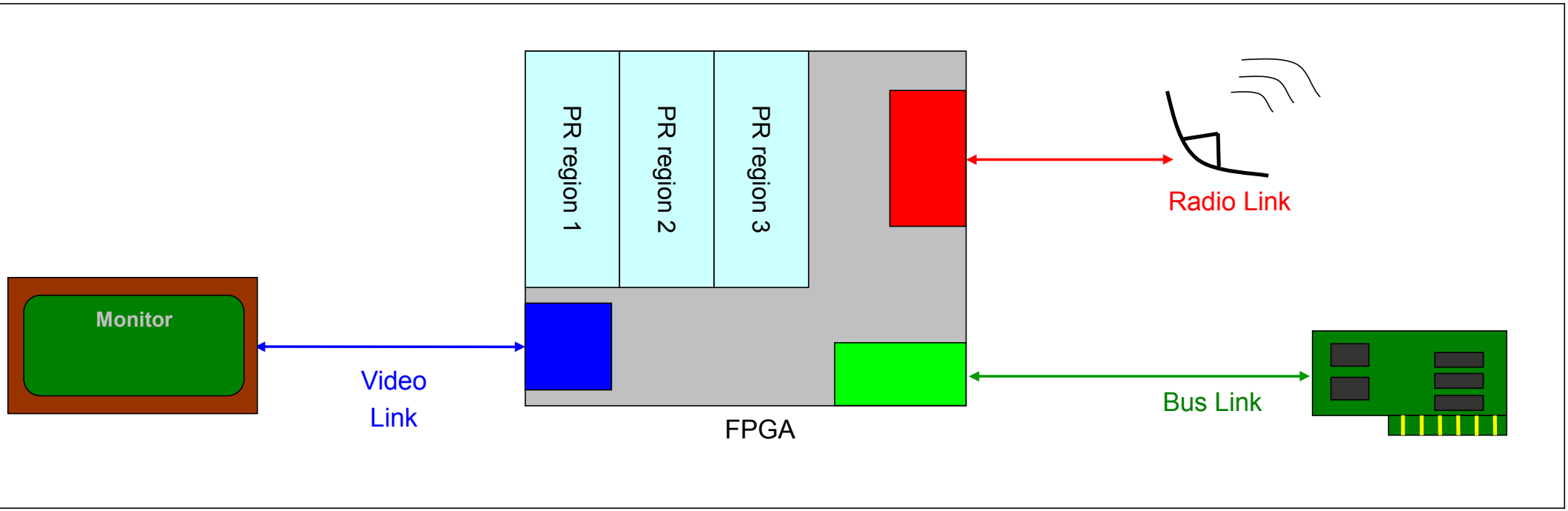
Technology and Benefits

- **Partial Reconfiguration enables:**
  - System Flexibility
    - Swap functions and perform remote updates while system is operational
  - Size and Cost Reduction
    - Time-multiplexing hardware requires a smaller FPGA
    - Reduces board space
    - Minimizes bitstream storage
  - Power Reduction
    - Via smaller or/and fewer devices
    - Swap out power-hungry tasks



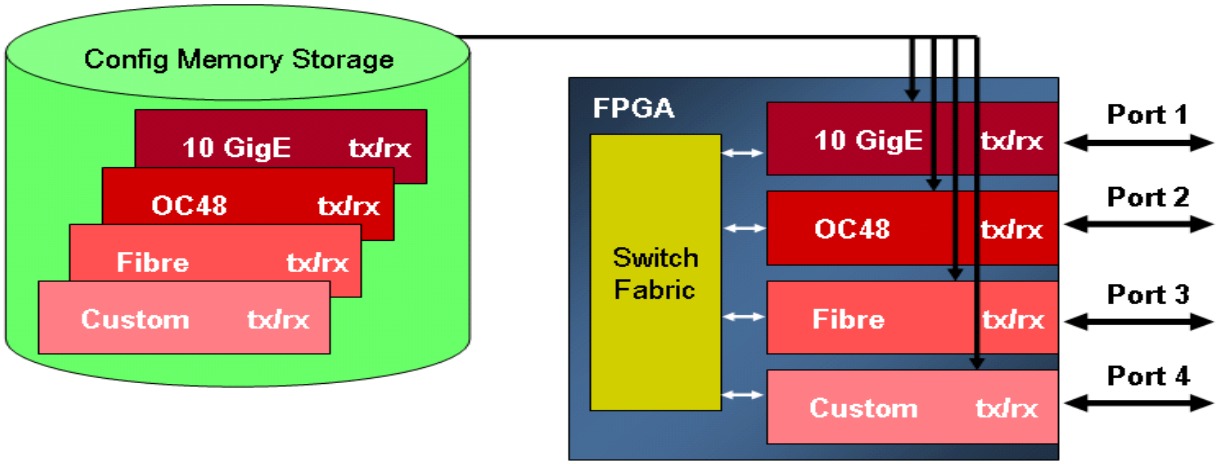
System Flexibility: Communication Hub

- **The FPGA can be a communications hub and must remain active**
  - Cannot perform full reconfiguration due to established links



Size and Cost Reduction: Time Multiplexing

- **Applications need to be able handle a variety of functions**
  - Supporting many at once can use a great deal of space
- The library of functions use case covers a wide number of applications
  - Time-based multiplexing of functions reduces device size requirement



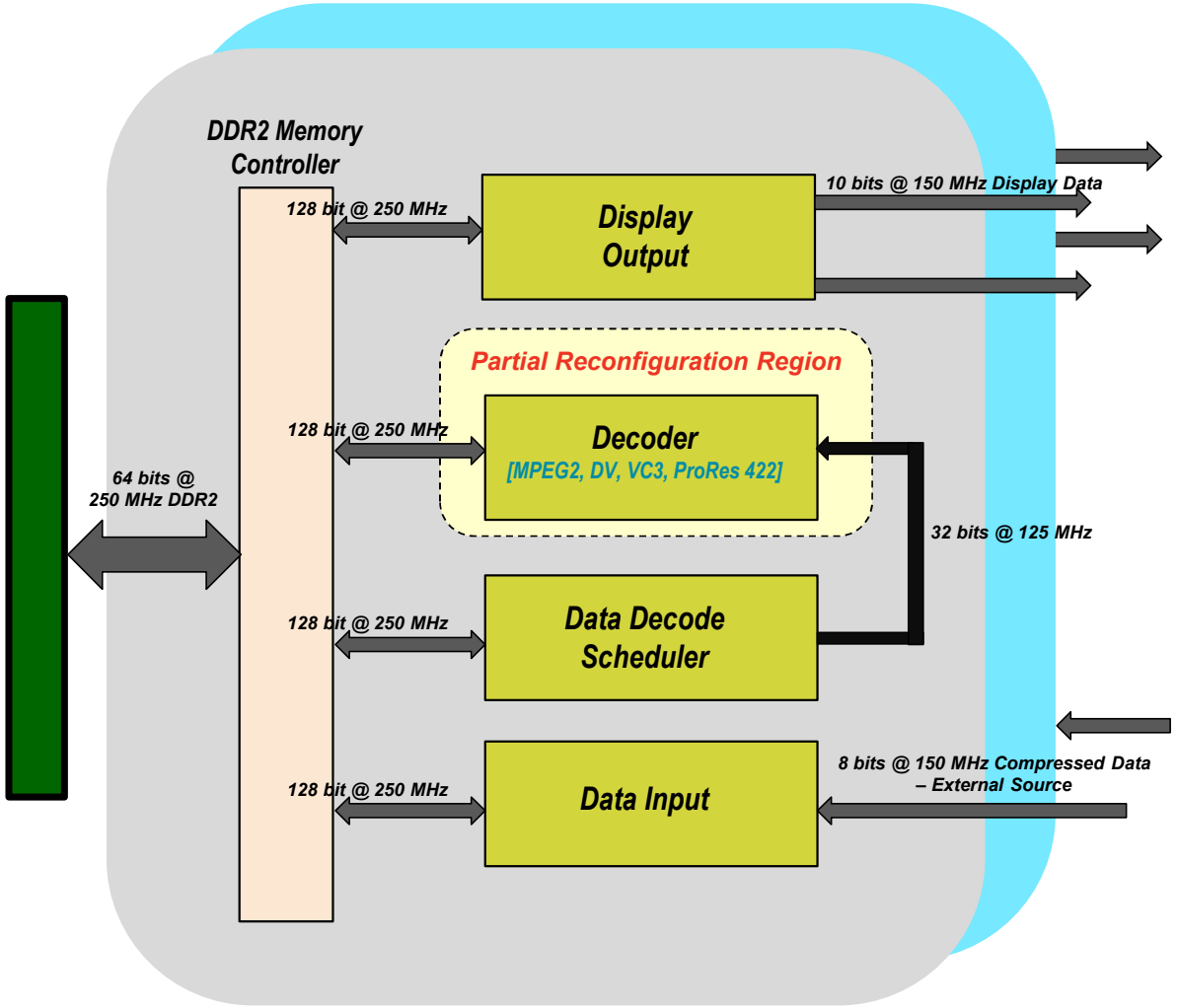
Power Reduction Techniques with PR

- **Board space and resources are limited**
  - Multi-chip solutions consume extra area, cost, and power
- **Many techniques can be employed to reduce power**
  - Swap out high-power functions for low-power functions when maximum performance is not required
  - Swap out black boxes for inactive regions
  - Swap high-power I/O standards for lower-power I/O when specific characteristics are not needed
  - Time-multiplexing functions will reduce power by reducing amount of configured logic

Customer Example

Flexible Video Processing

- **Swap decoders on the fly**
  - One channel remains up while the other changes
- **Released "flat" version first**
  - Two decoders per channel
- **Expanded functionality without changing hardware**
  - Deployed new bitstreams for more decoders without changing hardware



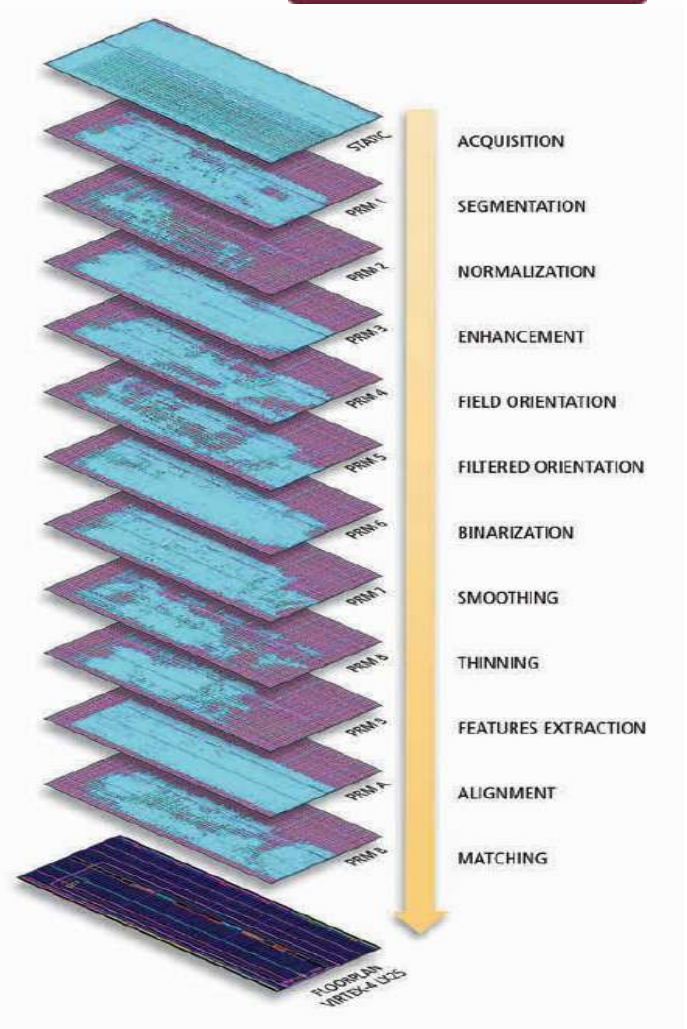
Customer Example

Sequential Processing

Universitat Rovira i Virgili
The public university of Tarragona

Read the detailed article in Xcell Issue 72

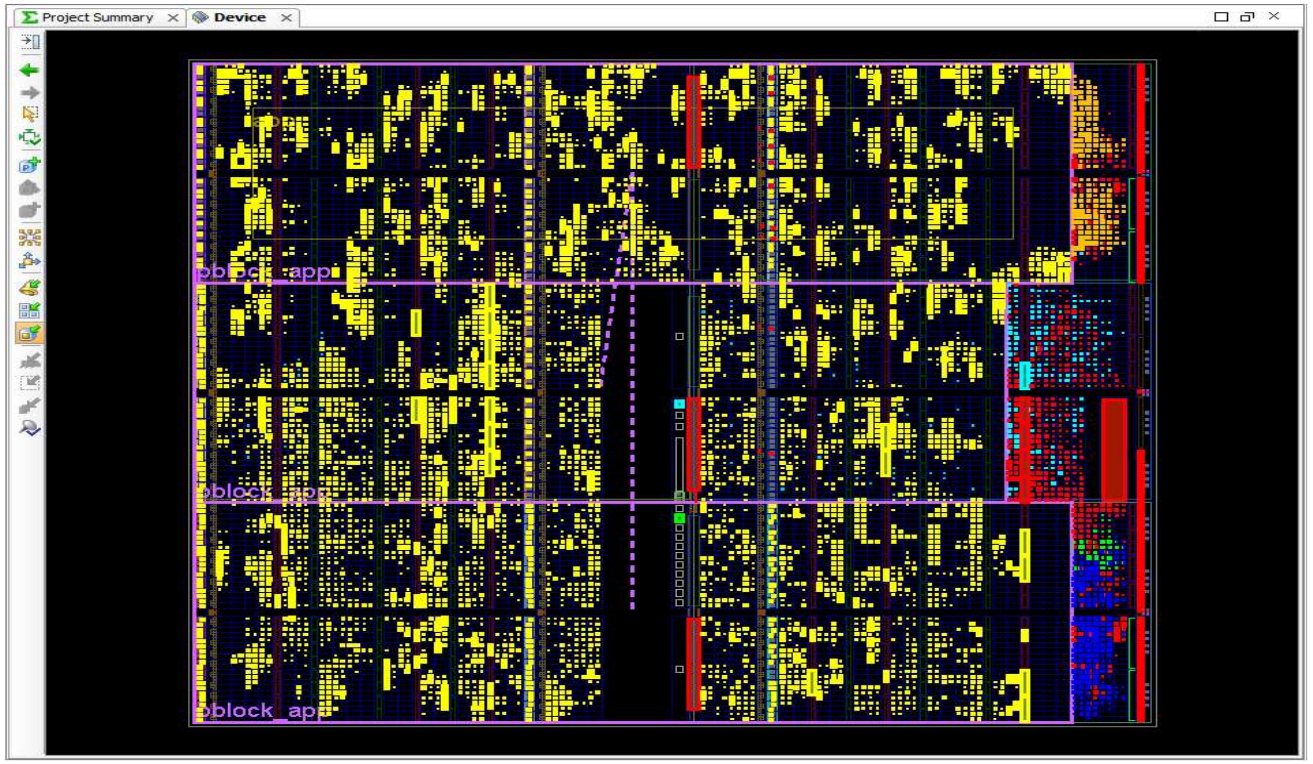
- **Automatic Fingerprint Authentication System**
  - Sequence of 12 different functions required
  - No two were needed at any one time
- **Load in functions on demand**
  - Steps could be skipped or revisited
- **Processing time greatly improved**
  - PC-based Software solution: 3.77 sec
  - Xilinx MicroBlaze solution: 143.19 sec
  - Xilinx Partial Reconfiguration: 0.7 sec
    - Includes 10ms total for all reconfiguration events
- **Efficient use of resources**
  - "Flat" solution requires 39k flops, 52k LUTs
  - PR solution uses region with less than 10k of each



Customer Example

Fast PCIe Configuration and Field Updates

- **Customer requirements:**
  - Reduce PROM size
  - Improve PCIe configuration time
  - Enable live field updates
- **Solution details:**
  - Config time reduced 73%
  - Static logic includes:
    - PCIe Gen2x1 core
    - Two 10G XAUI cores
    - AXI debug logic
    - ICAP loader

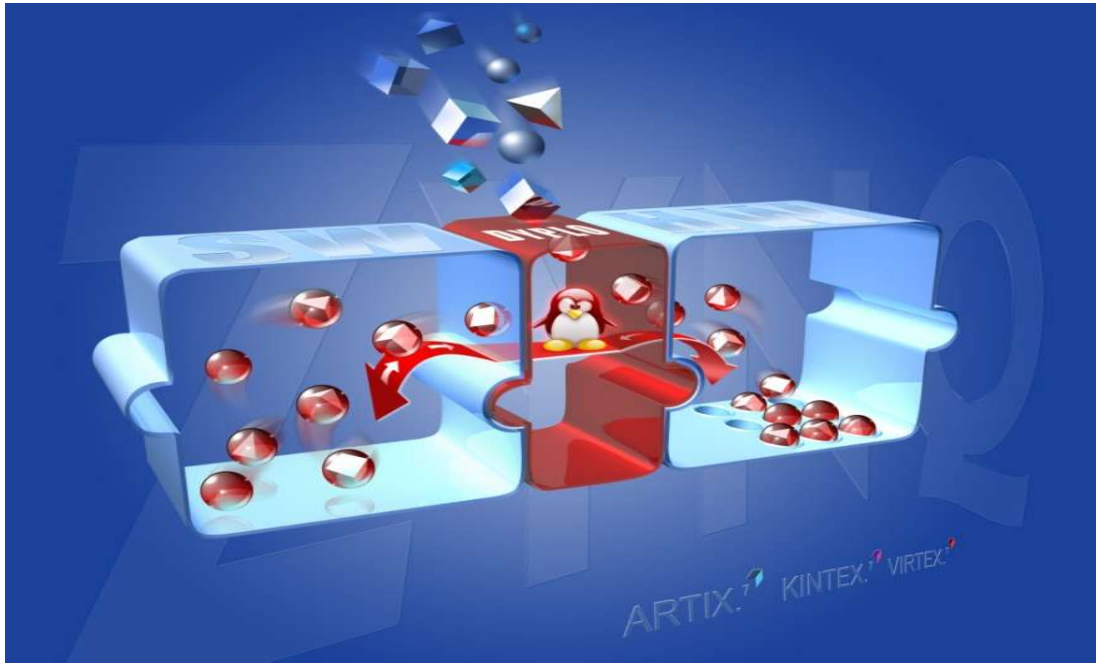


Customer Example

Hardware Acceleration

TOPIC EMBEDDED SYSTEMS

- **Dyplo = DYnamic Process LOader**
  - Solution distributes software functions between hardware and software spaces
- **Dyplo's unique selling points**
  - Optimized use of Zynq SoC device
  - Software driven Hardware development approach
  - Abstraction of implementation choices to system level
  - Simple use of partial reconfiguration blocks in hardware
  - Configuration Wizard tool to guarantee ease of use
- **Product launched in March 2015**
  - Read more at http://topic.nl/en/dyplo or Xcell issue 85

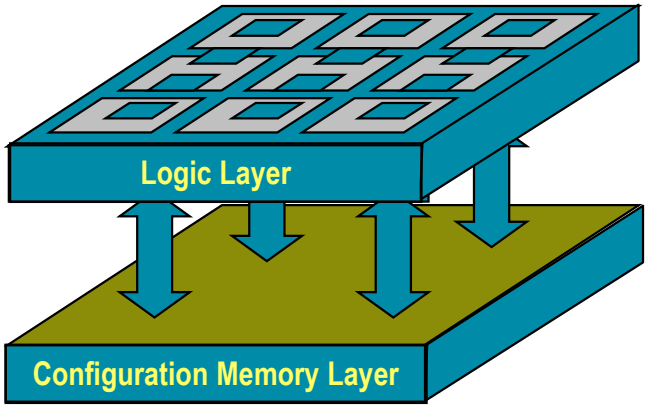


Outline

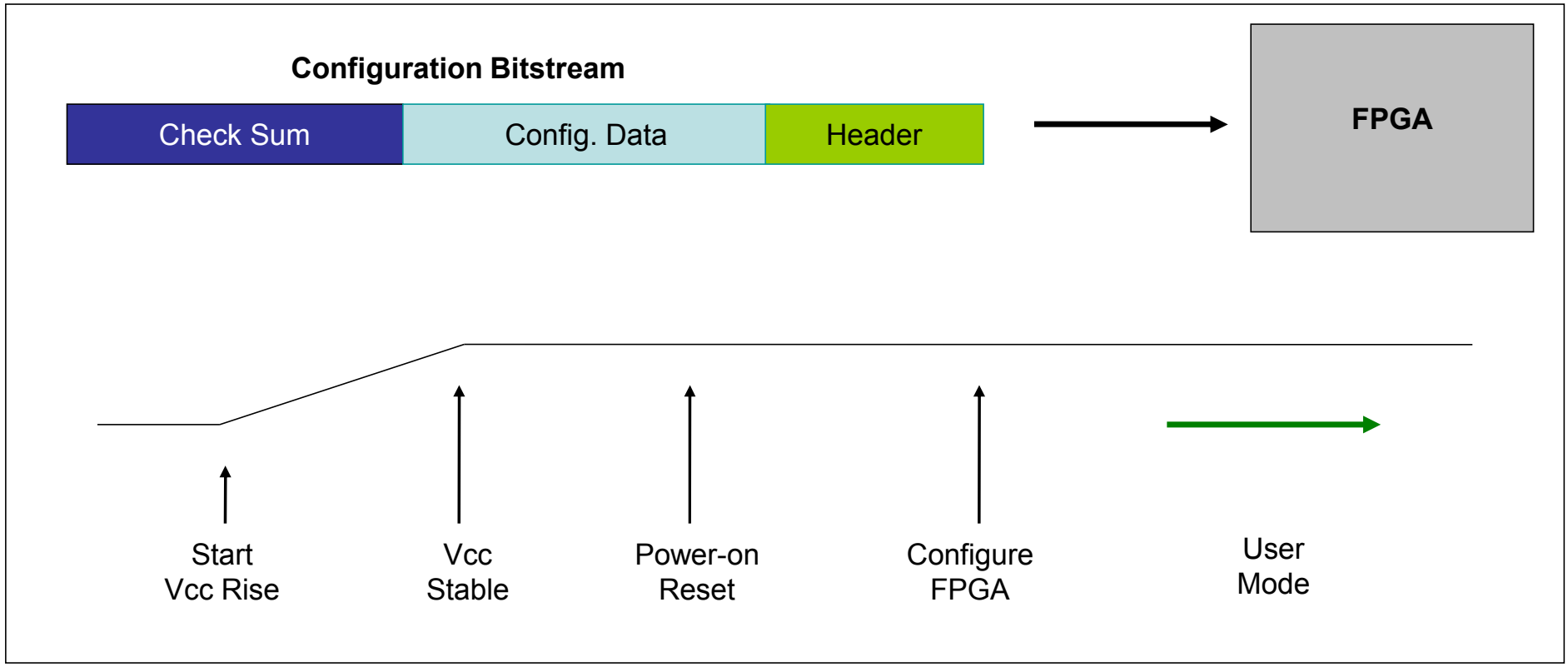
- What is Partial Reconfiguration(PR)?
- *PR Technology*
- PR Terminology
- PR Design Flow
- Summary

Programmability 101

- **Think of an FPGA as two layered device:**
  - Configuration memory layer
  - Logic layer
- **Configuration memory controls function computed on logic layer**

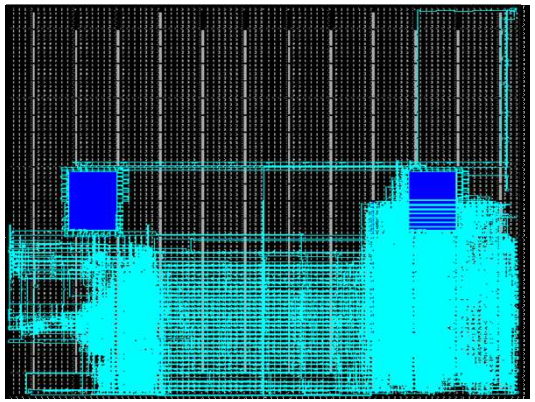


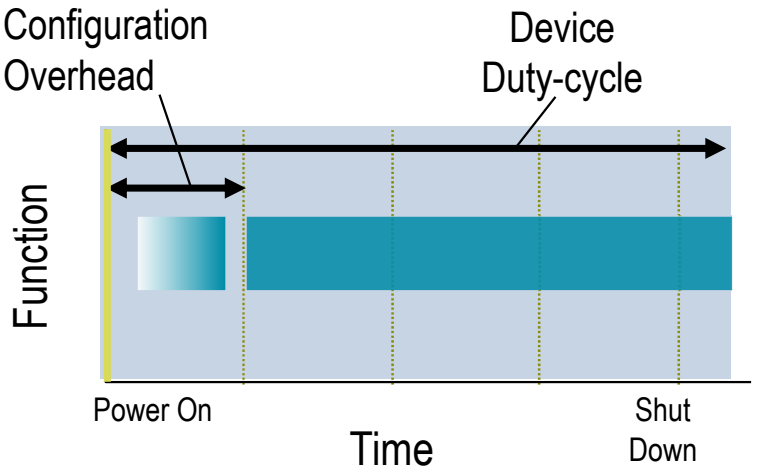
“Normal” Configuration



‘Typical’ Configuration Mode

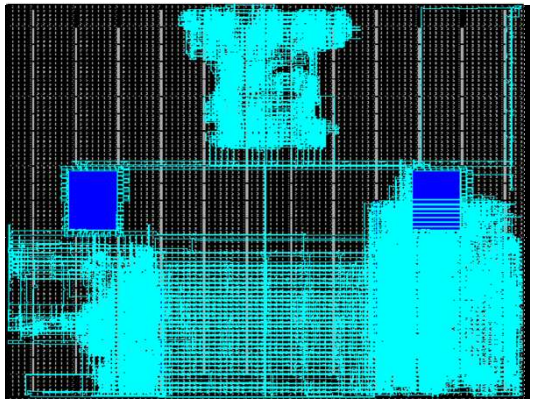
- **Fixed configuration**
  - Data loads from PROM or other source at power on
  - Configuration fixed until the end of the FPGA duty cycle
- **Used extensively during traditional design flow**
  - Evaluate functionality of design as it is developed

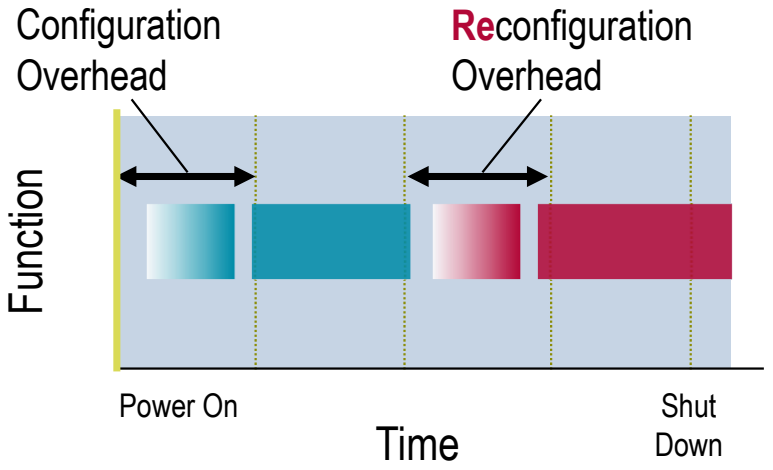




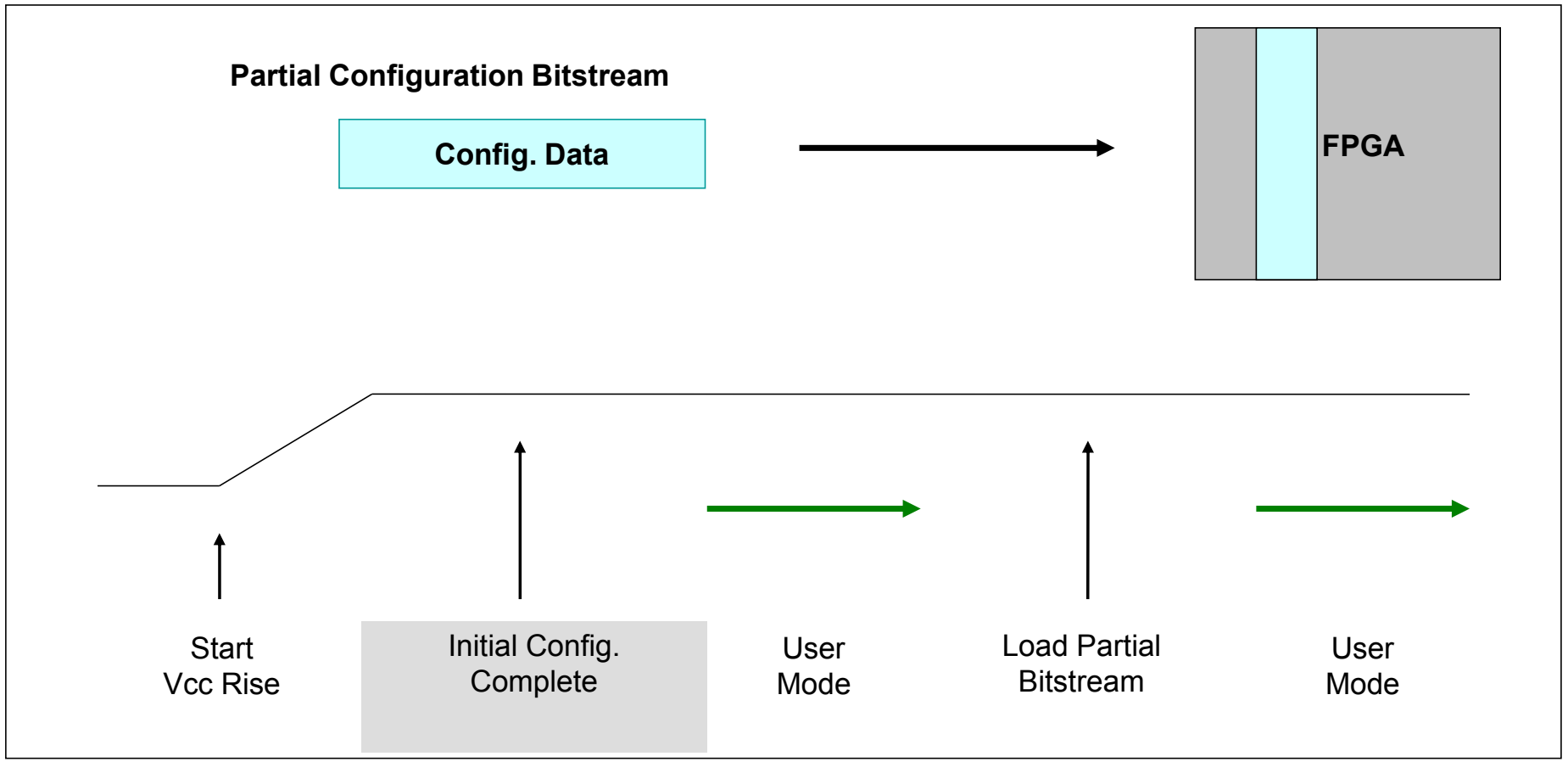
Reconfiguration

- Configuration memory is no longer fixed during the system duty cycle
- Initial bitstream loaded at power-on
- Different, full device bitstreams loaded over time



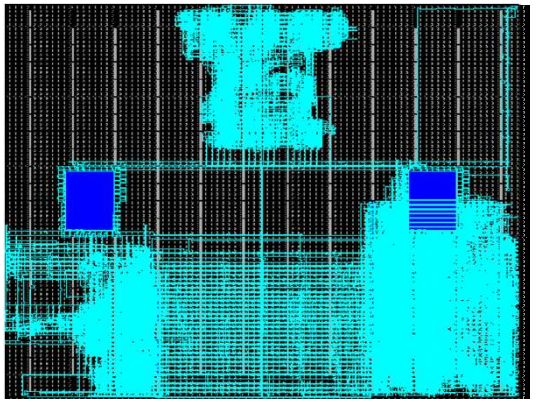


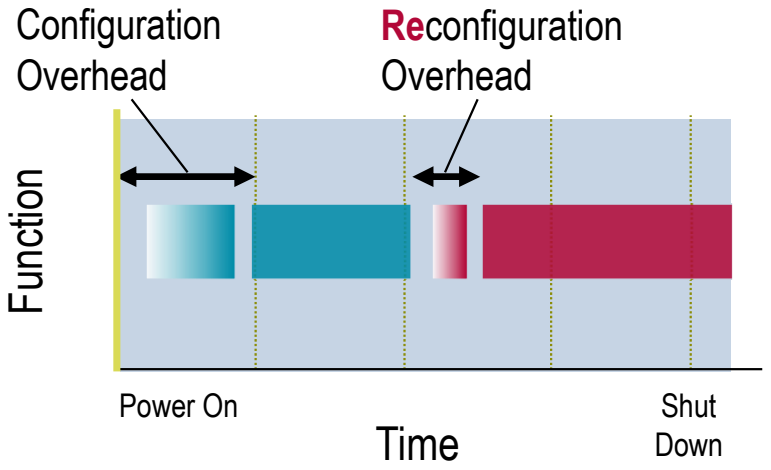
Partial Configuration



Partial Reconfiguration

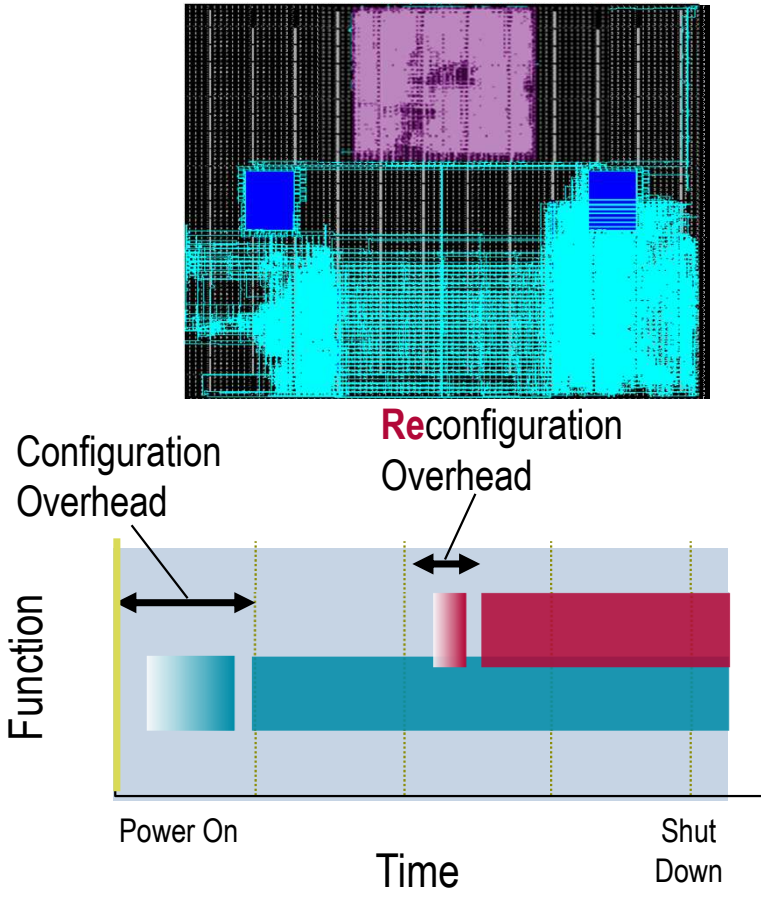
- Only a subset of configuration data is altered
- But all computation halts while modification is in progress…
- Main benefit:
  reduced configuration overhead





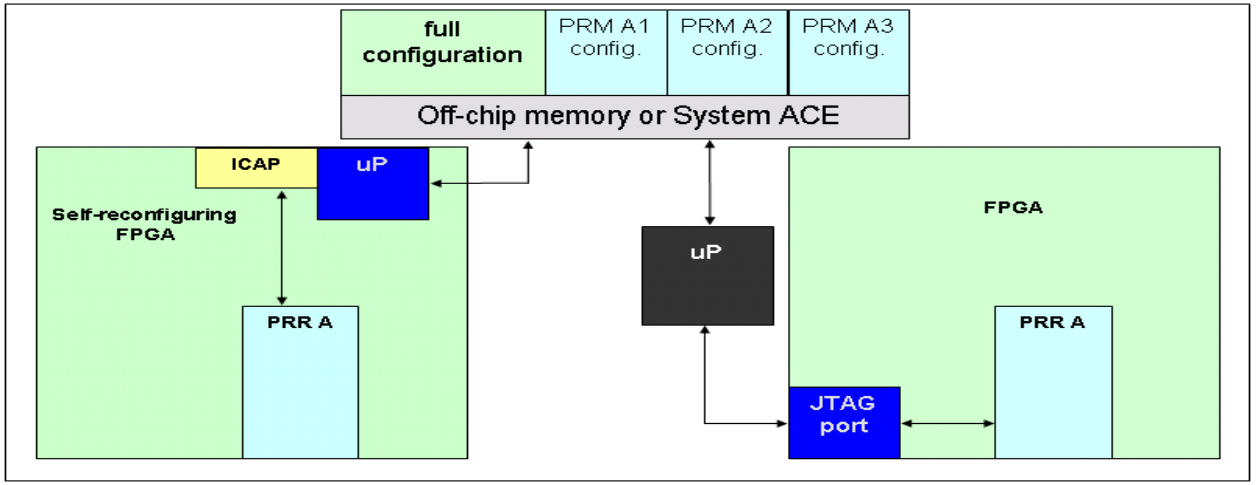
Dynamic Reconfiguration

- A subset of the configuration data changes…
- But logic layer continues operating while configuration layer is modified…
- Configuration overhead limited to circuit that is changing…



How Can We Reconfigure?

- **Initiation of reconfiguration is determined by the designer**
  - On-chip state machine, processor or other logic
  - Off-chip microprocessor or other controller
- **Delivery of the partial bit file uses standard interfaces**
  - FPGA can be partially reconfigured through the SelectMap, Serial or JTAG configuration ports, or the Internal Configuration Access Port
- **Logic decoupling should be synchronized with the initiation and completion of partial reconfiguration**
  - Enable registers
  - Issue local reset



Outline

- What is Partial Reconfiguration(PR)?
- PR Technology
- *PR Terminology*
- PR Design Flow
- Summary

Hierarchical Implementation Definitions

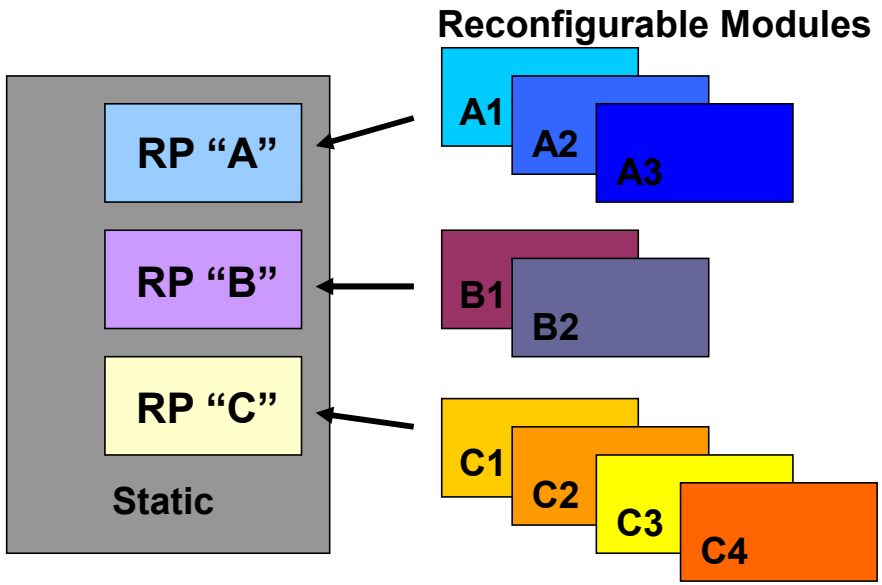
- **Partition**
  - A logical block (entity or instance) to be used for design reuse
  - User determines implementation versus preservation for each block
- **Bottom-up synthesis**
  - Separate synthesis projects resulting in multiple netlists or design checkpoints
  - No optimization across projects
- **Top-down synthesis; NOT used for Partial Reconfiguration (normal flow)**
  - One synthesis project where synthesis flattens design for optimization
  - Often called flat synthesis
  - No support for hierarchical implementation

Terminology

- **Reconfigurable Partition (RP)**
  - Design hierarchy instance marked by the user for reconfiguration
- **Reconfigurable Module (RM)**
  - Portion of the logical design that occupies the Reconfigurable Partition
  - Each RP may have multiple Reconfigurable Modules
- **Static Logic**
  - All logic in the design that is not reconfigurable
- **Configuration**
  - A full design consisting of Static Logic and one Reconfigurable Module for each Reconfigurable Partition
- **Partition Pins**
  - Ports on a Partition; Interface between Static and Reconfigurable Logic

Configurations

- **A Configuration is a complete FPGA design**
  - Consists of Static Logic and one variant for each reconfigurable instance
- **Maximum number of RMs for any RP determines minimum number of Configurations required**
  - Example: Possible Configurations for this design
    1. Static + A1 + B1 + C1
    2. Static + A2 + B2 + C2
    3. Static + A3 + B2 + C3
    4. Static + A3 + B2 + C4
  - Static Logic and repeated RMs are imported
  - Any combination of RMs can be selected to create unique full bit files



Reconfigurable Elements

- **What is reconfigurable?**
  - Nearly everything in the FPGA
    - Slice logic (LUTs, flip-flops, and carry logic, for example)
    - Memories (block RAM, distributed RAM, shift register LUTs)
    - DSP blocks
- **Logic that must remain in static logic includes**
  - Clock-modifying blocks (MMCM, DCM, PLL, PMCD)
  - Global clock buffers (BUFG)
  - Device feature blocks (BSCAN, ICAP, STARTUP, or PCIE, for example)
  - I/O components (IOLOGIC, IODELAY, IDELAYCTRL)

Reconfigurable Elements

- **Granularity of reconfigurable regions vary by device family**
  - Boundaries recommended, but not required, to align to Clock Regions
  - 7-Series (Vivado/ISE supported)
    - Slice region: 50 CLB high by 1 CLB wide
    - BRAM region: 10 RAMB36
    - DSP region: 20 DSP48
  - Virtex-6 (ISE supported)
    - Slice region: 40 CLB high by 1 CLB wide
    - BRAM region: 8 RAMB36
    - DSP region: 16 DSP48
  - Virtex-5 examples (ISE supported)
    - Slice region: 20 CLB high by 1 CLB wide
    - BRAM region: 4 RAMB36
    - DSP region: 8 DSP48
  - Virtex-4 examples (ISE supported)
    - Slice region: 16 CLB high by 1 CLB wide
    - BRAM region: 4 RAMB16 and 4 FIFO16
    - DSP region: 8 DSP48
  - Bit file sizes for each of these resource types will vary

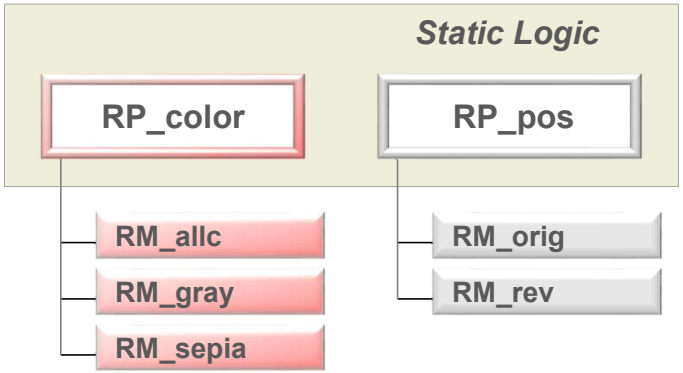
Outline

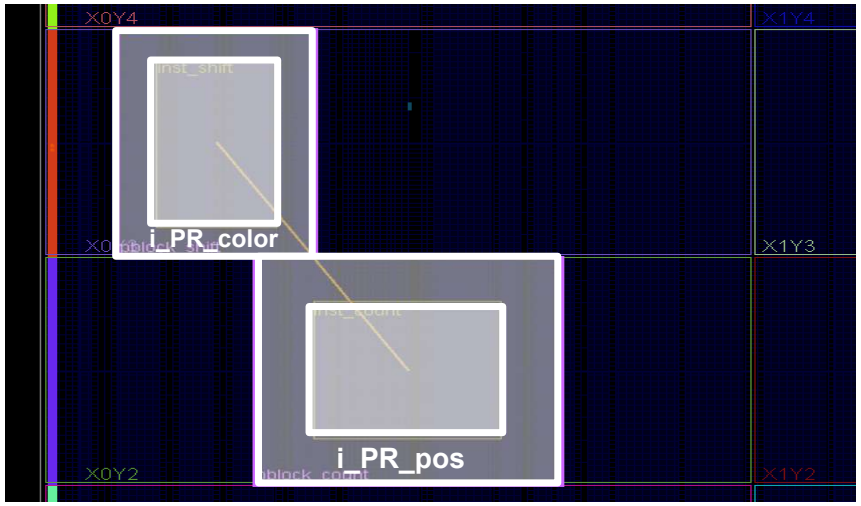
- What is Partial Reconfiguration(PR)?
- PR Technology
- PR Terminology
- *PR Design Flow*
- Summary

Intuitive Design Flow

Project Creation and Floorplanning

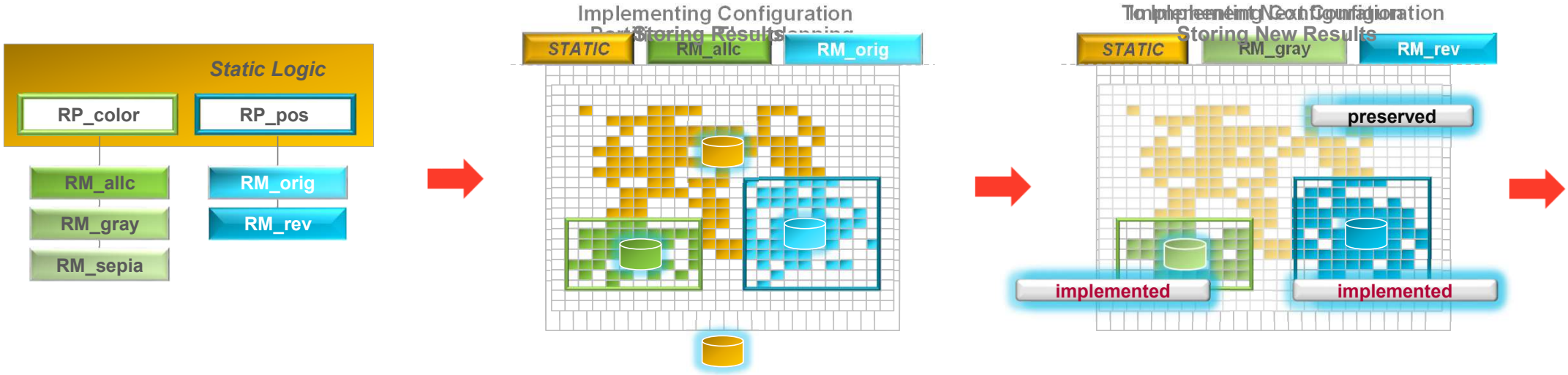
- **Structure your design**
  - Static Logic (unchanging design)
  - Reconfigurable Partitions (RP)
    - Instances to be reconfigured
  - Reconfigurable Modules (RM)
    - Functional variations for each RP
- **Synthesize bottom-up**
  - `synth_design -mode out_of_context`
- **Define resources to be reconfigured**
  - Pblocks map design modules to physical regions
    - Define XY ranges and resource types
- **Mark pblocks as reconfigurable**
  - `HD.RECONFIGURABLE` initiates flow





Leveraging Module Checkpoints for Partial Reconfiguration

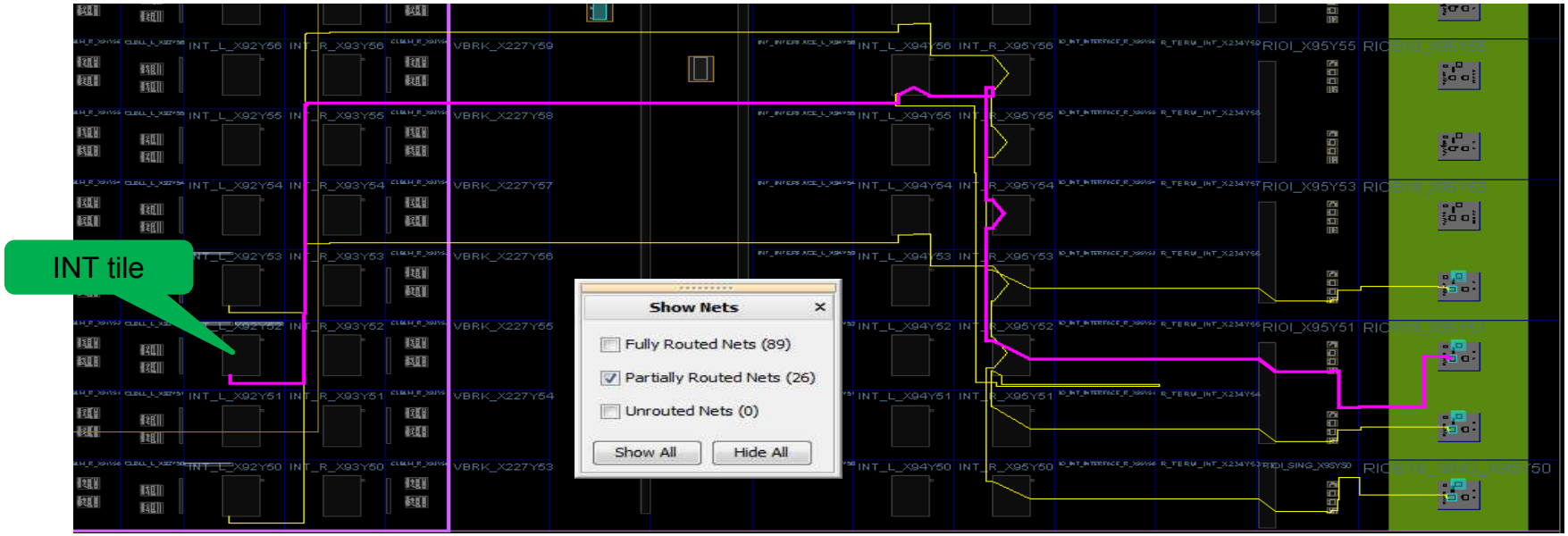
- **Partition methodology enables Partial Reconfiguration**
  - Allows clear separation of static logic and Reconfigurable Modules
  - Floorplan to identify silicon resources to be reconfigured
- **Design preservation accelerates design closure**
  - Lock static design database while implementing new modules



Vivado Software Features

No Proxy Logic Required

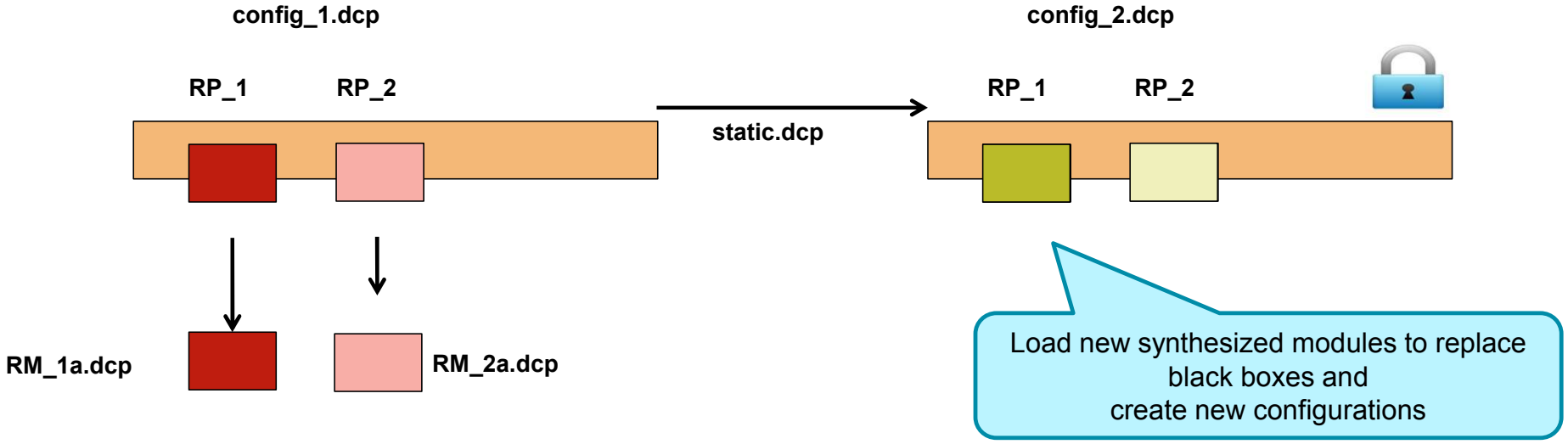
- **Partition Pins are junction between static and reconfigured logic**
  - Interface wires can be broken at interconnect tile site
  - “Anchor” between static and reconfigurable established mid-route
  - No overhead at reconfigurable partition interface
  - Decoupling logic still highly recommended



Vivado Software Design Management

Checkpoints for Each Partition

- **Vivado stores design data in checkpoints**
  - Save full design as a configuration checkpoint for bitstream creation
  - Save static-only checkpoint to be reused across multiple configurations
    - Routed static checkpoint can remain open in memory
    - Results are locked at the routing level
  - Reconfigurable modules can also be stored as their own checkpoints



Intuitive Design Flow

Implementation

- **Place and Route all design configurations**
  - Apply full design constraints in-context
  - Use normal timing closure, simulation and verification techniques
  - Currently non-project Tcl flow only – sample scripts provided
- **Final Verification**
  - Validates consistency of place and routed results across the entire system
- **Generate Bitstreams**
  - write_bitstream automatically creates all full and partial bit files

Configuration Details

- **Partial bit files are processed just like full bit files**
  - Bit file sizes will vary depending on region size and resource type
  - Contain just address & data, sync & desync words, optionally final and frame-based CRC value
    - No startup sequences, DONE flag

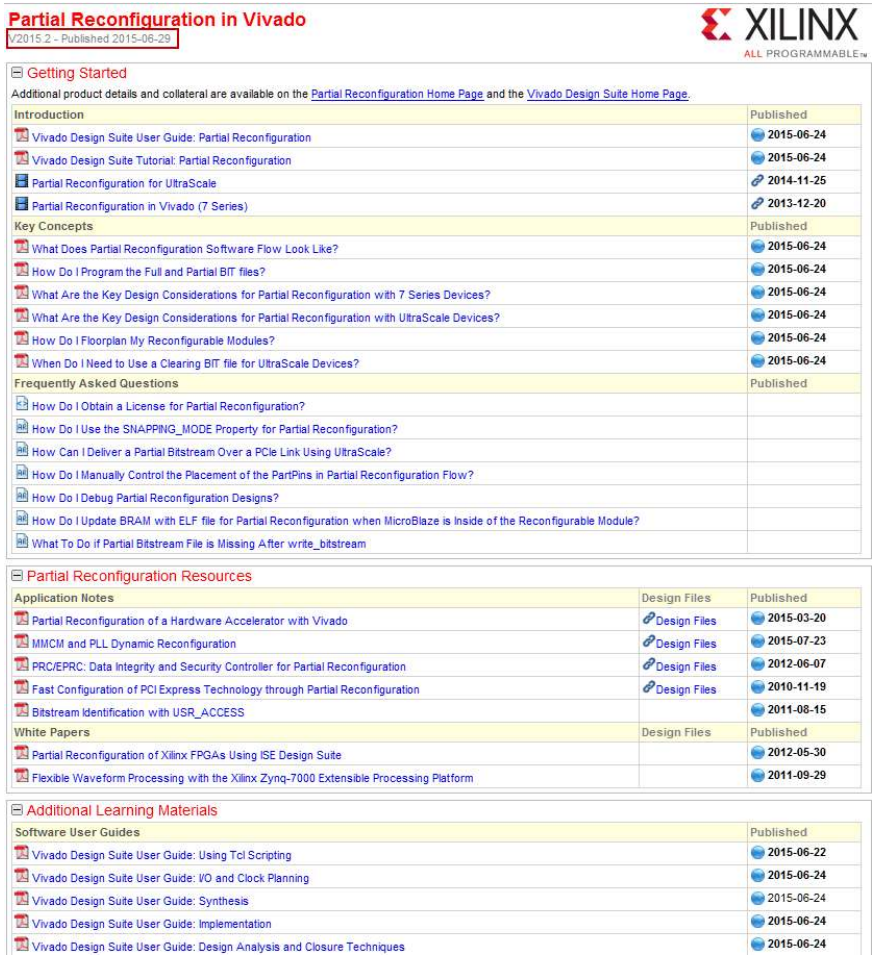
- **Partial Reconfiguration time depends on two factors:**
  1. Configuration bandwidth

Configuration Mode	Max Clock Rate	Data Width	Max Bandwidth
SelectMap / ICAP	100 MHz	32-bit	3.2 Gbps
Serial Mode	100 MHz	1-bit	100 Mbps
JTAG	66 MHz	1-bit	66 Mbps

  2. Partial bit file size
     - Estimate in Vivado, confirm in Rawbit file

Partial Reconfiguration Collateral

- **Learn about Partial Reconfiguration**
  - User Guide **UG909**
  - Tutorial **UG947**
  - PR Design Hub in DocNav
  - **XAPP1159**
  - More IP and references in the works
- **Read about Partial Reconfiguration**
  - Xcell articles in Issues 72, 73, 75, 77, 78, 79, 80, 84, 85



Partial Reconfiguration in Vivado

V2015.2 - Published 2015-06-29

XILINX ALL PROGRAMMABLE

Getting Started

Additional product details and collateral are available on the Partial Reconfiguration Home Page and the Vivado Design Suite Home Page.

Introduction	Published
Vivado Design Suite User Guide: Partial Reconfiguration	2015-06-24
Vivado Design Suite Tutorial: Partial Reconfiguration	2015-06-24
Partial Reconfiguration for UltraScale	2014-11-25
Partial Reconfiguration in Vivado (7 Series)	2013-12-20
Key Concepts	**Published**
What Does Partial Reconfiguration Software Flow Look Like?	2015-06-24
How Do I Program the Full and Partial BIT files?	2015-06-24
What Are the Key Design Considerations for Partial Reconfiguration with 7 Series Devices?	2015-06-24
What Are the Key Design Considerations for Partial Reconfiguration with UltraScale Devices?	2015-06-24
How Do I Floorplan My Reconfigurable Modules?	2015-06-24
When Do I Need to Use a Clearing BIT file for UltraScale Devices?	2015-06-24
Frequently Asked Questions	**Published**
How Do I Obtain a License for Partial Reconfiguration?	
How Do I Use the SNAPPING_MODE Property for Partial Reconfiguration?	
How Can I Deliver a Partial Bitstream Over a PCIe Link Using UltraScale?	
How Do I Manually Control the Placement of the PartPins in Partial Reconfiguration Flow?	
How Do I Debug Partial Reconfiguration Designs?	
How Do I Update BRAM with ELF file for Partial Reconfiguration when MicroBlaze is Inside of the Reconfigurable Module?	
What To Do if Partial Bitstream File is Missing After write_bitstream	

Partial Reconfiguration Resources

Application Notes	Design Files	Published
Partial Reconfiguration of a Hardware Accelerator with Vivado	Design Files	2015-03-20
MMCM and PLL Dynamic Reconfiguration	Design Files	2015-07-23
PRC/EPRC: Data Integrity and Security Controller for Partial Reconfiguration	Design Files	2012-06-07
Fast Configuration of PCI Express Technology through Partial Reconfiguration	Design Files	2010-11-19
Bitstream Identification with USR_ACCESS		2011-08-15
White Papers	**Design Files**	**Published**
Partial Reconfiguration of Xilinx FPGAs Using ISE Design Suite		2012-05-30
Flexible Waveform Processing with the Xilinx Zynq-7000 Extensible Processing Platform		2011-09-29

Additional Learning Materials

Software User Guides	Published
Vivado Design Suite User Guide: Using Tcl Scripting	2015-06-22
Vivado Design Suite User Guide: I/O and Clock Planning	2015-06-24
Vivado Design Suite User Guide: Synthesis	2015-06-24
Vivado Design Suite User Guide: Implementation	2015-06-24
Vivado Design Suite User Guide: Design Analysis and Closure Techniques	2015-06-24

Outline

- What is Partial Reconfiguration(PR)?
- PR Technology
- PR Terminology
- PR Design Flow
- *Summary*

Summary

- **Partial Reconfiguration is an Expert Flow**
- **Understanding PR terminology provides a commonality for PR design communication**
- **PR enables**
  - System flexibility
  - Size and cost reduction
  - Power reduction
- **The PR flow has four primary steps**
  1. Set up the design structure
  2. Constrain RPs and run DRCs
  3. Place & Route configurations
  4. Create bit files